Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Matemática
Programa de Mestrado Profissional
em Matemática em Rede Nacional

Inteligência Artificial e Aprendizado de Máquina: da teoria ao algoritmo pronto no Ensino Médio

Diogo Alves Brandão

Brasília

2023

Diogo Alves Brandão

Inteligência Artificial e Aprendizado de Máquina: da teoria ao algoritmo pronto no Ensino Médio

Dissertação apresentada ao Departamento de Matemática da Universidade de Brasília, como parte dos requisitos do Mestrado Profissional em Matemática em Rede Nacional - PROFMAT, para obtenção do grau de Mestre.

Universidade de Brasília - UnB

Departamento de Matemática - MAT

PROFMAT - SBM

Orientador: Prof. Dr. Vinícius de Carvalho Rispoli

Brasília

2023

Posição vertical

Diogo Alves Brandão
Inteligência Artificial e Aprendizado de Máquina: da teoria ao algoritmo pronto no Ensino Médio/ Diogo Alves Brandão. – Brasília, 2023-
105 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Vinícius de Carvalho Rispoli

Dissertação de Mestrado – Universidade de Brasília - UnB
Departamento de Matemática - MAT
PROFMAT - SBM, 2023.

1. Inteligência Artificial. 2. Aprendizado de Máquina 3. Ensino Médio. I. Vinícius de Carvalho Rispoli. II. Universidade de Brasília. III. PROFMAT - SBM. IV. Título.

CDU XYZ 02:141:005.7

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Matemática

Inteligência Artificial e Aprendizado de Máquina: da teoria ao algoritmo pronto no Ensino Médio

por

Diogo Alves Brandão

Dissertação apresentada ao Departamento de Matemática da Universidade de Brasília, como parte dos requisitos do Mestrado Profissional em Matemática em Rede Nacional - PROFMAT, para obtenção do grau de

MESTRE

Brasília, 14 de dezembro de 2023

Comissão Examinadora:

Prof. Dr. Vinícius de Carvalho Rispoli - FGA/UnB (Orientador)

Prof. Dr. José Eduardo Castilho - FUP/UnB (Membro Interno)

Prof. Dr. Osmar Aléssio - UFTM (Membro Externo)

"Na adversidade, uns desistem, enquanto outros batem recordes"
Ayrton Senna

Agradecimentos

Esta seção foi minuciosamente planejada desde o primeiro momento do curso. Por anos aguardei ansiosamente para escrever as palavras que se seguem. O motivo de tal ânsia é pelo fato de aqui está registrado a gratidão por pessoas especiais que, direta ou indiretamente, contribuíram com essa peça.

De início, rendo graças a Deus por ter me abençoado desde meu nascimento (prematuro de 7 meses, pesando apenas 1.940 gramas, com anemia e icterícia, além de posterior ventilação assistida, transfusão de sangue e reanimação cardíaca) até os dias atuais. Já que citei o período anterior, é salutar destacar que eu acredito e confio em Deus, mas isso não afasta a responsabilidade delegada a cada um de nós (conhecida como livre arbítrio), tão tacitamente relegada atualmente. Acredito como verdade indubitável que a prática repetida com o tempo é capaz de influenciar o rumo de nossas vidas, em todos os sentidos. Há quinze anos internalizei que sucesso não é um evento, mas um hábito. Portanto, diuturnamente, escolho uma vida mais fisicamente ativa e dieteticamente saudável (sem puritanismo). Aliás, a própria etimologia da palavra dieta significa um hábito e não evento como se costuma empregar. Não obstante, há quatro anos acrescentei inteligência emocional a minha rotina.

Apesar das mazelas de nosso país, devo gratidão a República Federativa do Brasil e a Secretaria de Estado de Educação do Distrito Federal que concederam oportunidade de realizar todo o curso com dispensa das atividades laborais, sem prejuízo pecuniário. Se atualmente posso estudar e pesquisar é porque tenho as necessidades fisiológicas, como alimentação, atendidas. Infelizmente, milhões de pessoas sequer têm suas necessidades básicas contempladas, mormente diante do cenário bélico atual. Finalmente, se a você foi concedido vida e disposição, busque sua melhor versão todos os dias questionando como pode ser melhor a cada dia.

Agradeço à minha belíssima esposa Karina, pessoa determinada, sábia e inteligente, que tem transformado os momentos simples em inesquecíveis. Obrigado pelo apoio incondicional, pelos aconselhamentos assertivos e pela compreensão necessária para que eu possa me dedicar aos estudos e a esta obra. Todas as manhãs, ela ensina-me, pelo exemplo, antes mesmo da aurora, como não se esquivar das obrigações diárias. As viagens tornam-se coloridas e cheias de vida com sua companhia.

Agradeço veementemente aos meus pais, Norisvaldo e Marisa. Sou eternamente grato a ambos. Ele é afável, acolhedor, sempre disposto a ajudar e me incentivar. É o idoso, com disposição de jovem, que não entende que apenas um esporte já é difícil o suficiente. Neste sentido, ele está a praticar vários quilômetros de natação, ciclismo e cor-

rida em uma única prova (*ironman*), meu exemplo da busca pela atividade física. Ela é guerreira, meu exemplo da busca pela alimentação saudável. Pessoa determinada, destemida e comunicativa, inspira-me na racionalidade dos gastos e na dedicação incansável aos cuidados educacionais e dos animais. Obrigado pelas discussões e conselhos, os quais me fazem uma pessoa melhor. A ambos agradeço por compartilhar as experiências de morar em Portugal.

Agradeço ao meu irmão Douglas, destemido, decidido e inteligente, me inspira grandemente a ser uma pessoa capaz e independente. Sempre que preciso de motivação, observo como meu irmão piloto lida com determinação, capacidade e preparo. Agradeço a minha irmã Lara Letícia, pessoa muito inteligente, naturalmente habilidosa em matemática e inglês, o prodígio da família. Muito obrigado LL pelos momentos de diversão nos games, que são tão importantes quanto as obrigações.

Agradeço a esta universidade por edificar minha graduação e mestrado. Agradeço a existência do PROFMAT, a Sociedade Brasileira de Matemática e a excelência do seu corpo docente, em especial desta universidade. Agradeço ao meu orientador, sempre disponível, cirúrgico e rápido em me ajudar. Muito obrigado por conceder autonomia e viabilizar tempo adicional para esta obra. Espero que em breve possamos trabalhar juntos novamente.

Muito obrigado aos meus familiares e amigos que me ajudam a viver em plenitude. Adriano Batista, Bruno Liberato, Daniel Sousa, Édion Alberto, Pedro Henrique, Thiago Henrique e Tiago Wiliam são alguns dos meus amigos que completam meu dia. Muito obrigado aos amigos da academia Haguihara, reduto de pessoas educadas e focadas no treinamento, em especial ao Marcelo por me receber por tantos anos no quintal de sua casa. Pude amadurecer ideias desta obra com amigos de treino, entre eles Ernesto Giovenardi. Agradeço a Benedito Mendes, Bruno Mata, Christian Gomes, Helio Escaleira, Saymon Hemkemaier, entre outras pessoas que deixam meu treino mais agradável.

Agradeço ao casal Zavatti pela hospitalidade, confiança e oportunidade em me receber e acolher no seu lar. Muito obrigado por viabilizar quatro inesquecíveis meses em Houston, Texas. Pude estudar e pesquisar como se estivesse em casa, com o privilégio de acompanhar o nascimento da encantadora Catarina. Muito obrigado a equipe do Wanderstay pelo compartilhamento cultural e por fornecer inspiração. Dee, I want to thank you for your confidence in me, because you allowed me to take care of your cozy hostel. Ariyanna Jaimes, thank you so much for teaching me and encouraging me to learn English. I would also like to thank both a thousand times for sharing great times.

Agradeço à escola CEM 111 do Recanto das Emas, em especial André Gustavo, Andreza Sudré, Bruno Leonardo, Flávia Cristina e Larissa Nascimento. Um agradecimento muito especial à escola CED 06 do Gama, onde pude fazer grandes amigos e construir laços laborais. Agradeço em especial ao Sebastião pelas palavras de incentivo. Agradeço

aos amigos Adriane Arruda, Bruno Leonardo, Cristiano Luz, Fabiana Angélica, Hailton Junior, Hugo Dutra, Jânio Café, Leonardo Miranda, Sandro Carvalho, Sheila Rezende e vários outros que entre os anos de 2015 e 2021 estavam no CED 06. Muito obrigado por acreditar e fomentar meus estudos.

Agradeço aos não muitos seguidores do meu perfil no instagram @diogo_di_ e do canal Matemática Fitness no youtube, que reverberam nossos trabalhos e instigam a produção de conteúdo digital de qualidade. Nos dias atuais, é inegável que as redes sociais são ferramentas poderosas para a divulgação de obras.

Resumo

Esta obra aborda o Aprendizado de Máquina da extensão teórica ao funcionamento prático, com ênfase no algoritmo classificador da Máquina de Suporte Vetorial, associado à matemática do Ensino Médio, com a finalidade de favorecer a plena formação educacional. Este trabalho fundamenta-se nas novas estratégias e diretrizes pedagógicas que direcionam para o uso e implementação do Aprendizado de Máquina como um recurso didático e tecnológico da matemática. A abordagem integrada inclui vários conceitos, adversidades e métodos avaliativos do Aprendizado de Máquina, além de promover a expansão artificial do conjunto de treinamento e redução de dimensionalidade. É descrito o funcionamento do modelo linear, de margem suave e com o truque do kernel, incluindo dois exercícios teóricos para o discente. É produzido um brevíssimo tutorial sobre a linguagem de programação Python como suporte da implementação. São propostas seis etapas de implementação com Máquina de Suporte Vetorial, em cada há métricas, curvas e matrizes para avaliação do desempenho.

Palavras-chaves: Inteligência Artificial. Aprendizado de Máquina. Ensino Médio.

Abstract

This work addresses the Machine Learning from theory to hands-on with an emphasis on Support Vector Machine classification algorithm associated with high school mathematics in order to improve knowledge and skills. This dissertation is based on new strategies and pedagogical guidelines that direct to the use and implementation of Machine Learning as a didactic and technological resource in Mathematics. Our integrated approach includes many concepts, challenges and performance metrics in Machine Learning, also builds data augmentation and dimensionality reduction. This work describes how work linear, soft margin and kernel trick models, including two theoretically-based activities for the students. This dissertation contains a short Python tutorial in order to help the implementation. Six project steps are proposed with the Support Vector Machine, each one having performance evaluations by metrics, graphics and matrices.

Key-words: Artificial Intelligence. Machine Learning. High School.

Lista de ilustrações

Lista de abreviaturas e siglas

SVM	Máquina de Suporte Vetorial
ML	Aprendizado de Máquina
IA	Inteligência Artificial
DCNEM	Diretrizes Curriculares Nacionais do Ensino Médio
BNCC	Base Nacional Comum Curricular
OCDE	Organização para a Cooperação e Desenvolvimento Econômico
FMI	Fundo Monetário Internacional
ROC	Curva de característica de operação do receptor
MNIST	Banco de dados Modified National Institute of Standards and Technology database
PCA	Análise de Componentes Principais
OvR	Classificador um contra todos (One-versus-Rest)
OvO	Classificador um contra um (One-versus-One)

Sumário

1 INTRODUÇÃO

Talvez a primeira indagação que venha à mente do leitor quando inicia o estudo da inteligência artificial seja, a propósito, sua própria existência. Haverá uma máquina que pense como seres humanos e que seja mais inteligente?

Essa pergunta foi respondida por Richard Feynman, renomado físico e vencedor do prêmio Nobel, em uma palestra realizada em setembro de 1985 (VEISDAL, 2019). Inicialmente, afirmou que as inteligências deveriam ser definidas para que a pergunta seja uma questão, faremos isso adiante. Considerando o conceito de inteligência como capacidade artificial de reter informações e pensamento como capacidade natural de refletir sobre situações inesperadas ou indefinidas, o autor afirma que as máquinas não vão pensar como seres humanos, mas serão mais inteligentes. A pergunta inicial não tem resposta unânime, mesmo entre os renomados estudiosos da área. Todavia, chama a atenção a resposta formulada por Feynman há mais de 38 anos. Não obstante, a inteligência humana é a referência que os pesquisadores aspiram alcançar com a Inteligência Artificial. Compreender e reproduzir como uma mera porção de matéria orgânica faz atividades de grande complexidade é ponto central.

O parágrafo anterior evidencia a relevância da temática e dos elementos que as constituem. As instruções que o humano envia à máquina necessitam de aparatos matemáticos. Número, operador e lógica são amplamente utilizados como elementos dos códigos usados para programar. Neste sentido, a linguagem de programação permite que a máquina execute um texto injuntivo. A matemática foi, e ainda é, a ciência elementar para a base da computação. Os fundamentos da matemática permeiam e pululam na inteligência artificial.

O presente trabalho explora o aprendizado de máquina, mormente a Máquina de Suporte Vetorial (SVM), para ser apresentado desde o modelo teórico ao algoritmo pronto no ensino médio. No entanto, insta salientar que no título as palavras “no Ensino Médio” não foram colocadas antes dos dois-pontos, pois parte da demonstração do funcionamento do aprendizado de máquina abordado não se restringe aos conhecimentos em nível de ensino médio, há seção destinada ao docente com o uso de matemática superior, necessária para o profundo entendimento. Não obstante, entre teoria e prática, foram propostos exercícios teóricos em nível de ensino médio. Almeja-se disseminar o aprendizado de máquina como recurso didático e tecnológico que favoreça a plena formação educacional, contribuindo com as mudanças e formulações das novas estratégias e diretrizes pedagógicas.

1.1 Objetivo Geral

Orientar o uso estruturado, da teoria à implementação, do Aprendizado de Máquina como um recurso didático e tecnológico associado à matemática do Ensino Médio.

1.2 Objetivos Específicos

- Identificar as legislações educacionais que permitem a aplicabilidade do projeto;
- Caracterizar os conceitos do Aprendizado de Máquina com ênfase no SVM;
- Expor métodos avaliativos e adversidades do Aprendizado de Máquina;
- Descrever o funcionamento aprofundado do SVM para o corpo docente;
- Resolver exercícios teóricos do SVM em nível de Ensino Médio;
- Produzir um tutorial lacônico sobre a utilização da linguagem de programação Python para a implementação;
- Implementar o Aprendizado de Máquina, por meio do SVM, utilizando a linguagem de programação Python.

1.3 Como o trabalho está organizado

A dissertação está estruturada em 8 Capítulos. O Capítulo 1 cuida de introduzir esta dissertação, elencando os objetivos e organização. O Capítulo 2 identifica as normas legais, incluindo as que permitem a aplicabilidade do projeto e do tema proposto no ensino médio. O Capítulo 3 apresenta conceitos, categorias, métodos avaliativos e adversidades do aprendizado. O Capítulo 4 explora a Máquina de Suporte Vetorial, descreve e elucida a matemática por detrás do modelo linear, não linear e com o Truque do Kernel. O Capítulo 5 resolve exemplos teóricos em nível de ensino médio. O Capítulo 6 produz um brevíssimo tutorial sobre o uso de Python para a implementação. O Capítulo 7 implementa o Aprendizado de Máquina várias vezes com sistemas que reconhecem o dígito manuscrito, inclusive o escrito pelo discente. O Capítulo 8 cuida das considerações finais.

2 LEGISLAÇÃO PERTINENTE AO PROJETO

Citaremos lei, resoluções, pareceres, portarias e outros documentos normativos que permitem a aplicabilidade do projeto e do tema proposto no Ensino Médio. Ademais, no tempo em que escrevo, há amplo debate acerca de Inteligência Artificial (IA) no Senado Federal.

De início, a Constituição Federal de 1988 garante a educação como direito social. Além disso, estabelece competência privativamente à União legislar sobre diretrizes e bases da educação nacional. Por outro lado, proporcionar os meios de acesso à educação é competência comum da União, dos Estados, do Distrito Federal e dos Municípios. Não obstante, a Carta Magna contempla uma seção exclusivamente dedicada à educação, o qual não será objeto de estudo neste trabalho, igualmente à Lei n 9.394/1996, que estabelece as diretrizes e bases da educação nacional.

2.1 DCNEM

As Diretrizes Curriculares Nacionais para o Ensino Médio (DCNEM) foram atualizadas pela resolução nº 3, de 21 de novembro de 2018 do Conselho Nacional da Educação, tendo em vista a paridade com o avanço tecnológico. Nota-se uma preocupação maior com inclusão dos assuntos abordado neste trabalho. Abaixo transcrevemos parte do documento.

> Art. 8º As propostas curriculares do ensino médio devem:
> (...)
> II - garantir ações que promovam:
> (...)
> b) cultura e linguagens digitais, pensamento computacional, a compreensão do significado da ciência, das letras e das artes, das tecnologias da informação, da matemática, bem como a possibilidade de protagonismo dos estudantes para a autoria e produção de inovação.

Além disso, delimita a disciplina que abarca o tema.

> Art. 12. A partir das áreas do conhecimento e da formação técnica e profissional, os itinerários formativos devem ser organizados, considerando:
> (...)

> II - matemática e suas tecnologias: aprofundamento de conhecimentos estruturantes para aplicação de diferentes conceitos matemáticos em contextos sociais e de trabalho, estruturando arranjos curriculares que permitam estudos em resolução de problemas e análises complexas, funcionais e não-lineares, análise de dados estatísticos e probabilidade, geometria e topologia, robótica, automação, **inteligência artificial**, programação, jogos digitais, sistemas dinâmicos, dentre outros, considerando o contexto local e as possibilidades de oferta pelos sistemas de ensino. (grifo nosso)

2.2 BNCC

A Base Nacional Comum Curricular (BNCC) é um documento normativo que define o conjunto orgânico e progressivo de aprendizagens essenciais que todos os alunos devem desenvolver ao longo das etapas e modalidades da Educação Básica. Nela há 9 menções do termo "pensamento computacional", das quais destaca-se o trecho a seguir que tem íntima relação com este projeto: "associado ao pensamento computacional, cumpre salientar a importância dos algoritmos e de seus fluxogramas, que podem ser objetos de estudo nas aulas de Matemática." (BRASIL, 2018, p. 271). Não obstante, a própria norma disserta sobre a temática pensamento computacional, asseverando que "envolve as capacidades de compreender, analisar, definir, modelar, resolver, comparar e automatizar problemas e suas soluções, de forma metódica e sistemática, por meio do desenvolvimento de algoritmos" (BRASIL, 2018, p. 474).

Neste sentido, ratifica a possibilidade de o estudante utilizar, propor e/ou implementar soluções envolvendo diferentes tecnologias para solucionar problemas complexos explorando o pensamento computacional. Além disso, destaca que a área de matemática e suas tecnologias deve dispor especial importância às tecnologias digitais e aplicativos para dar continuidade ao desenvolvimento do pensamento computacional.

Por fim, elencamos os códigos das habilidades em cada competência específica na área de matemática e suas tecnologias que guardam fortes relações com este trabalho. Em competência específica 1: EM13MAT101, EM13MAT102, EM13MAT104 e EM13MAT105. Em competência específica 2: EM13MAT202. Em competência específica 3: EM13MAT302, EM13MAT311 e EM13MAT315. Em competência específica 4: EM13MAT405 e EM13MAT407. Finalmente, em competência específica 5: EM13MAT501, EM13MAT503 e EM13MAT510.

Ainda sobre o tema, a resolução nº 1/2022, de 4 de outubro de 2022, do Ministério da Educação por meio da Câmara de Educação Básica do Conselho Nacional de Educação, trata de normas sobre computação na educação básica em complemento à BNCC e define a necessidade de políticas para o apoio ao desenvolvimento de recursos didáticos compatíveis com as habilidades listadas anteriormente.

Insta citar o parecer nº 2/2022, aprovado em 17 de fevereiro de 2022, do Ministério da Educação por meio da Câmara de Educação Básica do Conselho Nacional de Educação, que trata de normas sobre computação na educação básica em complemento à BNCC e disserta sobre a importância da Inteligência Artificial e Aprendizado de Máquina na educação básica.

2.3 Currículo em Movimento do Novo Ensino Médio do Distrito Federal

O Currículo em Movimento do Novo Ensino Médio do Distrito Federal apresenta as perspectivas curriculares para o Ensino Médio no âmbito do Distrito Federal. Neste sentido, o Currículo assevera que "...ao chegarem aos Anos Finais, [os alunos] possam ser estimulados a desenvolver o pensamento computacional, por meio da interpretação e da elaboração de algoritmos..." (BRASÍLIA, 2020, p. 78). Esta edição do Currículo destacou a importância das tecnologias digitais para o ensino da matemática, mormente por trazer subsídio para o ensino.

Além disso, destaca-se os seguintes objetivos de aprendizagem em matemática. MAT57FG: Descrever, por meio de um fluxograma, quando possível, um algoritmo que resolva um problema; MAT58FG: Utilizar pseudocódigo como linguagem inicial para estruturar um algoritmo que posteriormente poderá ser formalizado com sintaxe padrão; MAT59FG: Estruturar correspondências entre o algoritmo e as sintaxes da linguagem de programação escolhida.

2.4 Marco Regulatório da Inteligência Artificial

Na atualidade, há um projeto de lei tramitando no Congresso Nacional que regulamenta a IA no Brasil, estabelecendo princípios, direitos e deveres para o uso de IA. A norma, denominada Marco Regulatório da IA, encontra-se em amplo debate no Senado Federal. Não é nosso escopo discorrer sobre o assunto. Não obstante, foi citado a norma, pois é mister destacar a oportunidade para o corpo docente trabalhar de forma interdisciplinar. No projeto de lei há, por exemplo, a definição de um sistema de IA, vale a leitura.

3 APRENDIZADO DE MÁQUINA

Uma questão que surge ao iniciar o estudo é sobre a definição de aprendizado de máquina. Desde o início da humanidade, o homem busca motivação para novas descobertas na natureza. A inspiração do aprendizado de máquina é o funcionamento do cérebro humano. O prodígio Alan Turing - conhecido por propor um experimento para diferenciar humano de máquina, o famoso teste de Turing – foi ainda mais preciso, ele sugeriu que, ao invés de tentar produzir um programa para simular a mente de um adulto, seria melhor tentar produzir um que simulasse a mente de uma criança e submetê-lo ao processo de educação (TURING, 1950). O "processo de educação", tal como dito por Turing, é similar ao aprendizado de máquina. Não obstante, acerca das metas e limitações, o Livro da Matemática (KINDERSLEY, 2020, p. 301) informa que

> Técnicas como aprendizado de máquina, em que IA programam a si mesmas por tentativa e erro, e sistemas especialistas, em que a IA recorre a um banco de dados de conhecimento fornecido por programadores humanos, ampliaram as habilidades de IA. Apesar disso, a maioria são **IAs estreitas**, pois se destinam a fazer uma tarefa muito bem, melhor que um humano, mas não aprendem a fazer nada além e não têm consciência do que não sabem. Uma **IA geral** que possa dirigir o próprio aprendizado do mesmo modo que uma inteligência evoluída (como a humana) é o próximo objetivo da ciência da computação. (grifo nosso)

Neste sentido, abaixo alguns termos elementares:

Dados ou conjunto de dados são as amostras ou os exemplos fornecidos à máquina, é a matéria prima básica do aprendizado, equivalente a fonte de conhecimento, que pode servir para instruir. Assim, salvo em aprendizado por reforço, o conjunto de dados é essencial e geralmente muito grande. O conjunto de dados é formado pela união do conjunto de treinamento e conjunto de teste. "Os dados podem vir da natureza, produzido pelo homem ou gerado por outro algoritmo" (BURKOV, 2019, p. 1, tradução nossa).

Algoritmos fundamentais são os modelos estatísticos e matemáticos preestabelecidos que servem para generalizar os dados e fundamentar as decisões. São exemplos de algoritmos fundamentais: k-Nearest Neighbors, Naive Bayes, Máquina de Suporte Vetorial, Redes Neurais, entre outros.

Treinamento é o processo que, a partir do algoritmo fundamental e alguns exemplos (chamado de conjunto de treinamento), ajusta o algoritmo nos exemplos fornecidos, para que com isso faça melhores previsões de exemplos nunca vistos. "Esse processo pode ser mensurado por minimizar a função de perda, ou maximizar outras funções" (WILMOTT,

2019, p. 35, tradução nossa).

Modelo geralmente se refere ao algoritmo fundamental já treinado para determinada aplicação. Em geral, um modelo precisa prever bem para dados completamente desconhecidos. Não raras vezes, é sinônimo de algoritmo fundamental.

Aprendizado de máquina (ML), do inglês *Machine Learning*, "é ramo da inteligência artificial que tem a capacidade de obter novos conhecimentos com a experiência e realizar atividades não definidas explicitamente em suas instruções programadas" (RAHMAN, 2022, p. 36). Outra definição com termos precisos, orientada à ciências exatas, foi proposta da forma: "alega-se que um programa de computador aprende pela experiência E em relação a algum tipo de tarefa T e alguma medida de desempenho P se o seu desempenho em T, conforme medido por P, melhora com a experiência E" (MITCHELL, 1997, p. 2, tradução nossa). A máquina melhora (ou deveria) com novas informações. É como um atleta de alto nível que melhora (ou deveria) com o treinamento.

3.1 Categorias do aprendizado

Existem algumas categorias do ML, das quais destacamos três. Em duas categorias, o importante é a quantidade de ajuda (informações a respeito dos exemplos) que fornecemos à máquina. Na outra categoria, ensinamos a máquina a fazer coisas, por meio de recompensas e punições.

3.1.1 Aprendizado supervisionado

No aprendizado supervisionado, a fase de treinamento deve incluir não somente o conjunto de treinamento, mas também a informação relevante sobre cada exemplo. Normalmente, chamamos essa informação relevante de etiqueta, rótulo ou *label*. Em outras palavras, para o treinamento fornecemos à máquina o que seria a solução da previsão de cada exemplo. Por exemplo, suponha que a máquina deva reconhecer qual é o algarismo escrito à mão, para isso, devemos antes treiná-la. Neste sentido, é necessário que no treinamento seja fornecido também qual é o dígito em cada imagem. Neste trabalho, a ênfase é em aprendizado supervisionado.

3.1.2 Aprendizado não supervisionado

No aprendizado não supervisionado, os dados de treinamento não são rotulados. A máquina deve procurar por padrões ou relações, sem indicação clara do que é relevante. Por exemplo, tarefas de detecção de anomalias quando a instituição financeira ou de pagamento bloqueia transações incomuns em cartões de crédito para evitar fraudes.

3.1.3 Aprendizado por reforço

No aprendizado por reforço, o agente (nessa categoria, a máquina é comumente chamada de agente) pode assistir e interagir com o ambiente. A idéia central do agente é executar ações e coletar *feedback*, a fim de obter o maior número de recompensas. É comum em programas que aprendem a vencer jogos, por exemplo, os pontos de vitória no final de um jogo de xadrez informam ao agente que fez a coisa certa. Neste sentido, o programa AlphaGo da DeepMind ficou famoso em 2017 por vencer o campeão mundial em um jogo de tabuleiro. Ele aprendeu analisando milhões de jogos e, após desativado o aprendizado, jogou contra o campeão.

3.2 Aprendizado baseado em modelo

A maioria das atividades em aprendizado de máquina faz generalizações. Em outras palavras, o que justifica o uso de um algoritmo de ML é o fato de o algoritmo prever bem. Existem duas abordagens principais no que se refere à generalização.

Aprendizado baseado em instância é aquele em que a máquina aprende por meio da memorização e então generaliza para novos casos empregando uma medida de similaridade.

Aprendizado baseado em modelo é aquele em que a partir de um conjunto de exemplos é construido um modelo que os represente e utiliza-se esse modelo para fazer previsões. Neste artigo, a ênfase de generalização é em aprendizado baseado em modelo.

3.2.1 Conjunto de treinamento

É a parcela do conjunto de dados que serve como base para o treinamento do modelo. Ou seja, são os exemplos a partir do qual o modelo é construido. Em geral, representa cerca de 75-80% do conjunto de dados.

3.2.2 Conjunto de teste

É a parcela do conjunto de dados que serve como base para comprovar que o modelo funciona bem. São exemplos nunca visto antes que avaliam o modelo. Em geral, representa cerca de 20-25% do conjunto de dados.

3.2.3 Objetivo do treinamento

Os **parâmetros** "são variáveis que definem o modelo e são modificados diretamente com base nos dados de treinamento" (BURKOV, 2019, p. 19, tradução nossa). Neste sentido, para este projeto, o objetivo do treinamento é encontrar os coeficientes que ajustam o algoritmo fundamental e faz dele um ótimo modelo em algum sentido.

Por exemplo, suponha que esteja em busca de resposta para: dinheiro traz felicidade às pessoas? Assim, reune os dados do Índice de Vida Melhor, disponível no site da Organização para a Cooperação e Desenvolvimento Econômico (OCDE), e as informações sobre o produto interno bruto (PIB) per capita, disponível no site do Fundo Monetário Internacional (FMI), a fim de construir o gráfico da Figura 1.

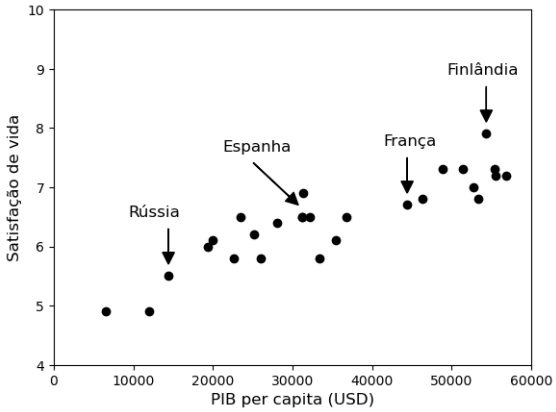


Figura 1 – Dinheiro traz felicidade às pessoas?

Fonte: Elaborada pelo autor.

Vamos assumir que existe uma tendência, pois parece que a satisfação de vida aumenta linearmente à medida que o PIB per capita do país cresce. Inicialmente escolhemos o algoritmo regressão linear para ser treinado. Assim, o objetivo do treinamento é encontrar os parâmetros (coeficientes) θ_0 e θ_1, tais que:

$$\text{satisfação de vida } = \theta_1 \times \text{ PIB per capita } + \theta_0$$

Para a busca dos parâmetros, iremos minimizar a função que calcula a distância entre as previsões do modelo e os exemplos de treinamento (função de custo).

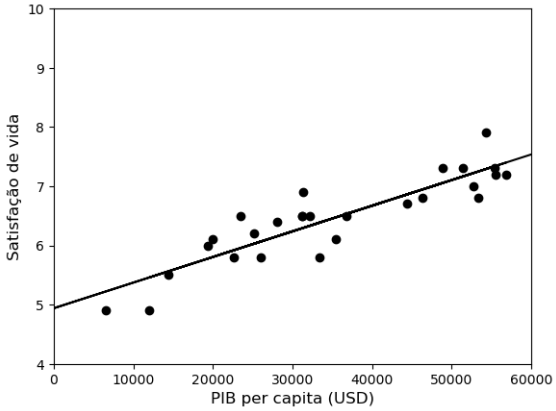


Figura 2 – Regressão Linear

Fonte: Elaborada pelo autor.

A Figura 2 mostra o modelo regressão linear treinado com os dados. Neste caso, $\theta_1 = 4,3277 \times 10^{-5}$ e $\theta_0 = 4.9394$. Isto é, até aqui, o dinheiro traz felicidade!

Observe que minimizar a função que calcula sua distância até os exemplos nem sempre é o almejado. Por vezes, buscaremos o oposto: a função que maximiza sua distância até alguns exemplos de treinamento (explicaremos com mais detalhe em breve).

3.2.4 Sobreajuste

A Figura 3 mostra um modelo polinomial excessivamente ajustado aos dados anteriores. Conhecido como sobreajuste, o modelo se adequa tanto aos dados de treinamento que acaba tendo ótimo desempenho no conjunto de treinamento, mas generaliza mal para novos exemplos.

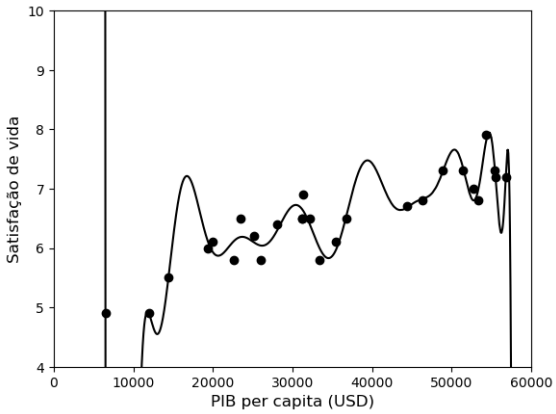


Figura 3 – Modelo com sobreajuste nos dados de treinamento

Fonte: Elaborada pelo autor.

GÉRON (2019, p. 24) chama de "regularização quando restringe um modelo para simplificar e reduzir o risco de sobreajuste." Neste sentido, o modelo linear definido anteriormente possui dois parâmetros, θ_0 e θ_1. Poderíamos regularizar o modelo restringindo $\theta_1 = 0$, neste caso teríamos uma média. Ou ainda, regularizar o modelo da figura acima, restringindo o grau do polinômio.

Não obstante, BURKOV (2019, p. 18) afirma que um "**hiperparâmetro** é (quase sempre) um valor que influencia o jeito do algoritmo trabalhar. Eles não são aprendidos pelos próprios algoritmos e devem ser ajustados antes do treinamento."

3.2.5 Teste e validação do modelo

A maneira inequívoca de avaliar se um modelo generalizará bem em casos novos é testá-lo na prática. Porém, colocar o modelo em produção pode ser desastroso caso não funcione bem. Possívelmente, muitos usuários reclamarão acarretando risco reputacional

à organização. Não é o que esperamos, (nem o administrador de empresas) portanto devemos encontrar meios que mensurem os erros e indiquem possíveis melhorias.

É comum dividir os dados em dois conjuntos: conjunto de treinamento e conjunto de teste. O conjunto de treinamento é usado para a construção do modelo, e eventualmente, subdividido para validação e ajustes do modelo. Por outro lado, o conjunto de teste não pode ser usado para o treinamento. Ele serve como casos completamente desconhecidos para o modelo, perfeito para simular o meio externo, em geral, depois que você já tem alguma confiança no modelo. Adiante, vamos trazer algumas métricas e formas de avaliar o modelo que podem conduzir para melhorias.

3.2.5.1 Matriz de confusão

Usada em algoritmos de classificação, a matriz de confusão é um jeito fácil de observar as previsões do modelo por contabilizar todas as classificações. Em uma matriz, as linhas representam as classes reais e as colunas as classes previstas.

Por exemplo, suponha que nosso modelo funcione para descobrir qual é o algarismo escrito à mão. Assim, podemos escrever um dos algarismos de zero à nove. O modelo deve ler o algarismo e relevá-lo (predizer ou prever). Uma matriz de confusão desse modelo pode ser vista na Figura 4.

		Classe prevista									
		Zero	Um	Dois	Três	Quatro	Cinco	Seis	Sete	Oito	Nove
Classe real	Zero	90	2	3	4	5	6	1	1	1	4
	Um	1	80	3	4	5	6	5	1	1	0
	Dois	1	2	70	4	5	6	4	0	2	3
	Três	1	2	3	80	5	6	3	1	0	1
	Quatro	0	2	3	4	90	6	7	0	2	3
	Cinco	1	0	0	0	5	90	1	0	9	0
	Seis	1	2	1	4	0	6	80	8	0	0
	Sete	0	1	0	4	5	6	0	78	1	2
	Oito	1	2	3	4	0	0	1	1	95	3
	Nove	1	2	3	4	5	6	0	2	0	90

Figura 4 – Exemplo de matriz de confusão de classificador multinomial

Fonte: Elaborada pelo autor.

Aqui fizemos uma tabela, a apresentação usual é uma matriz. Observe que cada linha representa uma classe real e cada coluna uma classe prevista. Por exemplo, o elemento na linha nove e coluna oito é o 1. Isto significa que houve somente um erro em que o modelo achou que era o algarismo sete, mas na verdade, o manuscrito é do algarismo oito. Os elementos na diagonal principal são os acertos, os elementos fora dela contabilizam os erros.

A Figura 5 é uma matriz de confusão exclusiva para modelo de classificação binária, aquele em que o modelo deve decidir somente entre duas opções. Para isso vamos supor, independentemente do exemplo anterior, que temos interesse em saber se o manuscrito é

ímpar. Assim, construímos um modelo que retorna `positivo` para manuscritos ímpares ou `negativo` caso contrário. Neste sentido, considere a seguinte matriz de confusão:

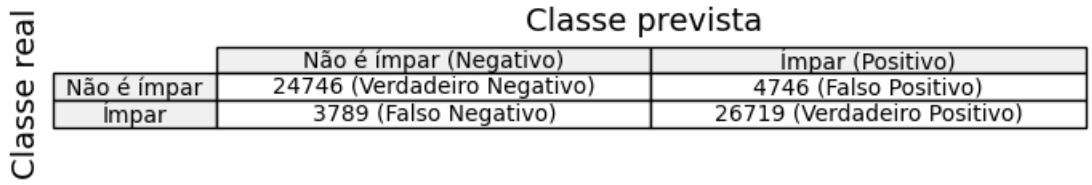


Figura 5 – Matriz de confusão do classificador binário

Fonte: Elaborada pelo autor.

Há algumas vantagens na exegese da matriz de confusão de classificação binária, a exemplo das informações entre parênteses, que foram adicionadas exclusivamente para melhor entendimento e serão exploradas doravante.

3.2.5.2 Precisão e revocação

Existem várias fórmulas para calcular métricas na matriz de confusão, das quais, neste artigo, iremos explorar a revocação, precisão e taxa de falsos positivos.

Revocação, ou taxa de verdadeiros positivos, ou ainda sensibilidade, é a proporção de instâncias positivas que são detectadas corretamente pelo modelo. É dada pela fórmula:

$$\text{revocação} = \frac{\text{Verdadeiro Positivo}}{\text{Verdadeiro Positivo} + \text{Falso Negativo}}.$$

Se avaliada isoladamente, a revocação é uma métrica inútil. Explico melhor, suponha que estamos interessado em julgar se o algarismo é ímpar ou não. Um modelo desinteligente que julgue tudo como ímpar apresentará uma revocação perfeita. Isto é, todas as instâncias ímpares serão detectadas corretamente. Destarte, é imperioso a interpretação conjunta com outra métrica.

Neste sentido, precisão é a acurácia das predições positivas. Em outras palavras, quando o modelo prediz como ímpar, qual é a taxa de acerto?

$$\text{precisão} = \frac{\text{Verdadeiro Positivo}}{\text{Verdadeiro Positivo} + \text{Falso Positivo}}.$$

Frequentemente a precisão e revocação são avaliadas em conjunto. Na prática, aumentar a precisão reduz a revocação e vice-versa. GÉRON (2019, p. 75) ilustra com maestria tais métricas:

> [...] em alguns contextos, você se preocupa basicamente com a precisão, e em outros, com a revocação. Por exemplo, se treinou um classificador com o objetivo de detectar vídeos seguros para crianças, provavelmente você prefere um classificador que rejeite muitos vídeos bons (baixa revocação) e mantenha apenas os seguros (alta precisão), em vez de um classificador que tenha uma revocação muito alta, mas permita que alguns vídeos ruins sejam exibidos em seu produto. [...] Em contrapartida, suponha que você treine um classificador para detectar ladrões de lojas em imagens de vigilância: provavelmente é bom que o classificador tenha somente 30% de precisão, desde que tenha 99% de revocação (claro, os guardas de segurança receberão alguns alertas falsos, mas quase todos os ladrões de lojas serão pegos).

3.2.5.3 Escolha da troca entre precisão e revocação

Alguns algoritmos fundamentais, entre eles o SVM, atribuem uma pontuação baseada em uma função de decisão para cada instância a fim de estabelecer a classe prevista.

Por exemplo, a Figura 6 foi obtida do banco de dados Modified National Institute of Standards and Technology database (MNIST), nela existem doze manuscritos, cada qual com um score atribuído. Simplificadamente, o modelo após treinado estabelece um score para cada manuscrito por meio de cálculos matemáticos. Preverá a classe ímpar (positivo) a todos que o score seja maior que o limiar, ou preverá a classe não ímpar (negativo) a todos que o score seja menor que o limiar. Em outras palavras, é o limiar de decisão que vai indicar a classe do exemplo desconhecido (previsão). Considere que o exemplo abaixo seja proveniente da matriz de confusão apresentada na Figura 5 e que o limiar de decisão esteja posicionado no zero.

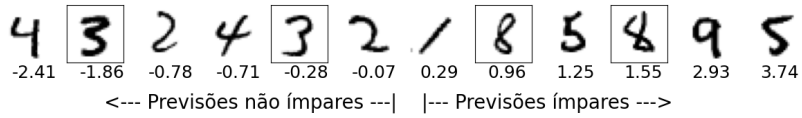


Figura 6 – Limiar de decisão posicionado no zero

Fonte: Elaborada pelo autor.

A escala do score é de 10^{-4}. Os manuscritos estão organizados em ordem crescente de score. O limiar de decisão está entre os manuscritos centrais (de classe 2 e 1). Os que estão destacados com um quadrado são erros de previsão. Os erros de previsão podem ser falsos negativos, se à esquerda do limiar, ou falsos positivos, se à direita do limiar. Organizado desse modo, a precisão é aproximadamente $0,8491$ e a revocação 0.8758, conforme se extrai da matriz de confusão apresentada na Figura 5.

Note que um leve aumento no limiar enseja o deslocamento da fronteira de decisão para a direita, provocando aumento de falsos negativos e diminuição de falsos positivos. Em outras palavras, ao aumentar a precisão, diminui-se a revocação (e vice-versa). Por exemplo, caso a fronteira de decisão fosse deslocada para a direita, posicionada entre os

manuscritos de classe oito e nove, com novo limiar de decisão em $2,24 \times 10^{-4}$, a nova precisão seria de aproximadamente $0,8492$ e a revocação $0,8757$.

Como ilustrado por GÉRON (2019), caso o modelo em análise objetivasse a detecção de conteúdos seguros para crianças (onde negativo é um conteúdo não seguro e positivo é seguro), seria salutar mover o limiar de decisão para um número maior, na esperança de aumentar a precisão por tender o valor dos falsos positivos para zero. Já que falso positivo se traduz em o modelo prever seguro um conteúdo que não é próprio para crianças (erro grave). Note que, em consequência da manobra, haverá um decréscimo na revocação, ensejando o aumento na previsão de inapropriado um conteúdo seguro para crianças. Neste caso, poder-se-ia adicionar um humano para supervisionar essa circunstância.

Por fim, cabe ressaltar que é comum a apresentação gráfica da precisão e revocação em função do limiar escolhido. Para o exemplo da Figura 5, tem-se o gráfico na seção 7.1.4.4.

3.2.5.4 Curva ROC

Outra forma semelhante de avaliar o funcionamento de um algoritmo classificador binário é construir a curva de característica de operação do receptor (ROC), denominada curva ROC, do inglês *Receiver Operating Characteristic*. É utilizado o conceito da taxa de falsos positivos, cuja definição é:

$$\text{taxa de falsos positivos } = \frac{\text{Falso Positivo}}{\text{Falso Positivo } + \text{ Verdadeiro Negativo}}.$$

A curva ROC é a revocação em função da taxa de falsos positivos. A seção 7.1.4.4, demonstra uma curva ROC no contexto da Figura 5. Uma maneira de mensurar a curva é calcular a área abaixo do gráfico. Neste sentido, um classificador perfeito tem área igual a um.

3.2.6 Expansão do conjunto de treinamento (*data augmentation*)

Frequentemente necessitamos ampliar o conjunto de treinamento. Ora porque uma classe contém significatavamente menos dados que outras, prejudicando a proporção entre elas. Ora porque necessitamos de maior variação representativa dos dados, em busca do comportamento real das variáveis estudadas. Obstante, em geral, não é simples obter novos dados, por vezes moroso e dispendioso. Neste sentido, expansão do conjunto de treinamento (do inglês *data augmentation*) consiste na técnica de aumentar artificialmente o conjunto de treinamento.

A Figura 7 mostra manuscritos do dígito 1 extraidos do banco de dados conhecido como MNIST, cada manuscrito contém também sua posição no banco. Caso necessite de maior variação representativa dos dados, a fim de maior semelhança com a escrita comum, poderiamos fazer cópias alteradas rotacionando os números e adicioná-las ao conjunto de treinamento. Essa seria uma forma de aumentar artificialmente os exemplo, neste caso específico, buscamos uma representação realista para o algarismo, por vezes escrito inclinado. Não obstante, existem outras técnicas para a expansão.

Figura 7 – Manuscritos do conjunto de dados MNIST

Fonte: Elaborada pelo autor.

3.2.7 Redução de dimensionalidade com PCA

Em alguns casos, é possível reduzir o número de características das instâncias a fim de tornar o treinamento eficiente e facilitar a busca pela solução. A imagem anterior mostra que os pixels nas bordas dos manuscritos são brancos, assim é razoável presumir que não perderá muita informação ao descartá-los. Além disso, os pixels vizinhos, em geral, são correlacionados. Destarte, substituí-los pela média ajudaria na eficiência, reduzindo o custo computacional.

Neste sentido, a Análise de Componentes Principais (PCA), um algoritmo de redução de dimensionalidade popular, localiza um hiperplano próximo aos dados para projetar os dados, reduzindo sua dimensionalidade. Para isso, escolhe um hiperplano que preserva a variância máxima, pois, em geral, há menos perda de informação que outras projeções.

A Figura 8 mostra três possíveis retas para projeção dos dados. Note que somente a projeção na linha sólida preserva a variância máxima.

3.3 Adversidades com os dados de treinamento

É fato que os dados são os alicerces do ML. Existe a crescente área de ciência de dados que estuda os dados a fim de extrair informações valiosas. Neste sentido, Pensamento Computacional de WING (2006) afirma que o ML vem sendo usado com dados de tamanho e dimensão inimagináveis até apenas alguns anos atrás. A eficácia irracional dos dados, tradução livre para *The Unreasonable Effectiveness of Data*, é o título dado por NORVIG et al. (2009) para um artigo que defende mais importância aos dados do que aos

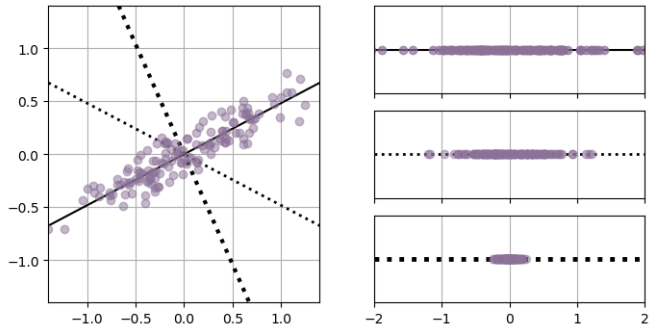


Figura 8 – Três possíveis retas para projeção dos dados

Fonte: Elaborada pelo autor.

algoritmos, no que diz respeito aos problemas complexos. Não obstante, BANKO; BRILL (2001), da Microsoft, foram ainda mais longe, publicaram um artigo em que mostraram desempenho quase idêntico de diferentes algoritmos quando alimentados com conjunto de dados suficientemente grande. Destarte, vamos apresentar algumas adversidades com os dados de treinamento que, se não observadas, não raro as vezes, culminam em algoritmos inócuos.

3.3.1 Quantidade insuficiente

Pelo exposto no parágrafo anterior, em geral, é necessário enorme quantidade de dados para que os algoritmos trabalhem como esperado. Normalmente, para problemas simples é requerido centenas de exemplos, para problemas complexos, tais como reconhecimento de imagens ou voz, milhões de exemplos. (GÉRON, 2019)

3.3.2 Não representativos

Os dados de treinamento devem ser representações dos novos casos. Observe que a Figura 7 mostra os manuscritos sem interferência de outros objetos – como linhas de folhas pautadas, sombreamento, feixe de luz, etc. Há apenas algarismo com boa nitidez. Neste sentido, seria desarrazoado conceber que o modelo preveria com acerto a imagem de um manuscrito com interferência de outro objeto, por exemplo, folha pautada. Em outras palavras, é importante buscar correspondência entre os dados de treinamento e os dados apresentados posteriormente. O que se ensina à máquina é o que se espera que aprenda.

Outro exemplo de dados não representativos foi propositalmente exibido na Figura 2, pois foram omitidos alguns países (entre eles o Brasil). A Figura 9 contempla todos os países. A linha pontilhada representa o modelo de regressão linear treinado anteriormente. A linha sólida é obtida por treinar todos os dados. Falamos que o dinheiro traz felicidade, agora, com o coeficiente angular da reta sólida medindo $1,838 \times 10^{-5}$, isso parece bem menos evidente. Na verdade, países ricos não são mais felizes do que países moderadamente

ricos. Por isso, é importante que os dados sejam representativos, sem ruído de amostragem ou viés de amostragem.

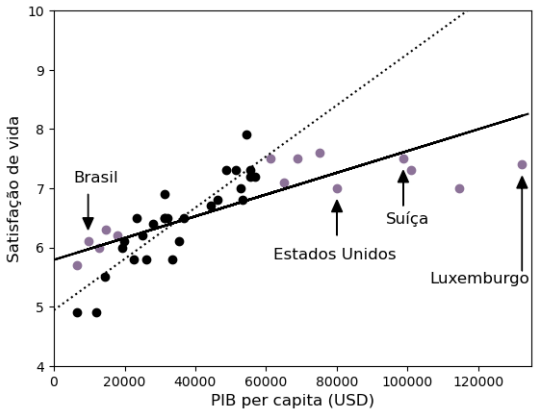


Figura 9 – Regressão linear nos dados completos

Fonte: Elaborada pelo autor.

3.3.3 Baixa qualidade

À medida que erros, outliers e ruídos permeiam o conjunto de treinamento, a máquina terá dificuldade para detectar padrões básicos. Ações como descartar alguns outliers e preencher com a média os valores faltantes de algumas instâncias podem atenuar os prejuízos dos dados de baixa qualidade.

3.3.4 Características irrelevantes

É imperioso que os dados tenham características relevantes suficientes. Selecionar e extrair as características mais importantes dos dados fazem parte do processo conhecido como engenharia de features (do inglês *feature engeneering*).

4 MÁQUINA DE SUPORTE VETORIAL

Também chamado de máquina de vetores de suporte (do inglês *Support Vector Machines*), ou simplemente SVM, é um algoritmo fundamental que serve de base para muitos problemas de aprendizado de máquina. Amplamente utilizado por sua versatilidade, pois aplica-se em tarefas de classificação e regressão. "É um dos modelos mais populares no aprendizado de máquina e qualquer pessoa interessada no assunto deve tê-lo em sua caixa de ferramentas" (GÉRON, 2019, p. 121).

Não obstante, "SVM é atualmente a abordagem pré-fabricada mais popular para aprendizagem supervisionada: se você não tiver nenhum conhecimento prévio especializado sobre o domínio, o SVM é um excelente primeiro método a testar." (RUSSELL; NORVIG, 2013, p. 860)

Um de seus fundadores asseverou que os algoritmos SVM "serão utilizados para o desenvolvimento de vários programas para reconhecimentos de padrões." (VAPNIK, 1982, p. 364, tradução nossa).

4.1 Classificação ou regressão

Quando se fala em aprendizado supervisionado, modelo de classificação é aquele em que os rótulos (também chamado de etiquetas, *labels* ou *outputs*) são finitos. Destarte, uma predição sempre será um dos rótulos apresentados no conjunto de treinamento. Ou seja, há um número restrito das possibilidades de predições. (BURKOV, 2019). São exemplos os problemas que envolvem decisões entre duas possibilidades: "Spam" ou "Não Spam", "Ligado" ou "Desligado". Essas são denominadas classificações binárias. Quando há três ou mais possibilidades são denominadas classificações multiclasses (ou multinomial).

Existem duas principais estratégias para transformar problemas de classificações multiclasses em binárias. A primeira consiste em avaliar uma determinada classe (todos os dados de mesmo rótulo) contra todas as outras, repetindo o procedimento para todas as classes. Esse método é chamado um contra todos, ou simplesmente OvR, do inglês *one-versus-the-rest*. A segunda consiste em avaliar cada par de classes e escolher a classe que vence mais confrontos, conhecido como um contra um, ou simplesmente OvO, do inglês *one-versus-one*.

Diferentemente, na regressão uma futura predição poderá ser completamente diferente de todos os rótulos apresentados. Por exemplo, estimar o valor de uma casa com base na quantidade de quartos, localização e área. Neste trabalho, iremos nos limitar a aprendizado supervisionado de classificação.

4.2 Funcionamento

De início, é salutar destacar que optamos pela notação de vetor representado com uma seta, e não como matriz de coluna única. Por exemplo, o vetor **a** é representado como $\vec{a}$. Além disso, o produto escalar dos vetores $\vec{a}$ e $\vec{b}$ é escrito como $\vec{a} \cdot \vec{b}$. Vale ressaltar que não usamos a multiplicação de matrizes porque resulta em uma matriz de célula única e não em um valor escalar como no produto escalar, inexoravelmente um abuso de notação.

Acerca do funcionamento em aprendizado supervisionado de classificação, o conjunto de dados é representado por vetores de características. Cada instância (também chamado de exemplo de entrada, *input*, observação ou caso) é representada por um vetor, em que cada componente descreve uma característica. Por exemplo, considere imagens de olhos de pessoas. Há várias maneiras de extrair as informações, destaco três: (1) registram-se as medidas de comprimento e largura de cada olho. Por exemplo, o vetor de características $\vec{x}_1 = (30, 9)$ representa um olho que mede 30mm de comprimento e 9mm de largura, atribuido à primeira instância; (2) registra-se a cor predominante da íris, usando a escala conhecida como RGB. Por exemplo, $\vec{v}_8 = (60, 60, 205)$ representa um olho predominantemente azul, atribuido à oitava instância; (3) registam-se as intensidades de cada pixel em escala de cinza. É mister que todas as imagens tenham a mesma quantidade de pixels, digamos 40x40. Neste caso, cada vetor de características terá 1600 componentes.

Das opções, qual, *a priori*, é adequada se o interesse for predizer sobre nascidos no ocidente/ oriente? Se o interesse for predizer sobre nascidos em países nórdicos ou não? Se o interesse for estudar sobre doenças oculares? Isso ilustra que há várias formas de interpretação. Além disso, os vetores de características têm quantidade finita de dimensão, onde cada componente é um número real e representa a mesma característica para todos os exemplos. Assim, para cada exemplo fornecido do conjunto de dados existe um vetor de características que o representa.

Após conhecido os vetores de características, o treinamento é útil para o algoritmo definir uma fronteira de decisão (hiperplano) que melhor separa os dados de classes diferentes. Na classificação binária, o espaço é dividido em dois semi-espaços cujos vetores de características de um determinado rótulo se concentrem em um semi-espaço, e do outro rótulo no outro semi-espaço. Ilustraremos com o caso mais simples, que será estudado no próximo capítulo. Não obstante, é mister uma definição.

Definição proposta por BINMORE (1980, p. 13): Um hiperplano H em $\mathbb{R}^k$, com o vetor $\vec{w}$ não nulo, é o lugar geométrico da forma:

$$H = \{\vec{x} \mid \vec{w} \cdot \vec{x} + b = 0\},$$

onde $\vec{w} \cdot \vec{x}$ representa o produto escalar e b um número real. Note que, em $\mathbb{R}^3$ um hiperplano é um plano, em $\mathbb{R}^2$ um hiperplano é uma reta, em $\mathbb{R}^1$ um hiperplano é um ponto. Destarte, segue um teorema.

Teorema: O vetor $\vec{w}$ é ortogonal ao hiperplano H.
Prova: Seja a_1 e a_2 dois pontos distintos no hiperplano. Logo, por definição:

$$\vec{w} \cdot \vec{a}_1 + b = 0, \text{ e}$$

$$\vec{w} \cdot \vec{a}_2 + b = 0.$$

Subtraindo as equações obtemos

$$\vec{w} \cdot (\vec{a}_1 - \vec{a}_2) = 0.$$

Assim, tem-se que o vetor $\vec{w}$ é ortogonal ao hiperplano porque é ortogonal a qualquer vetor arbitrário $(a_1 - a_2)$ no hiperplano. O vetor $\vec{w}$ mostra a direção que é normal ao hiperplano, portanto estabelece a orientação do hiperplano.

4.3 Classificação linear

A Figura 10 mostra algumas medidas de flores de duas espécies, trata-se de um popular conjunto de dados da flor íris (maiores informações na seção 5.1). Para cada ponto representado, existe um vetor de características $\vec{x}_i = (x^{(1)}, x^{(2)})$ associado, com $i = 1, \ldots, m$, onde m é a quantidade de instâncias, $x^{(1)}$ é o comprimento da pétala e $x^{(2)}$ é a largura da pétala. Além disso, é necessário atribuir um rótulo y_i para cada vetor de características, neste caso o valor numérico $+1$ para Versicolor e o valor -1 para Setosa. Portanto, $y_i \in \{\pm 1\}$. Os valores são escolhidos por conveniência matemática. Esses dados, que compõem o conjunto de treinamento, servem como base para treinar um SVM de classificação binária.

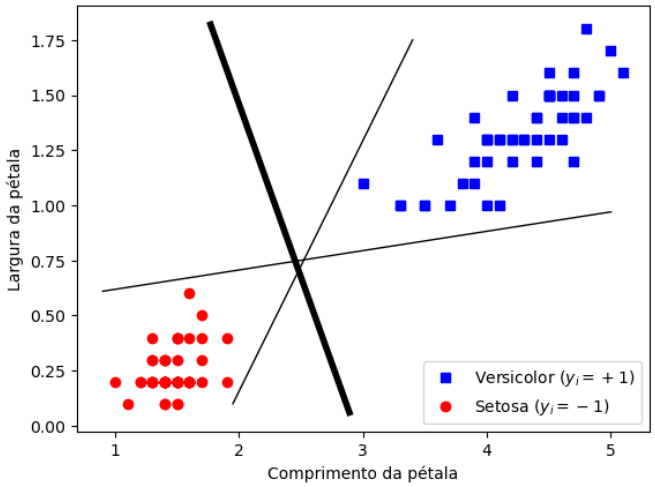


Figura 10 – Três hiperplanos, mas somente um tem maior margem de separação

Fonte: Elaborada pelo autor.

A Figura 10 mostra também três hiperplanos (neste caso retas), porém somente a reta em negrito tem maior margem de separação, isto é, a distância da reta aos pontos

mais próximos de cada classe é a maior possível. Note que as duas classes podem ser facilmente separadas por uma reta, ou seja, possui separabilidade linear. Um conjunto de dados é dito linearmente separável se cada semi-espaço tem somente pontos de uma classe. Observe que há três fronteiras de decisão, mas apenas uma é mais adequada. Uma fronteira de decisão serve para predizer sobre novas instâncias. Qual fronteira é "melhor"? O que caracteriza uma fronteira de decisão como melhor?

4.4 Escolha da fronteira de decisão no modelo de classificação

Esse capítulo também poderia ser entitulado como objetivo do treinamento, pois o objetivo é escolher os coeficiêntes que definem a melhor fronteira de decisão com margem larga. GÉRON (2019) ilustra o objetivo da classificação de margem larga como sendo semelhante à assentar uma margem em uma rua bastante larga (representadas na figura abaixo pelas linhas tracejadas paralelas) entre as classes.

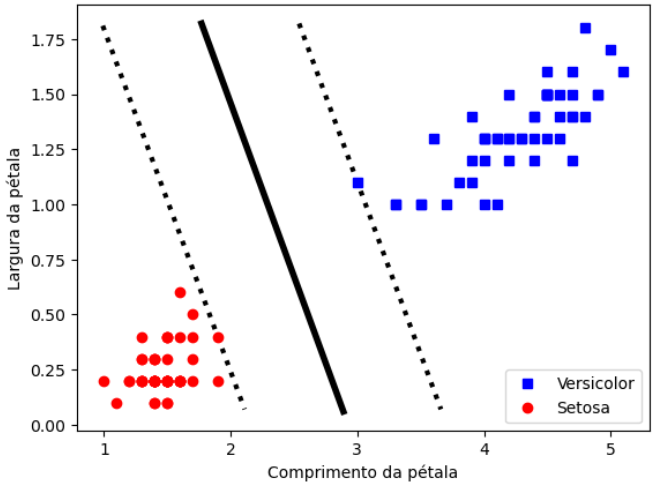


Figura 11 – Hiperplano de margem larga com vetores de suporte nas margens

Fonte: Elaborada pelo autor.

A Figura 11 ilustra que é preciso avaliar todo o conjunto de treinamento, contudo apenas as instâncias que cruzam as linhas tracejadas são determinantes para a fronteira de decisão. Essas instâncias são chamadas de vetores de suporte. Destarte, acrescentar outras instâncias "fora da margem" não altera a fronteira de decisão.

Supondo que o problema possui separabilidade linear, descrevemos a equação do hiperplano com dois parâmetros: um vetor normal $\vec{w}$ e um número real b. Assim, a equação é:

$$\vec{w} \cdot \vec{x} + b = 0, \tag{4.1}$$

com as variáveis representadas por $\vec{x} = (x^{(1)}, x^{(2)}, \ldots, x^{(k)})$, onde k é a dimensão do vetor

de características. A função de decisão correspondente a equação do hiperplano é:

$$y = \text{assinatura}(\vec{w} \cdot \vec{x} + b), \tag{4.2}$$

onde assinatura é um operador matemático que retorna $+1$ para entradas positivas e -1 para entradas negativas.

Segundo SCHÖLKOPF; SMOLA (2001), construir a função a partir dos dados empíricos é possível por dois motivos: Primeiro que entre todos os hiperplanos que separam as classes, existe um único "ótimo hiperplano", caracterizado por ter a maior margem de separação entre alguns pontos e o hiperplano. Isso é a solução para:

$$\underset{\vec{w},b}{\text{maximizar}} \min\{\|\vec{x} - \vec{x}_i\| \mid \vec{w} \cdot \vec{x} + b = 0, i = 1, 2, \ldots, m\},$$

como foi informado, m é a quantidade de instâncias. Segundo que a capacidade do hiperplano separar as classes diminui com o aumento da margem.

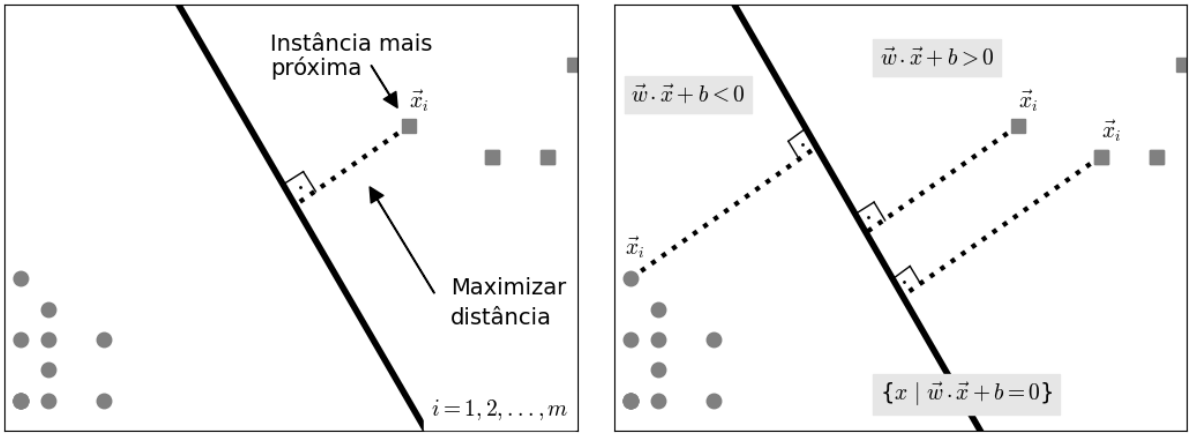


Figura 12 – Hiperplano que maximiza sua distância até as instâncias mais próximas

Fonte: Elaborada pelo autor.

À esquerda da Figura 12 lustra que o ótimo hiperplano é aquele que maximiza a sua distância até as instâncias mais próximas. Note que, à medida que movemos o hiperplano na intenção de aumentar a distância, o hiperplano acaba por se aproximar de outras instâncias. Como consequência disso, outras instâncias tornam-se mais próximas do hiperplano. À direita, o ótimo hiperplano é apresentado. Além disso, é revelado três regiões no plano em função da equação do hiperplano.

É mister notar que as informações estão com certa liberdade de ajuste, pois o desvio b permite a reescala. Em outras palavras, multiplicar ambos os lados da equação (4.1) por algum escalar resulta em hiperplanos equivalentes. Assim, vamos tornar único ou canônico o hiperplano, dimensionando para que os vetores de suporte satisfaçam as seguintes equações. (DSTA, 2023). O valor é escolhido por conveniência matemática.

$$\vec{w} \cdot \vec{x}^{+} + b = +1 \quad \text{e} \tag{4.3}$$

$$\vec{w} \cdot \vec{x}^{-} + b = -1, \tag{4.4}$$

em que $\vec{x}^{+}$ e $\vec{x}^{-}$ são os vetores de suporte das classes com rótulos $+1$ e -1, respectivamente. Essas equações descrevem as margens do hiperplano.

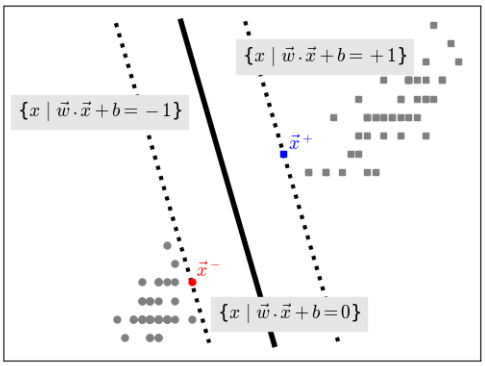


Figura 13 – Hiperplano com destaque dos vetores de suporte e margens

Fonte: Elaborada pelo autor.

A Figura 13 mostra o hiperplano com os vetores de suporte em destaque nas margens. O problema possui separabilidade linear, portanto as margens separam o espaço em duas regiões. Destarte, é mister que:

$$\begin{aligned} &\vec{w} \cdot \vec{x}_i + b \geq +1, \text{ se } y_i = +1 \text{ e} \\ &\vec{w} \cdot \vec{x}_i + b \leq -1, \text{ se } y_i = -1. \end{aligned} \tag{4.5}$$

Essas inequações podem ser resumidas por $y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1$, com $i = 1, \ldots, m$, onde m é a quantidade de instâncias.

Fazendo (4.3) – (4.4) e dividindo o resultado pelo tamanho de $\vec{w}$, obtemos a distância entre as margens:

$$\text{Largura da margem: } \frac{2}{\|\vec{w}\|}.$$

Maximizar a largura da margem é equivalente a encontrar $\vec{w}$ e b para minimizar $\|\vec{w}\|$, ou ainda

$$\min_{\vec{w},b} \frac{1}{2}\|\vec{w}\|^2, \tag{4.6}$$

$$\text{restrito a } y_i(w.x_i + b) - 1 \geq 0, \text{para todo } i = 1, \ldots, m. \tag{4.7}$$

Minimizar $\frac{1}{2}\|\vec{w}\|^2$ em vez de $\|\vec{w}\|$ é vantajoso, pois aquele possui derivada simples, enquato esse não é diverenciável em $w = 0$. Os algoritmos de otimização funcionam melhor em funções diferenciáveis (GÉRON, 2019).

4.5 Multiplicadores de Lagrange

As equações (4.6) e (4.7) juntas formam o chamado problema de otimização primário (SCHÖLKOPF; SMOLA, 2001), sendo (4.7) restrições de desigualdade. Problemas deste tipo podem ser remodelados, com os multiplicadores de Lagrange $\alpha_i \geq 0$ e a função de Lagrange (também conhecido como lagrangiano), para o chamado problema da dualidade (WILMOTT, 2019). Note que a função-objetivo (4.6) é convexa, as restrições de desigualdade são funções continuamente diferenciáveis e convexas, destarte a solução é a mesma para ambos os problemas (GÉRON, 2019). Assim, a função de Lagrange é:

$$L(\vec{w}, b) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{m} \alpha_i \Big(y_i(\vec{w} \cdot \vec{x}_i + b) - 1 \Big), \text{ com } \alpha_i \geq 0. \tag{4.8}$$

Para encontrar os pontos críticos, devemos calcular as derivadas parciais em relação a b e $\vec{w}$. Fazendo essas derivadas parciais iguais a zero temos respectivamente:

$$\sum_{i=1}^{m} \alpha_i y_i = 0, \text{ e} \tag{4.9}$$

$$\vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i. \tag{4.10}$$

Note que, pela equação acima, o vetor solução $\vec{w}$ é combinação linear de alguns vetores dos exemplos do conjunto de treinamento.

Substituindo (4.10) em (4.8) e considerando (4.9) obtemos o problema da dualidade. No que se refere a conveniência de resolver, prefere-se esse a resolver (4.6) e (4.7) (BURKOV, 2019; GÉRON, 2019). Devemos maximizar a função de Lagrange em relação a α_i (SCHÖLKOPF; SMOLA, 2001; WILMOTT, 2019).

$$\underset{\vec{\alpha} \in \mathbb{R}^m}{\text{maximizar}} \; L(\vec{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j, \text{ com } \vec{\alpha} = (\alpha_1, \ldots, \alpha_m) \tag{4.11}$$

$$\text{sujeito a } \alpha_i \geq 0, i = 1, \ldots, m, \tag{4.12}$$

$$\text{e } \sum_{i=1}^{m} \alpha_i y_i = 0. \tag{4.9}$$

Da condição de complementariedade de Karush-Kuhn-Tucker (KKT) da teoria de otimização segue que, para todo $i = 1, \ldots, m$, é necessário e suficiente que: (SCHÖLKOPF; SMOLA, 2001; VAPNIK, 1999)

$$\alpha_i[y_i(\vec{x}_i \cdot \vec{w} + b) - 1] = 0. \tag{4.13}$$

SCHÖLKOPF; SMOLA (2001) assevera que os $\vec{x}_i$ dos quais $\alpha_i > 0$ são chamados vetores de suporte. Essa terminologia é relacionada com o termo correspondente na teoria dos conjuntos convexos. De acordo com (4.13), eles estão localizados exatamente na margem. Todos os outros exemplos do conjunto de treinamento são irrelevantes: suas restrições (4.7) são satisfeitas automaticamente, e eles não aparecem na expansão (4.10). Não obstante, se o problema não possui separabilidade linear, então a formulação de Lagrange não é válida (WILMOTT, 2019).

Após encontrado os valores de α_i no problema dual, o modelo poderá classificar novos dados apresentados. Para predizer sobre um vetor de características $\vec{u}$, basta substituir a expansão (4.10) na função de decisão (4.2) e obter a classe correspondente:

$$y(\vec{u}) = \text{assinatura}\left(\sum_{i=1}^{m} \alpha_i y_i \vec{x}_i \cdot \vec{u} + b\right). \tag{4.14}$$

Por fim, cabe ressaltar que o conteúdo exposto é suficiente para resolver o primeiro exemplo teórico para o corpo discente em 5.1.

4.6 Margem suave

Até o presente momento, trabalhamos com situações em que não há elementos dentro das margens, ou seja, margem rígida. Existem duas desvantagens com a margem rígida. "Primeiro, ela só funciona se os dados forem linearmente separáveis. Segundo, ela é sensível a ruídos" (GÉRON, 2019, p. 122). Existe um modelo mais flexível, chamado de margem suave, que permite que os exemplos de treinamento cometam violações – caiam dentro da margem, ou até no lado errado da fronteira de decisão – mas lhe atribui a penalidade proporcional à distância necessária para movê-los de volta ao lado correto (RUSSELL; NORVIG, 2013).

Na Figura 14, que retoma o exemplo da íris, foi adicionado um ruído em destaque com um círculo. A estrela representa um novo elemento que não pertence ao conjunto de treinamento e deve receber uma classificação pelo modelo. À esquerda, um hiperplano semelhante ao hiperplano com separação de margem rígida. À direita, um hiperplano com separação de margem suave que considera o ruído como uma violação. O hiperplano da direita generaliza melhor, já que a predição para estrela é mais adequada a sua posição.

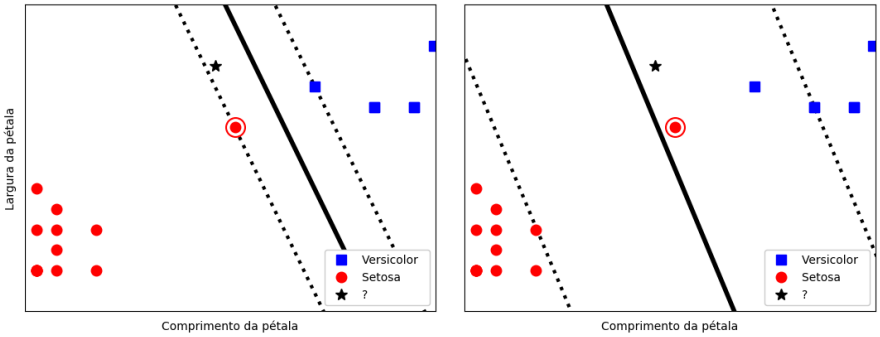


Figura 14 – Margem rígida (à esquerda) e margem que permite violações (à direita)

Fonte: Elaborada pelo autor.

O desafio é encontrar o equilíbrio entre maximizar a margem e limitar violações, assim introduziremos a função de perda chamada *hinge loss*: $\max(0, 1 - y_i(\vec{w} \cdot \vec{x}_i + b))$.

A função é zero quando (4.5) é satisfeita, isto é, o ponto x_i está na região da classe e não está entre as margens de separação. A função vale entre 0 e 1 se o ponto está dentro da margem, porém classificado corretamente. A função é maior que 1 se o ponto é classificado incorretamente e aparece no lado errado do hiperplano. Usamos essa função para penalizar instâncias em regiões erradas, proporcional à distância para movê-las de volta à margem correta.

Os objetivos conflitantes são: maximizar a margem e minimizar as violações, respeitanda a constante de regularização C entre elas. Para ambos os objetivos, devemos minimizar:

$$\frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{m} \max\Big(0, 1 - y_i(\vec{w} \cdot \vec{x}_i + b)\Big),$$

onde o hiperparâmetro $C > 0$ determina a prioridade em minimizar as violações, frente maximizar a margem. Um baixo valor para C prioriza a primeira parcela da expressão, ou seja, prioriza a margem larga e maior aceitação de violações. À medida que C é aumentado, a margem larga deixa de ter muita importância e os erros de classificação tormam-se escassos. Destarte, o modelo busca minimizar violações, sacrificando o tamanho da margem. O valor de C geralmente é escolhido experimentalmente (BURKOV, 2019). A Figura 14, à esquerda foi utilizado $C = 10$, à direita $C = 1$.

4.7 Truque do kernel

No caso em que o conjunto de treinamento não é linearmente separável por conta de um número relativamente pequeno de ruídos, a margem suave funciona bem. Contudo, caso o problema original não seja linearmente separável, mas

> pode parecer organizado em grupos, é possível usar o chamado truque do kernel (*kernel trick*). (WILMOTT, 2019, p. 111, tradução nossa).

A astúcia do truque é levar os dados para um espaço de dimensão superior com uma transformação que os tornem linearmente separáveis, com a enorme vantagem de baixíssimo acréscimo de cálculos matemáticos, que enseja eficiência em termos de processamento computacional. Destarte, (GÉRON, 2019, p. 124) considera o truque do kernel como uma "técnica matemática quase milagrosa".

Ilustraremos a lógica do truque a fim de compreender melhor seu funcionamento.

Na Figura 15, à esquerda um conjunto de treinamento no espaço de entrada em duas dimensões que parece organizado em grupos. Note que uma circunferência poderia ser a fronteira de decisão, contudo não é um hiperplano. Apesar de não ser linearmente separável em $\mathbb{R}^2$, o mesmo conjunto de treinamento após função de mapeamento ϕ torna-se linearmente separável em $\mathbb{R}^3$, à direita na figura. A equação (4.15) mostra a função de mapeamento. Neste caso, a circunferência torna-se um plano paralelo ao eixo $\sqrt{2}x_1x_2$, que é a fronteira de decisão no espaço de características. É fácil mostrar que aplicar a função de mapeamento à circunferência de raio 1, resulta em parcela do plano destacado. Para isso, use a equação parametrizada da circunferência de raio 1 $(\cos t, \sin t)$.

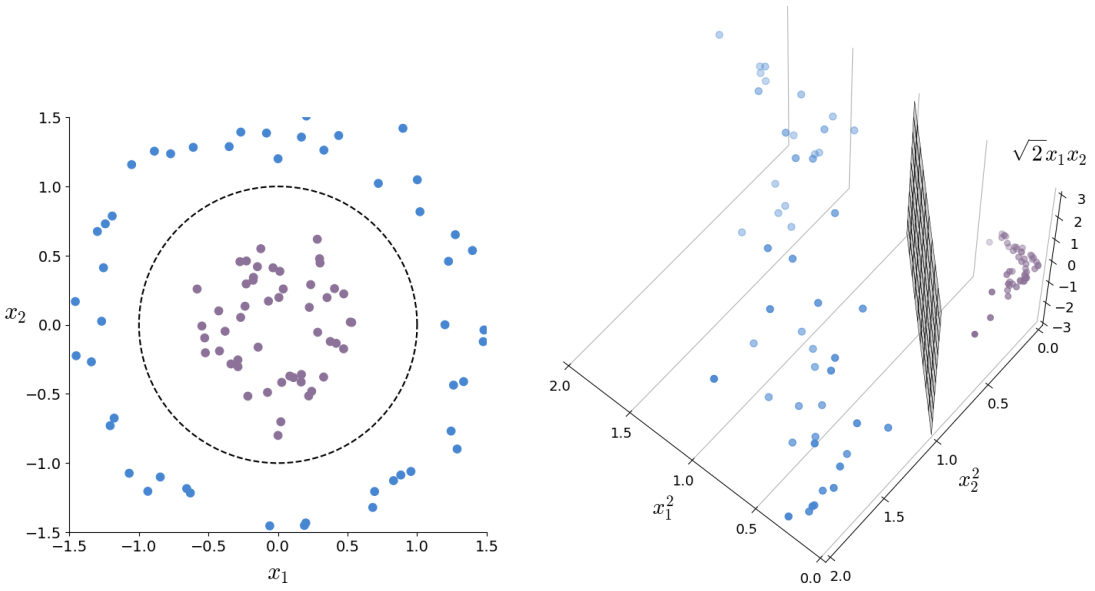


Figura 15 – Ilustração didática do truque do kernel

Fonte: Elaborada pelo autor.

Dado o espaço de entrada em $\mathbb{R}^2$, para fins didáticos, transformamos cada instância do conjunto de treinamento $\vec{x} = (x_1, x_2)$ em um ponto em $\mathbb{R}^3$, no espaço de características, via função de mapeamento polinomial de segundo grau:

$$\phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \tag{4.15}$$

Assim, no espaço de características, o objetivo se reduz a estimar um hiperplano a partir dos pontos de dados mapeados. Ou seja, recorrendo a (4.11) observa-se que para encontrar os αs (multiplicadores de Lagrange) devemos encontrar o produto escalar entre vetores no espaço de características. Calculando o produto escalar entre dois vetores, notamos a grande vantagem dessa técnica:

$$\phi(\vec{x}) \cdot \phi(\vec{x'}) = {x_1}^2{x'_1}^2 + {x_2}^2{x'_2}^2 + 2x_1x'_1x_2x'_2 = (x_1x'_1 + x_2x'_2)^2 = (\vec{x} \cdot \vec{x'})^2 \tag{4.16}$$

O ponto central é que em busca da região de separação linear, descobrimos que não necessitamos conhecer os detalhes da função de mapeamento ϕ. Pelo exposto, o produto escalar no espaço de características depende exclusivamente do produto escalar no espaço de entrada (WILMOTT, 2019). Em termos de processamento computacional, não precisamos aplicar ϕ as instâncias de treinamento! Sequer conhecê-la! Desde que o produto escalar das transformações dependa apenas do produto escalar das entradas e que conhecemos esse resultado (GÉRON, 2019).

$K(x, x') = (\vec{x} \cdot \vec{x'})^2$ é chamada de função kernel, e geralmente é escrita como $K(x, x')$. A função kernel pode ser aplicada a pares de dados de entrada para avaliar produtos escalares em algum espaço característico correspondente. Assim, podemos encontrar separadores lineares em espaços característicos de dimensão superior $\phi(x)$ simplesmente substituindo $\vec{x}_i \cdot \vec{x}_j$ em (4.11) por uma função kernel $K(x, x')$. Destarte, podemos aprender no espaço de dimensão superior, mas calculamos apenas as funções kernel em vez de uma lista completa de características para cada ponto de dados (RUSSELL; NORVIG, 2013).

Abaixo, manipulamos a equação (4.11), originalmente no espaço de entrada, para ser compreendida no espaço de características, via ϕ. Observe o que ocorre com kernel polinomial de segundo grau:

$$\underset{\alpha}{\text{maximizar}}\, L = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \phi(\vec{x}_i) \cdot \phi(\vec{x}_j).$$

Usando o resultado obtido em (4.16), segue que:

$$\underset{\alpha}{\text{maximizar}}\, L = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j)^2, \tag{4.17}$$

Lembre-se de que $\alpha_i \neq 0$ apenas para vetores de suporte. Assim, é possível encontrar os valores dos αs, observadas as equações (4.12) e (4.9). Não obstante, use os

multiplicadores de Lagrange em (4.10) e vai notar que não é possível encontrar $\vec{w}$, o vetor que orienta o hiperplano, sem usar a transformação ϕ (e a nossa ideia é não usar a transformação!). A menos que sua curiosidade em encontrar $\vec{w}$ seja insuperável, podemos prosseguir ignorando o fato de que queremos conhecê-lo. Destacando a parcela interna do operador matemático assinatura em (4.14) e plugando (4.10) nela, compreendida no espaço de características, obtemos:

$$\sum_{i=1}^{m} \alpha_i y_i \phi(\vec{x}_i) \cdot \phi(\vec{u}) + b$$

Lembre-se de que $\vec{u}$ é uma instância qualquer e aquela expressão é sua assinatura. Usando o resultado obtido em (4.16), segue que:

$$\sum_{i=1}^{m} \alpha_i y_i (\vec{x}_i \cdot \vec{u})^2 + b \tag{4.18}$$

É notável (e muito surpreendente!) o fato que mesmo sem o valor de $\vec{w}$ seja possível efetuar predições de novas instâncias, pois o essencial é o produto escalar das transformações. GÉRON (2019) assevera que "$\vec{w}$ deve ter o mesmo número de dimensões que $\phi(x)$, que pode ser gigantesco ou até infinito, de modo que você não consegue calculá-lo." Já WILMOTT (2019) assevera que alguns kernels aplicados em espaço de entrada de dimensão finita levam ao espaço de características de dimensão infinita, a exemplo do kernel Gaussiano.

Usa-se a mesma função kernel para calcular o viés b. No contexto trazido, pode ser calculado substituindo $\vec{u}$, na expressão acima, por um dos vetores de suporte, já que é conhecido. Após o cálculo, iguale a expressão ao valor da classe correspondente.

"Em aprendizado de máquina, um kernel é uma função capaz de calcular o produto escalar $\phi(\vec{x}) \cdot \phi(\vec{x}')$ com base apenas nos vetores originais $\vec{x}$ e $\vec{x}'$, sem ter que calcular (ou mesmo conhecer) a transformação ϕ." (GÉRON, 2019, p. 134). Abaixo, elencamos alguns kernels mais usados:

$$\begin{aligned}
\text{Linear:} \quad & K(\vec{x}, \vec{x}') = \vec{x} \cdot \vec{x}' \\
\text{Polinomial:} \quad & K(\vec{x}, \vec{x}') = (\gamma \vec{x} \cdot \vec{x}' + r)^d \\
\text{RBF gaussiano:} \quad & K(\vec{x}, \vec{x}') = \exp\Big(-\gamma \|\vec{x} - \vec{x}'\|^2 \Big)
\end{aligned}$$

Segundo RUSSELL; NORVIG (2013), existe o respeitável Teorema de Mercer o qual informa que qualquer função kernel, que respeita algumas condições, corresponde a algum espaço característico. Isto é, pode usar a função kernel porque sabe que ϕ existe, mesmo que não saiba o que é. (GÉRON, 2019).

4.8 Resumo esquemático para abordagem em sala de aula

4.8.1 Abordagem lúdica com geometria

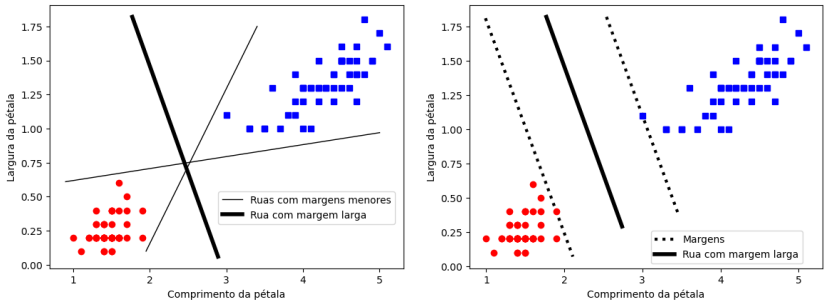


Figura 16 – Resumo esquemático usando abordagem lúdica

Fonte: Elaborada pelo autor.

Vocabulário recomendado: ponto, rua e margem.

Reflexão: Entre todas as "ruas"que separam os pontos de cores diferentes, qual é a que deixa a margem mais larga?

4.8.2 Abordagem com rigor intermediário

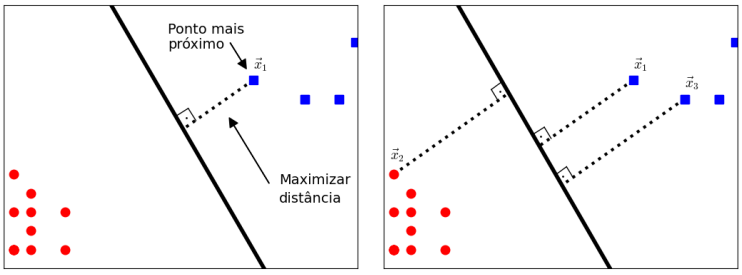


Figura 17 – Resumo esquemático usando abordagem com rigor intermediário

Fonte: Elaborada pelo autor.

Vocabulário recomendado: ponto, reta e distância.

Reflexão: Entre todas as retas que separam os pontos de cores diferentes, busque a reta cuja sua distância até o ponto mais próximo seja a maior possível.

5 EXERCÍCIOS TEÓRICOS PARA O CORPO DISCENTE

De início, é importante ressaltar que foi realizado um processo de transposição didática para a construção desta seção. Esta etapa serve como atividade auxiliar para melhor entendimento do Aprendizado de Máquina. Vale dizer que não se pretende criar extensos exercícios teóricos. O propósito é meramente viabilizar a teoria do SVM no Ensino Médio.

5.1 Primeiro exercício teórico (Íris)

5.1.1 Considerações ao corpo docente

O exemplo a seguir requer prévio conhecimento em operações básicas, multiplicação de matrizes ou produto escalar. Para além disso, o discente deve ter habilidade de encontrar o máximo de função quadrática. Caso julgue que os discentes se sentirão desconfortáveis em realizar manipulações algébricas envolvendo a notação Sigma, pode suprimir a notação e apresentar na forma expandida. Desta maneira, a notação de somatório seria remodelada.

A atividade permeia assuntos como: função quadrática, distância de ponto e reta, equação da reta e vetor (ou matriz).

É mister notar que o produto escalar pode ser reformulado como multiplicação de matrizes. Por exemplo, considere os vetores $\vec{a}(x_a, y_a)$ e $\vec{b}_2(x_b, y_b)$ representados, respectivamente, pelas matrizes $A = \begin{bmatrix} x_a \\ y_a \end{bmatrix}$ e $B = \begin{bmatrix} x_b \\ y_b \end{bmatrix}$. Considere ainda que A^T é a matriz transposta de A. Logo, o produto escalar é o único elemento da matriz resultado, isto é:

$$A^T B = \begin{bmatrix} x_a & y_a \end{bmatrix} \cdot \begin{bmatrix} x_b \\ y_b \end{bmatrix} = \begin{bmatrix} \underbrace{x_a.x_b + y_a.y_b}_{\text{produto escalar de } \vec{a} \text{ e } \vec{b}} \end{bmatrix}$$

5.1.2 Duração

sugere-se, no mínimo, 45 minutos.

5.1.3 Objetivo

Apresentar aos alunos uma forma de encontrar a equação da reta que representa a fronteira de decisão do SVM.

5.1.4 Exercício e sugestão de resolução

Os dados da flor íris anteriormente citados podem ser usados para calcular os multiplicadores de Lagrange na prática. Eles estão disponíveis e podem ser obtidos em `https://www.kaggle.com/uciml/iris`, ou importados no python usando o comando `from sklearn import datasets`. Existem três tipos de íris: Setosa, Versicolor e Virginica. Os dados contêm quatro informações: medidas de comprimento e largura, de sépalas e pétalas. Destarte, o leitor pode escolher duas informações de interesse de duas espécies e encontrar a equação do hiperplano, que neste caso é uma reta, desde que seja possível a separação linear.

Anteriormente, plotamos a largura em função do comprimento da pétala das espécies Versicolor e Setosa. Pode-se extrair todas as informações na fonte, contudo a figura abaixo contém apenas algumas observações. Como visamos à prática matemática pelo discente, considere somente os vetores $\vec{x}_1$ e $\vec{x}_2$, a fim de tornar a tarefa exequível no Ensino Médio.

Atividade I: A Figura 18 releva vetores de características. Eles devem ser usados em (4.11) com a restrição (4.9) para maximizar a função de Lagrange, com a finalidade de encontrar os valores de α_i, com $i = 1, \ldots, m$. Considerando $m = 2$, **convidamos o leitor a praticar**. A solução é apresentada adiante.

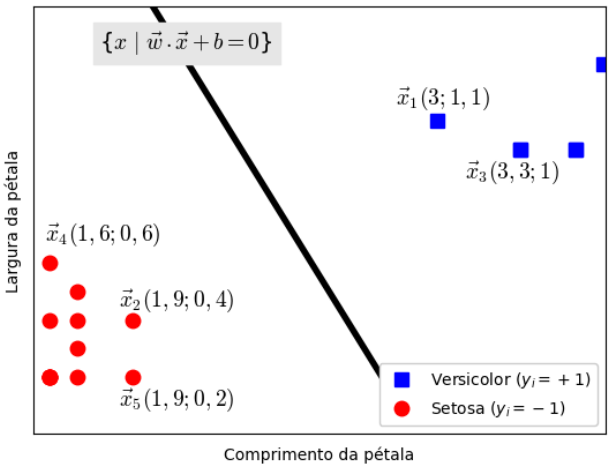


Figura 18 – Exercício da flor íris para discentes

Fonte: Elaborada pelo autor.

Como informado, m poderia ser um dos seguintes inteiros $\{2, 3, 4, 5\}$. Para fins de trabalho sem o uso de técnicas de cálculo, use $m = 2$, pois ainda que se queira solucionar

o problema com outro valor para m, a limitação (4.12) seria infringida. [Apenas para apetecer ao docente que queira constatar tal fato: usar $m = 3$ e desse modo, restaria na função de Lagrange duas variáveis; em seguida, encontrar o máximo – calcular a derivada parcial de primeira ordem de cada variável, tornando inviável no Ensino Médio – tudo isso para constatar que α_3 contraria (4.12) e obriga a retificação de m]. Isso acontece pois os vetores $\vec{x}_1$ e $\vec{x}_2$ definem a equação do hiperplano, chamados de vetores de suporte. Todos os outros exemplos - *ex vi* (4.13) - não aparecem na função de Lagrange, suas limitações (4.7) são satisfeitas automaticamente.

Resolução da atividade I: Os vetores são $\vec{x}_1(3; 1, 1)$ da classe $y_1 = 1$ e $\vec{x}_2(1, 9; 0, 4)$ da classe $y_2 = -1$. Para $m = 2$, as equações (4.9) e (4.11) ficam, respectivamente:

$$\sum_{i=1}^{2} \alpha_i y_i = \alpha_1 y_1 + \alpha_2 y_2 = 0, \text{ logo } \alpha_1 = \alpha_2, \quad \text{e} \tag{5.1}$$

$$\begin{aligned}
\underset{\vec{\alpha}\in\mathbb{R}^2}{\text{maximizar}} L(\vec{\alpha}) &= \sum_{i=1}^{2} \alpha_i - \frac{1}{2} \sum_{i=1}^{2} \sum_{j=1}^{2} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j \\
&= \alpha_1 + \alpha_2 - \frac{1}{2}\Big(\alpha_1^2 y_1^2 \vec{x}_1 \cdot \vec{x}_1 + 2\alpha_1\alpha_2 y_1 y_2 \vec{x}_1 \cdot \vec{x}_2 + \alpha_2^2 y_2^2 \vec{x}_2 \cdot \vec{x}_2\Big) \\
&\overset{(5.1)}{=} 2\alpha_1 - \frac{\alpha_1^2}{2}\Big(y_1^2 \vec{x}_1 \cdot \vec{x}_1 + 2 y_1 y_2 \vec{x}_1 \cdot \vec{x}_2 + y_2^2 \vec{x}_2 \cdot \vec{x}_2\Big) \\
&= -0,85\alpha_1^2 + 2\alpha_1.
\end{aligned}$$

Encontrado o valor que maximiza a função acima, encontra-se os multiplicadores de Lagrange. Neste caso, $\vec{\alpha}_1 = \vec{\alpha}_2 = \frac{20}{17}$. Para todo i diferente de um ou dois, $\alpha_i = 0$ (caso use $m \neq 2$).

Atividade II: Use os multiplicadores de Lagrange em (4.10) e o resultado em (4.3), para descobrir em (4.1) a equação da reta que representa a fronteira de decisão. Novamente, **convidamos o leitor a praticar**.

Resolução da atividade II: Com os resultados anteriores e $m = 2$, a equação (4.10) torna-se:

$$\vec{w} = \sum_{i=1}^{2} \alpha_i y_i \vec{x}_i = \alpha_1 y_1 \vec{x}_1 + \alpha_2 y_2 \vec{x}_2 \overset{(5.1)}{=} \alpha_1(y_1\vec{x}_1 + y_2\vec{x}_2) = \Big(1,294; 0,823\Big).$$

Aplicando esse resultado em (4.3) e considerando que o vetor de suporte da classe positiva é $\vec{x}^+ = \vec{x}_1$, obtemos:

$$\vec{w} \cdot \vec{x}_1 + b = \Big(1,294; 0,823\Big) \cdot \Big(3; 1, 1\Big) + b = 1, \text{ logo } b = -3,788.$$

Neste sentido, a equação do hiperplano que melhor separa os dados é a reta:

$$1,294 \times \text{largura da pétala } + 0,823 \times \text{comprimento da pétala } - 3,788 = 0.$$

5.1.5 Resumo esquemático para abordagem em sala de aula

5.1.5.1 Atividade I

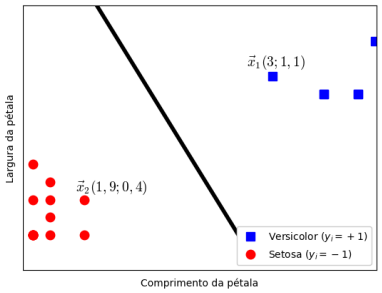


Figura 19 – Resumo esquemático primeira atividade I

Fonte: Elaborada pelo autor.

Informações fornecidas: $\vec{x}_1 = (3;1,1)$, $\vec{x}_2 = (1,9;0,4)$, $y_1 = 1$ e $y_2 = -1$.
Roteiro: Sabendo que $\alpha_1 y_1 + \alpha_2 y_2 = 0$, manipule essa informação para tornar a função L uma função quadrática *(feito a seguir)*. Após, o objetivo é encontrar os valores de α_1 e α_2 que tornam a função L máxima. Para isso, encontre o valor que maximiza a função (abscissa do vértice da função).

$$\begin{aligned} L &= \alpha_1 + \alpha_2 - \frac{1}{2}\left(\alpha_1^2 y_1^2 \vec{x}_1 \cdot \vec{x}_1 + 2\alpha_1\alpha_2 y_1 y_2 \vec{x}_1 \cdot \vec{x}_2 + \alpha_2^2 y_2^2 \vec{x}_2 \cdot \vec{x}_2\right) \\ &= - - 0,85\alpha_1^2 + 2\alpha_1. \end{aligned}$$

5.1.5.2 Atividade II

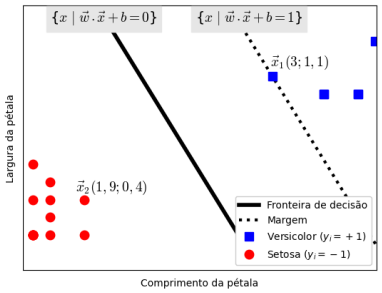


Figura 20 – Resumo esquemático primeira atividade II

Fonte: Elaborada pelo autor.

Informações fornecidas: $\vec{x}_1 = (3;1,1)$, $\vec{x}_2 = (1,9;0,4)$, $y_1 = 1$, $y_2 = -1$ e $\alpha_1 = \alpha_2 = \frac{20}{17}$.
Roteiro: Sabendo que $\vec{w} = \alpha_1 y_1 \vec{x}_1 + \alpha_2 y_2 \vec{x}_2$, encontre o valor de $\vec{w}$. Também sabemos que o $\vec{x}_1$ é vetor de suporte, portanto $\vec{w} \cdot \vec{x}_1 + b = +1$. Com isso, encontre o valor de b. Por fim, use os valores encontrados para descobrir a equação da reta de decisão dada por $\vec{w} \cdot \vec{x} + b = 0$. Vale ressaltar que é possível usar a equação $\vec{w} \cdot \vec{x}_2 + b = -1$, para encontrar o valor de b

5.2 Segundo exercício teórico (Truque do kernel)

5.2.1 Considerações ao corpo docente

O exemplo a seguir requer prévio conhecimento em operações básicas, multiplicação de matrizes ou produto escalar. Para além disso, o discente deve ter habilidade de encontrar o máximo de função quadrática. Caso julgue que os discentes se sentirão desconfortáveis em realizar manipulações algébricas envolvendo a notação Sigma, pode suprimir a notação e apresentar na forma expandida. Desta maneira, a notação de somatório seria remodelada. Foi mostrado em 5.1.1 que o produto escalar pode ser reformulado em multiplicação de matrizes.

A atividade permeia equação algébrica e vetor (ou matriz).

No exemplo a seguir, foi suprimida a etapa de encontrar a função e os multiplicadores de Lagrange, pois exigiria habilidade em encontrar o máximo de função com duas variáveis reais. Não cabe ao discente de ensino médio essa habilidade. Assim, a etapa suprimida encontra-se adiante para usufruto do docente.

Observe a Figura 21 (na lauda seguinte) para extração das informações. Consideramos os vetores de suporte, em destaque na figura, $\vec{x}_1 = (0; 1, 2)$, $\vec{x}_2 = (1, 2; 0)$ e $\vec{x}_3 = (0; -0.8)$ com rótulos, $y_1 = y_2 = +1$ e $y_3 = -1$. Há apenas três vetores de suporte, logo usaremos $m = 3$ em (4.9), ou seja:

$$\sum_{i=1}^{3} \alpha_i y_i = 0, \text{ assim } \alpha_1 + \alpha_2 = \alpha_3.$$

Aplicando esse resultado em (4.17), que já contempla o kernel polinomial de segundo grau, respeitado (4.12), segue que:

$$\begin{aligned} \underset{\alpha}{\text{maximizar}}\ L &= \sum_{i=1}^{3} \alpha_i - \frac{1}{2} \sum_{i=1}^{3} \sum_{j=1}^{3} \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j)^2 \\ &= -0,32\alpha_1^2 - 1,2416\alpha_2^2 + 0,5119\alpha_1\alpha_2 + 2\alpha_1 + 2\alpha_2 \end{aligned}$$

Para encontrar o máximo da função, devemos encontrar as derivadas parciais. Zerando cada derivada parcial, obtêm-se os coeficientes que maximizam a função. Destarte, $\alpha_1 = 4,5138$, $\alpha_2 = 1,7361$ e $\alpha_3 = 6,2499$. Como sabemos, para todos os vetores que não são vetores de suporte $\alpha_i = 0$ (4.13).

5.2.2 Duração

sugere-se 45 minutos

5.2.3 Objetivo

Apresentar aos alunos uma forma de encontrar a função de decisão do SVM com o truque do kernel.

5.2.4 Exercício e sugestão de resolução

A Figura 21, que reproduz a apresentada anteriormente, destaca os vetores de suporte com um círculo. A malha quadriculada no espaço de entrada serve de estimativa para os valores dos pontos. Como o propósito dessa atividade é ilustrar o truque do kernel, julgamos que inexiste prejuízo na estimativa. Assim, sempre que necessário, extraia da figura as informações posicionais necessárias.

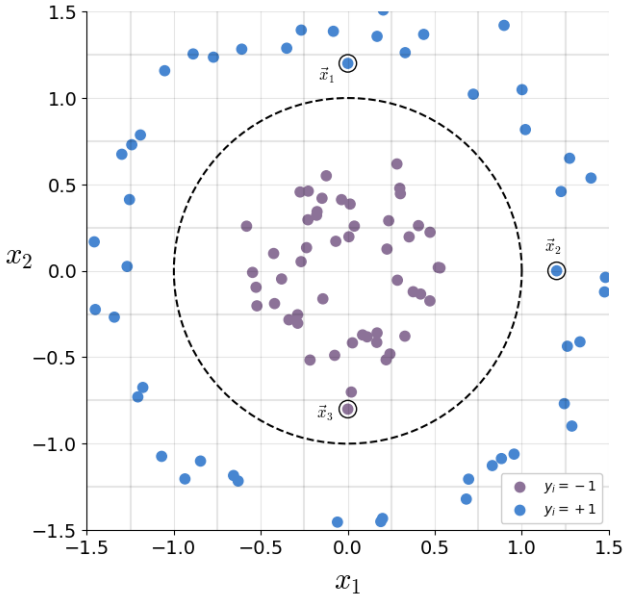


Figura 21 – Exercício com o truque do kernel para discentes

Fonte: Elaborada pelo autor.

Fornecido os multiplicadores de Lagrange, o objetivo é encontrar o valor do viés b e a expressão dentro da função de decisão (4.14) para uma instância arbitrária $\vec{u}$, usando kernel polinomial de segundo grau. De início, é mister estimar os vetores de suporte na figura. Seja $\vec{x}_1$ e $\vec{x}_2$ os vetores de suporte da classe externa ao círculo, com rótulo $+1$, e $\vec{x}_3$ o vetor de suporte da classe interna ao círculo, com rótulo -1. É estimado que $\vec{x}_1 = (0; 1, 2)$, $\vec{x}_2 = (1, 2; 0)$ e $\vec{x}_3 = (0; -0.8)$ com rótulos, respectivamente, $y_1 = y_2 = +1$ e $y_3 = -1$.

Os valores dos multiplicadores de Lagrange, representados por α, são $\alpha_1 = 4,51388$, $\alpha_2 = 1,73611$ e $\alpha_3 = 6,24999$. Com esses valores, buscamos a expressão na função de decisão para uma instância arbitrária $\vec{u}$, usando kernel polinomial de segundo grau (como visto anteriormente).

Atividade I: Fornecido os multiplicadores de Lagrange, use os vetores de suporte e suas respectivas classes para encontrar o valor do viés b. Em outras palavras, faça $\vec{u} = \vec{x}_3$ na expressão (4.18), com $m = 3$. Note que nessa expressão já foi aplicado o kernel. **Convidamos o leitor a praticar.** A estratégia consiste em aplicar o próprio vetor de suporte como se fosse uma instância arbitrária.

Resolução da atividade I: A expressão (4.18) para $m = 3$ e $\vec{u} = \vec{x}_3$ é:

$$\begin{aligned}\sum_{i=1}^{3} \alpha_i y_i (\vec{x}_i \cdot \vec{x}_3)^2 + b &= \alpha_1 y_1 (\vec{x}_1 \cdot \vec{x}_3)^2 + \alpha_2 y_2 (\vec{x}_2 \cdot \vec{x}_3)^2 + \alpha_3 y_3 (\vec{x}_3 \cdot \vec{x}_3)^2 + b \\ &= 1,6 + b\end{aligned}$$

Foi discutido que a expressão acima assume o valor correspondente à classe do vetor de suporte inserido. Assim, a expressão deve ser -1, ou seja $b = -2,6$. Cabe ressaltar que para uma melhor estabilidade do viés, em um cenário real, recomenda-se repetir o procedimento para todos os vetores de suporte e calcular a média dos vieses.

Atividade II: Substitua os valores conhecidos na expressão (4.18) para encontrar a função de decisão. **Convidamos o leitor a praticar**.

Resolução da atividade II: A expressão com a instância arbitrária $\vec{u}(u_x, u_y)$ é:

$$\begin{aligned}\sum_{i=1}^{3} \alpha_i y_i (\vec{x}_i \cdot \vec{u})^2 + b &= \alpha_1 y_1 (\vec{x}_1 \cdot \vec{u})^2 + \alpha_2 y_2 (\vec{x}_2 \cdot \vec{u})^2 + \alpha_3 y_3 (\vec{x}_3 \cdot \vec{u})^2 + b \\ &= 2,5u_x^2 + 2,5u_y^2 - 2,6\end{aligned}$$

Finalmente, a expressão acima usa o kernel polinomial de segundo grau para predizer a classe de novas instâncias, já que é possível prever a classe de uma instância $\vec{u}$ qualquer. Como visto em (4.2), se a expressão for positiva, $\vec{u}$ deve ser de classe $+1$, se a expressão for negativa, $\vec{u}$ deve ser de classe -1. Vale dizer que esse foi um cenário idealizado com intuito didático.

5.2.5 Resumo esquemático para abordagem em sala de aula

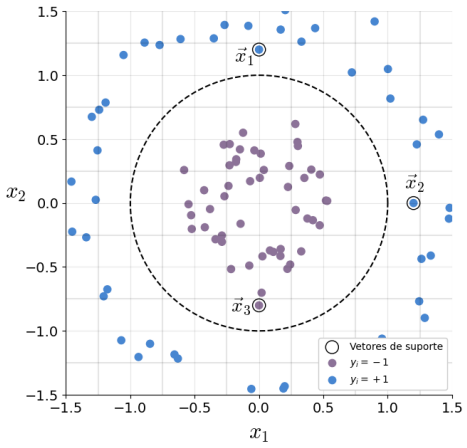


Figura 22 – Resumo esquemático segunda atividade I e II

Fonte: Elaborada pelo autor.

5.2.5.1 Atividade I

Informações fornecidas: $\vec{x}_1 = (0; 1, 2)$, $\vec{x}_2 = (1, 2; 0)$, $\vec{x}_3 = (0; -0, 8)$, $y_1 = y_2 = 1$, $y_3 = -1$, $\alpha_1 = 4, 51$, $\alpha_2 = 1, 73$ e $\alpha_3 = 6, 24$.
Roteiro: Sabe-se que a expressão da função de decisão assume o valor da classe do vetor inserido. Assim, substitua $\vec{u} = \vec{x}_3$ na expressão abaixo. Após, iguale o resultado ao valor de y_3 para descobrir o valor de b.

Expressão da função de decisão: $\alpha_1 y_1 (\vec{x}_1 \cdot \vec{u})^2 + \alpha_2 y_2 (\vec{x}_2 \cdot \vec{u})^2 + \alpha_3 y_3 (\vec{x}_3 \cdot \vec{u})^2 + b$

5.2.5.2 Atividade II

Informações fornecidas: $\vec{x}_1 = (0; 1, 2)$, $\vec{x}_2 = (1, 2; 0)$, $\vec{x}_3 = (0; -0, 8)$, $y_1 = y_2 = 1$, $y_3 = -1$, $\alpha_1 = 4, 51$, $\alpha_2 = 1, 73$, $\alpha_3 = 6, 24$ e $b = -2, 6$.
Roteiro: Use os dados para obter a expressão da função de decisão para uma instância arbitrária $\vec{u} = (u_x, u_y)$

Expressão da função de decisão: $\alpha_1 y_1 (\vec{x}_1 \cdot \vec{u})^2 + \alpha_2 y_2 (\vec{x}_2 \cdot \vec{u})^2 + \alpha_3 y_3 (\vec{x}_3 \cdot \vec{u})^2 + b$

6 BREVE TUTORIAL SOBRE O USO DE PYTHON PARA A IMPLEMENTAÇÃO

De início, é importante ressaltar que o propósito deste tutorial é suplementar as etapas de implementação, no que se refere a linguagem de programação Python. Serve de material auxiliar para melhor entendimento dos comandos utilizados e suporte para alterações no código, caso julgue necessário. Vale dizer, não se pretende criar um tutorial abrangente da linguagem de programação, existem boas dissertações com essa finalidade. Aqui, o objetivo é meramente sustentar a implementação proposta.

6.1 Iniciando no Python: instalação ou uso online

Não é simples colocar as instruções de instalação mais recentes nessa dissertação. Se não tem nenhuma experiência com Python, recomendamos que use o Google Colab (*https://colab.research.google.com/*), uma forma simples de programar online com o servidor da Google. A ferramenta pode ser utilizada em dispositivo móvel, todavia existem algumas limitações. Além disso, o tempo de processamento é limitado pelos servidores da Google e exige conexão permanente com a internet. Caso tenha alguma experiência com linguagem de programação ou paciência suficiente para poucos minutos de pesquisas, além de uma máquina com boas configurações, recomendamos o Jupyter (*https://jupyter.org/*) instalado no próprio computador. Outras direções podem ser encontradas no site oficial *https://www.python.org/*.

6.2 Programação com Python

6.2.1 Convenções usadas

De início, os iniciantes em programação devem conhecer a forma de apresentação dos exemplos de códigos nesta dissertação (e em alguns livros). Seguindo a maioria dos estudiosos na área, os exemplos de códigos são mostrados envoltos em retângulo.

Não obstante, quando o código inserido exibe alguma saída, ele é mostrado com prompts de comando do Python, isto é, três símbolos de desigualdade seguidos. A saída é mostrada sem os símbolos. Por exemplo, o código abaixo é composto de duas linhas, todavia somente a primeira linha que deve ser digitada, sem o símbolo >>>. Uma vez digitado o código da primeira linha sem o símbolo, deve clicar em executar a célula

para a máquina devolver a frase seguinte `Hello world!`. Assim, vale dizer novamente, os símbolos de desigualdade servem para indicar tudo que deve ser digitado.

```
>>> print("Hello world!")
 Hello world!
```

Usamos a convenção tipográfica de `Fonte monoespaçada` para citar comandos ou elementos do programa, tais como: nomes de variáveis ou funções, banco de dados, tipos de dados, entre outros.

Se é a primeira vez que lida com Python, não deve pular o código acima. A função `print()` exibe algo na tela. Dentro dos parênteses é especificado o que será exibido. Neste caso, por ser um texto (string ou cadeia de caracteres), é envolto em aspas. Destarte, o texto `Hello world!` será exibido.

6.2.2 Importando bibliotecas

Construir um modelo de ML do início, tal como feito anteriormente, é precioso para entender o pleno funcionamento estrutural e selecionar melhores hiperparâmetros. Na prática, fórmulas, cálculos, treinamento de modelos, análise de erros e gráficos são executados por *frameworks* do Python. As bibliotecas, como são conhecidos, são estruturas bem projetadas que reduzem o uso de códigos e sumarizam os procedimentos. Existem milhares de bibliotecas, das quais usaremos principalmente a Scikit-Learn, que é simples e eficiente para análise preditiva de dados. Na prática, não se constrói um SVM partindo das equações, deixamos o trabalho árduo para o Scikit-Learn. GRUS (2019) afirma que diante de um problema real, você não escreve um algoritmo de otimização manualmente, recorre ao Scikit-Learn para escolher um excelente algoritmo. As bibliotecas NumPy (*http://numpy.org*), Matplotlib (*http://matplotlib.org*), Tqdm, Python Imaging Library (PIL) e OpenCV (CV2) também são exploradas.

A importação das bibliotecas deve ser explicitamente declarada com `import`, pois os recursos não são carregados por padrão. Por exemplo, o comando `import numpy as np` importa a biblioteca `numpy` com o apelido de `np`. Isso é especialmente útil para biblioteca de nome extenso, já que para chamar funções da biblioteca `numpy`, apenas escreva `np`. A mudança de nome é promovida pelo `as`, representando o advérbio como.

6.2.3 Comentários

É possível incluir comentários dentro do próprio código, para isso basta usar a hashtag. Comentários são úteis para anotações, instruções ou descrições pertinentes. Em geral, se houver uma hashtag, será ignorado tudo que se encontra depois da hashtag naquela linha.

6.2.4 Variáveis

As variáveis servem para armazenar e modificar valores em memória, podem ser do tipo inteiro, decimal (float), booleano, string, lista, entre outros.

Apesar de não existir na implementação, é comum surgir dúvida sobre o tipo do objeto. Neste caso, usamos `type()`. Além disso, para saber sobre a quantidade de itens de um objeto ou variável, use `len()`. Por exemplo, as linhas abaixo criam e informam o tipo da variável `x`

```
>>> x = 1
>>> type(x)
 int
```

"Python é uma linguagem tipada dinamicamente, ou seja, geralmente não se importa com os tipos dos objetos se eles forem utilizados de forma válida." (GRUS, 2019, p. 40). Acerca das expressões booleanas, o Python permite o uso de símbolos para operações lógicas, por exemplo:

```
>>> True and False          #Resulta em False
>>> True & False           #False, pois o símbolo & representa and
>>> 8 == 3                 #False, pois == testa a igualdade
>>> 18 % 2 == 0            #True já que 2 divide 18
>>> 3 % 2 == 0             #False, pois % é o resto da divisão
>>> 3 % 2 == 1             #True, o resto da divisão é 1
>>> (7>6) or (6>5)         #Resulta em True
>>> (7>6) | (6>5)          #True, pois | representa or
>>> print((5>4) & (7==7))
 True
```

6.2.5 A função range

A função `range()` serve para retornar uma série numérica no intervalo. Por padrão, inicia-se no 0 com passo 1, mas esses valores podem ser alterados. Por exemplo, o código abaixo retorna um intervalo numérico. Note que, para não retornar um objeto iterável, convertemos o objeto iterável em lista com o uso de `list()`.

```
>>> list(range(5))
 [0, 1, 2, 3, 4]
```

Vamos criar uma variável com muitos elementos e conhecer a quantidade com o uso de `len()`. Observe que, de fato, são 70mil elementos.

```
>>> X = list(range(70000))
>>> len(X)
 70000
```

Também podemos fazer desta forma:

```
>>> y = list(range(70*1000))
>>> len(y)
 70000
```

6.2.6 Lista

As listas são úteis por armazenar uma coleção ordenada em uma única variável. Por exemplo, a primeira linha do código abaixo cria uma lista com cinco elementos na variável `x` (caso esteja digitando os códigos deste tutorial em sequência, a variável `x` agora será uma lista e não um inteiro como antes). A segunda linha armazena na variável `fisrt` o primeiro elemento da lista, conhecido como posição zero, por isso o índice zero entre colchetes. A terceira linha exibe esse elemento.

```
>>> x = [10, 11, 12, 13, 14]
>>> first = x[0]
>>> print(first)
 10
```

Também é possível extrair elementos de uma lista, por meio de intervalo. "A fatia **i:j** contém todos os elementos de **i** (incluído) a **j** (não incluído). Se o início **i** não for indicado, ela começará no início da lista. Se o final **j** não for indicado, ela terminará no final da lista." (GRUS, 2019, p. 22). Por exemplo, a primeira linha do código abaixo armazena na variável `X_train` os elementos de `X` que ocupam até a posição sessenta mil. A segunda linha armazena na variável `y_train` os elementos de `y` da posição sessenta mil em diante.

```
>>> X_train = X[:60000]      #X_train tem 60k elementos
>>> y_train = y[60000:]      #y_train tem 10k elementos
```

O código acima não exibe saída, mesmo assim denotamos com os símbolos. O comando `X_train, y_train = X[:60000], y[60000:]` resume as duas linhas acima de forma mais elegante. Agora, se deseja adicionar um elemento na lista, use `.insert`. Por exemplo, o código a seguir adiciona o elemento `18` na posição `2` em `y_train`.

```
>>> y_train.insert(2, 18)
>>> len(y_train)           #para quantidade
 10001
```

6.2.7 Compreensão de lista

A compreensão de lista é uma forma elegante (Pythonic) de transformar uma lista em outra.

```
>>> lista1 = list(range(7))   #inteiros de 0 a 6 (inclusive)
>>> lista2 = [x % 2 == 1 for x in lista1]
>>> lista2
 [False, True, False, True, False, True, False]
```

Como se vê, o código acima cria uma lista com os inteiros de 0 a 6 (inclusive). Após, usando compreensão de lista, cria outra lista em que cada elemento é o resultado da sentença `x % 2 == 1`, iterando `x` como elemento da `lista1`. Em outras palavra, rotula `True` para números ímpares e `False` para outros números da `lista1`.

6.2.8 Utilizando bibliotecas

O código `fetch_openml("mnist_784", version=1, as_frame=False)` armazena parte do banco de dados `fetch_openml`. O banco de dados contém uma infinidade de informação, mas pouca revelante ao projeto. Assim, o conteúdo entre parênteses (parâmetros) especifica o que se busca no banco de dados. Eventualmente, as interfaces de programação de aplicação (do inglês *Application Programming Interface*, ou simplesmente API) passam por atualizações. Destarte, uma dica para ficar atualizado é pesquisar pelo nome do banco de dados e/ou função para conhecer todas as opções de parâmetros.

As funções `imshow()`, `axis()` e `show()` da biblioteca matplotlib, assim como no parágrafo anterior, necessitam de parâmetros dentro dos parênteses. Aqui vale a mesma regra, pesquisar pelo nome da função para ter acesso à documentação atualizada, mormente por existirem vários parâmetros omitidos que são preenchidos automaticamente com o padrão e podem mudar com atualizações.

6.2.9 NumPy

GRUS (2019) afirma que a biblioteca NumPy é especial por ser base de muitas outras bibliotecas, além de oferecer arrays com um ótimo desempenho. Dessarte, vamos apresentar algumas funções relevantes dessa biblioteca.

Para moldar um array sem alterar os dados, use `.reshape`. Por exemplo, o comando `x.reshape(28,28)` usa a função `.reshape` para remodelar `x` no formato 28 linhas e 28 colunas.

Para conhecer as dimensões do objeto use `.shape`. Por exemplo, o código abaixo cria um array `4x2` com o comando `np.array`. Apaga elementos das linhas um e dois

com `np.delete()`, sem prejudicar seu formato. Após, expõe suas dimensões com `.shape`. Observe que a primeira linha importa a biblioteca como `np`.

```
>>> import numpy as np
>>> x = np.array([[1,2],[3,4],[5,6],[7,8]])
>>> x = np.delete(x, [0,1], axis = 0)
>>> x.shape
 (2, 2)
```

Use o comando `shift(array,[vertical,horizontal])` para deslocar um array, onde `array` é o nome do array, `vertical` é o número inteiro do deslocamento vertical e `horizontal` é o número inteiro do deslocamento horizontal. Por exemplo, o código abaixo cria uma matriz identidade de ordem 5, desloca uma casa na horizontal e exibe o resultado.

```
>>> import numpy as np
>>> from scipy.ndimage import shift
>>> x = np.identity(5, dtype=int)    #matriz identidade ordem 5
>>> shift(x, [0,1])  #desloca +1 casa no eixo das abscissas
 array([[0, 1, 0, 0, 0],
 [0, 0, 1, 0, 0],
 [0, 0, 0, 1, 0],
 [0, 0, 0, 0, 1]])
```

6.2.10 Indentação

O Python adota o recuo para delimitar blocos de código. Os blocos podem ser afastados por espaço da margem esquerda e dispostos hierarquicamente. A indentação, uma espécie de parágrafo, facilita a visualização e percepção do todo.

6.2.11 Funções

As funções utilizam a indentação para hierarquia das instruções. O código abaixo cria a função usando `def`. A variável `r` armazena temporariamente o dobro da entrada. A terceira linha usa `return` para devolver o cálculo armazenado. Note o espaçamento entre a margem e o código na linhas dois e três. Sem o espaçamento, o código apresentará erro. Digitar quatro espaços seguidos ou a tabulação(tecla Tab) produz o espaçamento necessário.

```
>>> def f(x):
>>>     r = 2*x
>>>     return r
```

É mister notar que o código acima não produz efeito de saída, apenas define a função. Aplicar a função com um valor é intuitivo, basta chamar a função com o valor desejado.

```
>>> f(3)
 6
```

6.2.12 Iteráveis

Não raro, surge a necessidade de iterar em uma coleção. Para isso, use `for`, comando com indentação que itera sobre os elementos, veja:

```
>>> for i in range(4):
>>>     print(i)
 0, 1, 2, 3
```

6.2.13 Análise preditiva com Scikit-learn

Grande parte do projeto utiliza a Scikit-learn. GRUS (2019) afirma que é a biblioteca mais popular de ML. Almejamos que a parte de cálculo exaustivo seja executado por essa biblioteca. Não obstante, ainda que se defina o algoritmo (por exemplo SVM), existem muitos outros detalhes a se definir que a biblioteca não faz por conta própria (a biblioteca até tenta ajudar deixando alguns ajustes no padrão, que eventualmente servirão, contudo esse pode não ser o seu caso). Os hiperparâmetros são exemplos de ajustes que devem ser fornecidos. Para se manter atualizado consulte sempre a documentação oficial digitando o nome da biblioteca ou/e da função.

Observe que o código abaixo cria um conjunto de dados que será utilizado no seguinte, então, certifique-se de ter esse código digitado, antes de executar o posterior.

```
>>> c_treino = [[-1,0],[0,1],[1,2],[1,0],[2,1],[3,2]]
>>> r_treino = [False, False, False, True, True, True]
```

Como se vê, o código acima cria o conjunto de treinamento com os elementos caracterizados por: os três primeiros formam uma reta paralela à função identidade; os outros três formam outra reta paralela à função identidade. É imperioso notar que para os três primeiros elementos do conjunto de treinamento foi atribuído o rótulo `False`, para os outros, `True`.

Para treinar um algoritmo SVM classificador linear, use `LinearSVC` associado com `.fit`. Por exemplo, o código abaixo importa e treina os conjuntos.

```
>>> from sklearn.svm import LinearSVC
>>> lin_clf = LinearSVC()
>>> lin_clf.fit(c_treino, r_treino)
```

Após o treino, use `.predict` para ver a previsão de um elemento. Por exemplo, o código abaixo exibe a previsão do elemento `[0,2023]`.

```
>>> lin_clf.predict([[0,2023]])
 array([False])
```

Para saber qual o *score* em cada classe use `.decision_function`, como no exemplo a seguir.

```
>>> lin_clf.decision_function([[0,2023]])
 array([-1570.62078838])
```

O treinamento também pode ser feito usando outras estruturas como `SVC()`, ou `OneVsOneClassifier(SVC())`, ou ainda `OneVsOneClassifier(LinearSVC())`.

Para redução de dimensionalidade com a PCA, primeiro ajuste a taxa de variância `t` que se deve preservar usando `PCA(n_components=t)`, onde `t` é um número entre zero e um. Após, use `.fit_transform()`, preenchendo dentro dos parênteses com o objeto a ser compactado. O código a seguir cria uma matriz identidade de ordem 3, reduz a dimensionalidade com uma taxa de variância explicada de 95% e exibe suas dimensões. Note que a biblioteca cuida de centralizar o conjunto de dados em relação à origem, etapa essencial para o PCA.

```
>>> import numpy as np
>>> from sklearn.decomposition import PCA
>>> x = np.identity(3, dtype=int)    #matriz identidade ordem 3
>>> pca = PCA(n_components=0.95)
>>> x = pca.fit_transform(x)         #aplica PCA em x
>>> x.shape
 (3, 2)
```

Após ajustado, caso queira apenas compactar um objeto, use `.transform()`. Para descompactar use `.inverse_transform()`. Digite o código abaixo depois do anterior e observe a descompactação. Neste caso, a diagonal principal não sofreu alteração. Os outros elementos são quase nulos, tão próximos do zero quanto 10^{-15}.

```
>>> pca.inverse_transform(x)
 array([[ 1.00000000e+00,  0.00000000e+00, -5.55111512e-17],
 [-5.55111512e-17,  1.00000000e+00, -2.22044605e-16],
 [-5.55111512e-17, -2.22044605e-16,  1.00000000e+00]])
```

6.3 Use as funções criadas na etapa I da implementação

A etapa I da implementação cria funções que são úteis em outras etapas. Por exemplo, use `plot_precision_recall(y_train_imp, y_scores, r1, r2)` para encontrar a curva e os valores da precisão e revocação, onde os elementos entre parênteses são, respectivamente, os rótulos do modelo de classificação binária, os *scores* de decisão do modelo binário, o intervalo do gráfico à esquerda, e o intervalo do gráfico à direita.

Para o gráfico da Curva ROC use `plot_roc_curve(y_train_imp, y_scores)`, onde os elementos entre parênteses são os mesmos do parágrafo anterior. A área abaixo do gráfico pode ser obtida com `roc_auc_score()`, preenchendo com os mesmos elementos entre parênteses do comando anterior.

Para quantificar os erros nos conjuntos e obter a acurácia, use `test_pre(lin_clf)` e `train_pre(lin_clf)`, onde o conteúdo entre parênteses é o comando para o modelo. Após, use os resultados obtidos em `loss()`, preenchendo dentro dos parênteses com comando para o modelo, resultado da primeira função e resultado da segunda função.

A matriz de confusão pode ser visualizada usando `confusion_matrix(y_test, test_pred)`, onde os elementos entre parênteses são, respectivamente, o rótulo do conjunto de teste e o resultado da função `test_pre`.

Por fim, vale ressaltar que muitas funções exigem importações de bibliotecas explícitas na etapa I.

7 IMPLEMENTAÇÃO NO ENSINO MÉDIO: RECONHECIMENTO DE DÍGITO

Essa parte do projeto almeja materializar as ideias anteriores, culminando na parte computacional prática, tão importante quanto a teoria já exposta. Insta ressaltar que as implementações têm como público alvo estudantes de ensino médio.

Nesta implementação, o modelo deve ser capaz de predizer qual número foi representado na imagem com acurácia aceitável. Para isso vamos construir um modelo baseado em máquina de suporte vetorial, capaz de classificar *input* em 0, 1, 2, 3, 4, 5, 6, 7, 8 ou 9.

O conjunto de dados que usaremos é conhecido por MNIST, formado por 70 mil imagens de algarismos escritos à mão por funcionários do US Census Bureau e estudantes do ensino médio dos Estados Unidos. Cada imagem é rotulada com o algarismo que a representa, possui tamanho de 28x28 pixels, pode ser representada por uma matriz com 28 linhas e 28 colunas. Cada elemento da matriz é o valor da intensidade do pixel, variando entre os inteiros 0 e 255, onde 0 representa branco e 255 representa preto.

Figura 23 – MNIST: à esquerda na forma matricial, à direita na forma gráfica

Fonte: Elaborada pelo autor.

Também é possível representar cada imagem como uma matriz de apenas uma linha e 784 colunas, basta segmentar as linhas da representação matricial anterior e concatena-las ao final da primeira. Escolhemos esse conjunto de dados por ser público, robusto, de grande credibilidade e bastante famoso.

Esta implementação está dividida em etapas para melhor entendimento. Na etapa I importamos os dados e treinamos um modelo mais simples. Nas etapas II e IV treinamos modelos mais robustos em troca de maior tempo de processamento. A etapa III expande artificialmente o conjunto de treinamento. A etapa V aplica redução de dimensionalidade nos dados e treina o modelo. Finalmente, a etapa VI é interativa, na qual o discente pode testar o modelo na sua grafia. A menos que seja ministrado um curso

avançado de Python, as etapas I e VI não devem ser exercícios propostos, devem ser apresentadas como exercícios resolvidos, pois julgamos que ao discente não foi apresentado as ferramentas adequadas para a resolução. As outras etapas podem ser apresentadas como atividades propostas. Durante toda a implementação de processamento de imagens usaremos o modelo classificatório do SVM, ora linear, ora não.

Recomendação 1: As etapas devem ser seguidas, preferencialmente, na ordem em que aparecem. Não obstante, caso queira excluir alguma etapa, considere a etapa I imprescindível. A etapa III é necessária para a etapa IV e recomendada para a etapa V.
Recomendação 2: Caso esteja programando online, com Google Colab, considerando que a máquina local não tenha muito poder de processamento, é recomendado mudar a execução do Google Colab para uma GPU. Para fazer isso basta no menu superior ir em *Runtime*, em seguida clicar em *Change runtime type*, escolha *T4 GPU* e salve. Se não houver poder de processamento suficiente, a etapa IV consumirá muitas horas de execução e com o cálculo na GPU a execução ocorrerá em menos de 10 minutos.
Recomendação 3: Este trabalho foi pensado para permitir copiar e colar o código no Python. Infortunadamente, nem tudo é perfeito, as indentações após `for` e `def` não são passíveis de cópia. Em outras palavras, caso queira copiar o código, lembre-se de que deve colocar, por conta própria, as indentações após tais comandos. É importante destacar que somente as indentações devem ser ajustadas, a linguagem ignora os espaços entre as letras. A recomendação é obter o código pronto em: <https://bit.ly/implementacao_thesis> Caso o link não funcione, clique aqui que você será encaminhado direto ao código na página do Google Colab.

7.1 Etapa I: Classificação linear

7.1.1 Apresentação

Treine um modelo supervisionado de classificação linear multiclasse (os rótulos podem ser os inteiros entre 0 e 9, inclusive) do SVM para reconhecer um número, de apenas um algarismo, escrito à mão. Os dados devem ser importados do conjunto de dados MNIST. O programa deverá receber uma imagem e classificar qual número é representado na imagem. O cálculo do desempenho deve ser feito considerando o número de acertos e erros nos conjuntos. Para o gráfico da precisão, revocação e curva ROC, considere um modelo de classificação binária com os seguintes rótulos: número ímpar ou número não ímpar. A análise de erro pode ser realizada com a matriz de confusão para o conjunto de teste.

7.1.2 Duração

sugere-se cinco tempos de no mínimo 45 minutos cada

7.1.3 Objetivo

Apresentar aos alunos uma forma de uso de programação para o reconhecimento de número manuscrito. Será de suma importância durante a atividade o conhecimento em probabilidade de eventos equiprováveis, taxas de falsos positivos, de verdadeiros positivos, revocação, precisão, noções básicas de matriz e matriz de confusão.

7.1.4 Sugestão de resolução e considerações

7.1.4.1 Organizar os dados

De início, faz-se necessária a importação do conjunto de dados. Existem várias formas de fazer, uma delas é obter diretamente em `http://yann.lecun.com/exdb/mnist/`. Outra, mais simples, basta apenas importar do Scikit-Learn. O código a seguir extrai o conjunto de dados MNIST:

```
from sklearn.datasets import fetch_openml
import numpy as np

mnist = fetch_openml( "mnist_784", version= 1, as_frame= False)
X, y = mnist["data"], mnist["target"].astype(np.uint8)
```

Insta observar que `X` recebe um `array` com 70000 linhas (cada linha é uma instância, também chamado de vetor característica) e 784 colunas (cada coluna é uma característica). Esse `array` contém todas as *features* das imagens, veja sua forma:

```
>>> X.shape
 (70000, 784)
```

Não obstante, `y` recebe um `array` com todos os rótulos, ou seja, é de fato o que cada imagem representa. Como temos 70 mil imagens, espera-se que a forma de `y` seja:

```
>>> y.shape
 (70000,)
```

O código abaixo mostra a primeira instância remodelada para o formato 28x28 e processada pela biblioteca `matplotlib`. A última linha do código revela seu rótulo.

```
import matplotlib.pyplot as plt

n = 0
some_digit = X[n]
some_digit_image = some_digit.reshape(28, 28)
plt.imshow(some_digit_image, cmap="binary")
plt.axis("off")
```

```
plt.show()
print("veja o rótulo dessa instância: ", y[n])
```

Note que, alterar o valor de `n` na segunda linha mostrará outra instância e seu rótulo. Neste caso, fazemos mera conferência.

Uma forma mais rápida de observar instâncias:

```
for i in range(100):
some_digit = X[i]
some_digit_image = some_digit.reshape(28, 28)
plt.subplot(10, 10, i+1 )
plt.imshow(some_digit_image, cmap="binary")
plt.axis("off")
plt.rcParams["figure.figsize"] = [5, 5]
plt.show()
```

Nota-se que essa implementação é de aprendizagem supervisionada, pois todas as instâncias estão rotuladas. Assim, vamos separar o conjunto de treinamento do conjunto de teste:

```
X_train, y_train = X[:60000], y[:60000]
X_test, y_test = X[60000:], y[60000:]
```

É importante destacar que os dados do MNIST já sugerem a separação, pois as primeiras 60 mil imagens formam o conjunto de treinamento e as últimas 10 mil imagens o conjunto de teste. Ademais, o conjunto assegura um embaralhamento razoavelmente proporcional entre as entradas, o que é salutar em nosso modelo, mormente para alguns métodos de validação, como discutido anteriormente.

7.1.4.2 Treinar o modelo

Hora de treinar o modelo linear. Na prática, procuram-se os parâmetros que melhores se ajustam no conjunto de treinamento. Isso deveria ser uma tarefa árdua, porém, devido as ferramentas disponíveis na atualidade, precipuamente frameworks, simplesmente use o comando `LinearSVC`:

```
from sklearn.svm import LinearSVC
from sklearn.multiclass import OneVsOneClassifier

#vamos treinar o classificador SVM Linear
lin_clf = OneVsOneClassifier(LinearSVC(random_state=0))
lin_clf.fit(X_train, y_train)
```

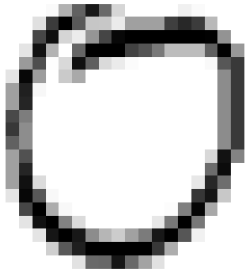
Insta ressaltar que usamos o classificador um contra um (OvO) por dois motivos. Primeiro, segundo Géron (2021), é mais rápido treinar muitos classificadores em pequenos conjuntos de treinamento do que treinar poucos classificadores em grandes conjuntos de treinamento. Segundo, a margem de separação é mais evidente entre apenas duas classes do que entre uma classe e todas as outras.

7.1.4.3 Espiar uma predição

Escolha uma instância no conjunto de teste e veja a predição que seu modelo fará. Observe se funcionará como esperado ou cometerá erro de predição. Abaixo escolhemos a instância `n=2023`. Inicialmente a imagem da instância, em seguida, a predição:

```
n = 2023
some_digit = X_test[n]
some_digit_image = some_digit.reshape(28, 28)
plt.imshow(some_digit_image, cmap="binary")
plt.axis("off")
plt.show()
```

```
lin_clf.predict([X_test[n]])    #predição
```



Nesse caso, o modelo acertou!!! Retornou um `array([0], dtype=uint8)`. O funcionamento do modelo OvO consiste em treinar 45 classificadores binários (zero contra um, zero contra dois...) e obter o *score* de decisão para a imagem a fim de selecionar a classe vencedora como predição. Veja o resumo do *score* por classe:

```
>>> lin_clf.decision_function([X_test[n]])
 array([[ 9.3260059, 1.67216723, 7.326801, 6.3239499, 0.671995,
 8.328487, -0.3269042, 3.6765925, 5.3243126, 2.678081]])| \\
```

O *score* está organizado em ordem crescente, iniciando pelo de classe 0. De fato, o *score* mais alto corresponde à classe correta. Quanto menor o *score*, menor a correlação com a classe. O classificador ficou com pequena dúvida em relação à classe 5. Por outra lado, está bastante confiante que não é classe 6, pois obteve *score* de aproximadamente -0,3.

7.1.4.4 Avaliar o modelo

Como saber se o modelo está bem ajustado, sem sobreajuste ou subajuste? Uma maneira de avaliar é observar o percentual de predições erradas nos conjuntos. Criar funções para essa tarefa é simples: a função `test_pre` retorna a predição de todas as instâncias do conjunto de teste, a função `train_pre` retorna a predição de todas as instâncias do conjunto de treinamento, por fim a função `loss` compara cada predição com seu próprio rótulo e imprimir o número de predições erradas nos conjuntos.

```
import tqdm

def test_pre(u):
test_pred_ = []
for i in tqdm.trange(len(y_test)):
test_pred_.append(u.predict([X_test[i]]))
return(test_pred_)
```

```
def train_pre(u):
train_pred_ = []
for i in tqdm.trange(len(y_train)):
train_pred_.append(u.predict([X_train[i]]))
return(train_pred_)

def loss(u, test_pre, train_pre):
con = [x for x in range(len(y_train)) if
train_pre[x] != y_train[x]]

print("Erros no conj de Treinamento: ", len(con))
print("Percentual: ", len(con)/len(y_train) )

con = [x for x in range(len(y_test)) if test_pre[x] !=
y_test[x]]

print("Erros no conj de teste: ", len(con))
print("Percentual: ", len(con)/len(y_test) )
print("Acurácia: ", 1- (len(con)/len(y_test)) )
```

Agora, aplicamos as funções no modelo linear classificatório:

```
>>> test_pred = test_pre(lin_clf)
>>> train_pred = train_pre(lin_clf)
>>> loss(lin_clf, test_pred, train_pred)
 Erros no conj de treinamento:   2233
 Percentual:  0.037216
 Erros no conj de teste:  829
 Percentual:  0.0829
 Acurácia:  0.9171
```

91% de acurácia, nada mal!!! Observe que os percentuais de erro no conjunto de treinamento e no conjunto de teste são moderadamente próximos, o que pode sugerir algum sobreajuste do modelo treinado.

No passado, definimos a precisão e revocação. Sabe-se que são utilizadas em classificadores binários. O modelo treinado é um classificador multiclasse, portanto para que a ferramenta trabalhe como esperado vamos reformular o funcionamento. Considerar todos os rótulos na dicotomia ou é ímpar ou não é ímpar parece adequado. Destarte, criar novo conjunto de rótulos que retorne `verdadeiro` para instâncias 1, 3, 5, 7 ou 9 e `false` para outras instâncias se faz necessário. Isso é realizado por `y_train_imp`, ao estilo pythônico (*pythonic*). Retreinaremos o modelo linear do SVM, agora chamado de `lin_clf_imp` para

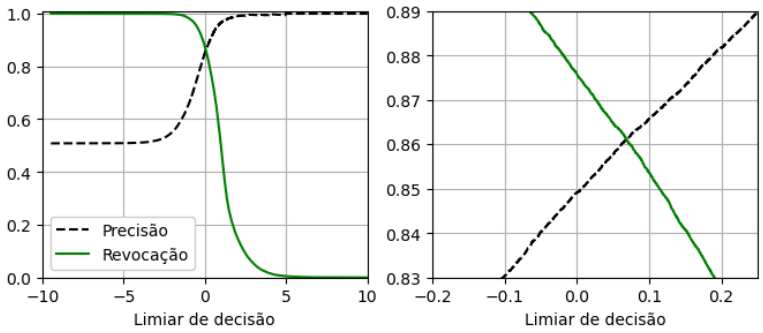
simular um classificador binário: ou é de classe ímpar, ou não é de classe ímpar. `y_scores` guarda o *score* de decisão de cada instância no conjunto de treinamento. `train_pred_imp` extrai as decisões a partir do *score*.

```
#queremos plotar a precisão, revocação e curva ROC
#para isso devemos criar etapas BINÁRIAS do classificador
y_train_imp = [x % 2 == 1 for x in y_train]

lin_clf_imp = LinearSVC(random_state=0)
lin_clf_imp.fit(X_train, y_train_imp)
y_scores = lin_clf_imp.decision_function(X_train)
train_pred_imp = (y_scores > 0)
```

Encontrar os valores e plotar o gráfico da precisão e revocação em função do limiar de decisão é simples, para isso criaremos uma função. Plotaremos dois gráficos idênticos, salvo que, um apresenta visão global e outro visão detalhada da região de encontro das funções.

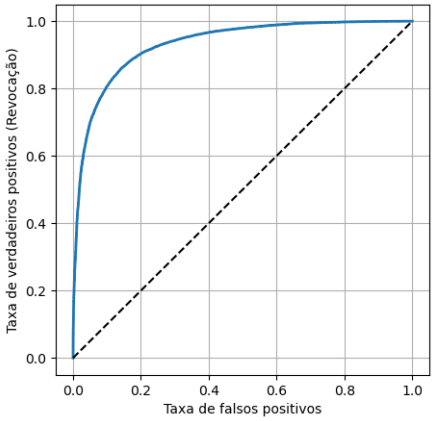
```
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import precision_recall_curve

def plot_precision_recall(y_train_imp, y_scores,
region1, region2):
precisions, recalls, thresholds = precision_recall_curve(
y_train_imp, y_scores)
plt.figure(figsize=(8,3))
plt.subplot(1,2,1)
plt.plot(thresholds,precisions[:-1],"k--",label="Precisão")
plt.plot(thresholds,recalls[:-1],"g-",label="Revocação")
plt.axis(region1)
plt.xlabel("Limiar de decisão")
plt.grid()
plt.legend(loc=3, framealpha=1)
plt.subplot(1,2,2)
plt.plot(thresholds, precisions[:-1],"k--",label="Precisão")
plt.plot(thresholds, recalls[:-1],"g-",label="Revocação")
plt.axis(region2)
plt.grid()
plt.xlabel("Limiar de decisão")
plt.show
print("Limiar em zero, a precisão é: ",
precision_score(y_train_imp, train_pred_imp))
print ("Limiar em zero, a revocação é: ",
```

```
recall_score(y_train_imp, train_pred_imp))

r1 = [-10, 10, 0, 1.01]
r2 = [-0.2, 0.25, 0.83, 0.89]
plot_precision_recall(y_train_imp,y_scores,r1,r2)
```



Com o limiar em zero, a precisão é próxima de $0,8491$ e a revocação $0,8758$. Note que as Figuras 5 e 6 são informações desse contexto. Para plotar o gráfico ROC, revocação em função da taxa de falsos positivos, definiremos uma função:

```
from sklearn.metrics import roc_curve

def plot_roc_curve(y_train_imp, y_scores, label=None):
fpr, tpr, thresholds = roc_curve(y_train_imp, y_scores)
plt.plot(fpr, tpr, linewidth=2, label=label)
plt.plot([0,1], [0,1], "k--") #diagonal tracejada
plt.grid()
plt.ylabel("Taxa de verdadeiros positivos(Revocação)")
plt.xlabel("Taxa de falsos positivos")
plt.show()

plot_roc_curve(y_train_imp, y_scores)
```



A linha contínua representa a curva ROC, a segmentada é referência. Em resumo, quanto mais próxima a curva é da borda superior esquerda, melhor o modelo. Uma forma de quantificar a curva ROC é calcular a área sob a curva. Um classificador perfeito tem área igual a 1, um classificador exclusivamente aleatório tem área igual a 0,5.

```
from sklearn.metrics import roc_auc_score

print("Área abaixo do gráfico: ", roc_auc_score(y_train_imp,
y_scores))
```

A área abaixo do gráfico é de 0,93u.a. Isso é um bom sinal.

7.1.4.5 Analisar o erro

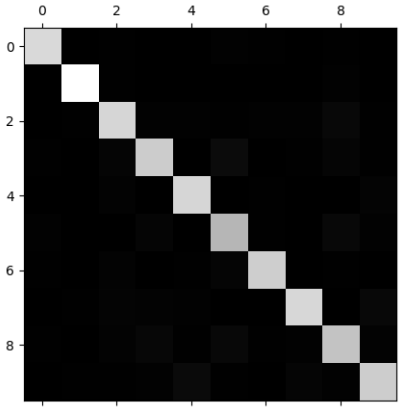
Outra métrica relevante para analisar um classificador multiclasse é a matriz de confusão. É possível aprimorar o desempenho do modelo, uma forma de faze-lo é analisar os tipos de erros para ajustes futuros.

```
#Matriz de confusão
from sklearn.metrics import confusion_matrix
conf_mx = confusion_matrix(y_test, test_pred)
conf_mx
```

```
[[ 946,    0,    7,    0,    0,   11,    7,    1,    6,    2]
[   0, 1111,    5,    3,    0,    1,    1,    1,   13,    0]
[   4,    8,  930,   11,   11,    8,   10,    9,   35,    6]
[   6,    1,   26,  887,    1,   52,    2,    6,   24,    5]
[   2,    0,   14,    0,  933,    1,    5,    2,    4,   21]
[  11,    3,    3,   24,    3,  791,    8,    2,   37,   10]
[   8,    1,   14,    2,    6,   22,  898,    0,    7,    0]
[   1,    5,   20,   15,    9,    0,    0,  935,    4,   39]
[   5,    3,   14,   31,    5,   38,    8,    9,  847,   14]
[   3,    5,    6,   10,   47,    7,    1,   24,   13,  893]]
```

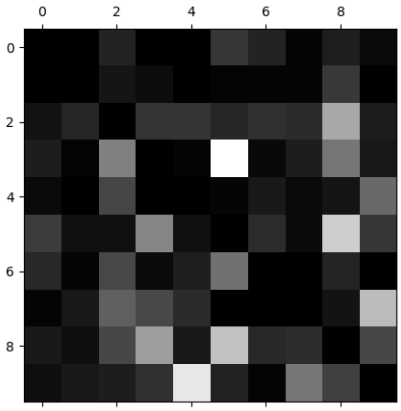
Observe que os maiores números estão na diagonal principal, indicando que os acertos são maiores que os erros. O maior elemento fora da diagonal principal localiza-se em 4x6. Isso significa que há bastante erro absoluto de predição quando trata-se do número 3 e o modelo acredita ser 5. Provavelmente existem muitas imagens dessa classe ou nosso modelo não funciona tão bem nessa classe. Talvez, visualizar a matriz de confusão em cores seja mais conveniente.

```
plt.matshow(conf_mx, cmap=plt.cm.gray)
plt.show()
```



As linhas correspondem classes reais, as colunas classes previstas pelo modelo. Quanto mais claro a imagem na diagonal principal, mais acertos absolutos. Os acertos são menores nas classes 5 e 8. Por fim, relativizar o erro torna-se indispensável para a comparação. Destarte, vamos dividir cada valor na matriz pelo número de imagens da classe afim.

```
row_sums = conf_mx.sum(axis=1, keepdims=True)
norm_conf_mx = conf_mx / row_sums
np.fill_diagonal(norm_conf_mx, 0)
plt.matshow(norm_conf_mx, cmap=plt.cm.gray)
plt.show()
```



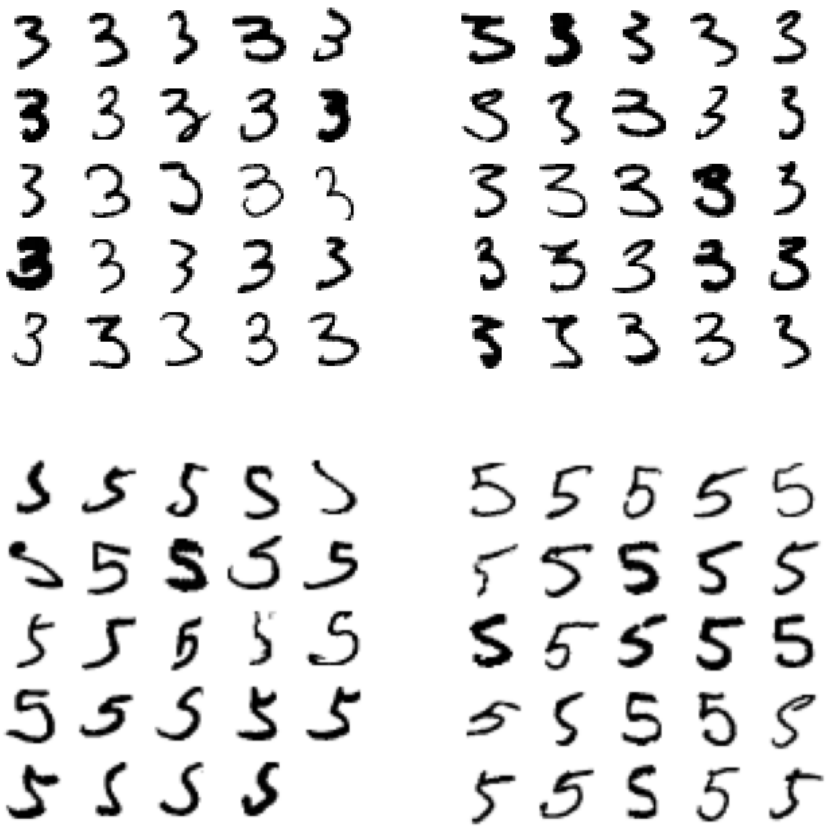
Novamente, as linhas correspondem classes reais, ao passo que colunas classes previstas pelo modelo. Não obstante, a coluna da classe 5 é moderadamente clara, mormente na linha da classe 3, sugerindo índice alto para erro de predição da classe 3 que o modelo prediz 5. Para observar melhor, plotaremos algumas imagens da classe 3 e 5:

```
import matplotlib as mpl

def plot_digits(instances, images_per_row=10, **options):
size = 28
images_per_row = min(len(instances), images_per_row)
```

```
images = [instance.reshape(size,size) for instance in
instances]
n_rows = (len(instances) - 1) // images_per_row + 1
row_images = []
n_empty = n_rows * images_per_row - len(instances)
images.append(np.zeros((size, size * n_empty)))
for row in range(n_rows):
rimages = images[row * images_per_row : (row + 1) *
images_per_row]
row_images.append(np.concatenate(rimages, axis=1))
image = np.concatenate(row_images, axis=0)
plt.imshow(image, cmap = mpl.cm.binary, **options)
plt.axis("off")

def show_digits(digit1, digit2):
mixx = np.squeeze(test_pred)
cl_a, cl_b = digit1, digit2
X_aa = X_test[(y_test == cl_a) & (mixx == cl_a)]
X_ab = X_test[(y_test == cl_a) & (mixx == cl_b)]
X_ba = X_test[(y_test == cl_b) & (mixx == cl_a)]
X_bb = X_test[(y_test == cl_b) & (mixx == cl_b)]
plt.figure(figsize=(8,8))
plt.subplot(221); plot_digits(X_aa[:25], images_per_row=5)
plt.subplot(222); plot_digits(X_ab[:25], images_per_row=5)
plt.subplot(223); plot_digits(X_ba[:25], images_per_row=5)
plt.subplot(224); plot_digits(X_bb[:25], images_per_row=5)
plt.show()

show_digits(3, 5)
```

Se observar os manuscritos abaixo em blocos, organizam-se como uma matriz 2x2. Neste sentido, as imagens da diagonal principal foram assinaladas corretamente pelo modelo, ao passo que os blocos superior direito e inferior esquerdo perfazem predições errôneas. De fato, há alguma promiscuidade no manuscrito, onde até mesmo um humano teria dificuldade em reconhecer. Note que há somente imagens de três e cinco.

Insta ressaltar que a matriz de confusão foi construída no conjunto de teste, destarte analisamos erros no conjunto de teste. Não obstante, pode ser útil construir a matriz de confusão para o conjunto de treinamento. Todavia, analisar o erro com dados em que o modelo já foi treinado (conjunto de treinamento) pode trazer tendência à análise, se não interpretada dentro do contexto.



7.2 Etapa II: Classificação não linear

7.2.1 Apresentação

Treine um modelo supervisionado de classificação não linear multiclasse (os rótulos podem ser os inteiros entre 0 e 9, inclusive) do SVM para reconhecer um número, de apenas um algarismo, escrito à mão. Os dados devem ser importados do conjunto de dados MNIST, como realizado na etapa anterior. O programa deverá receber uma imagem e classificar qual número é representado na imagem. O cálculo do desempenho deve ser feito considerando o número de acertos e erros nos conjuntos. Para o gráfico da precisão, revocação e curva ROC, considere um modelo de classificação binária com os seguintes rótulos: número ímpar ou número não ímpar. A análise de erro pode ser realizada com a matriz de confusão para o conjunto de teste.

7.2.2 Duração

sugere-se quatro tempos de, no mínimo, 45 minutos cada

7.2.3 Sugestão de resolução e considerações

Como informado anteriormente, a etapa I é essencial por importar os dados e as bibliotecas utilizadas doravante.

7.2.3.1 Treinar o modelo

Os dados já foram importados e organizados, destarte para ajusta-los ao modelo use o comando `SVC`. Ao treinar um modelo não linear, há maior tempo de processamento frente ao linear. O Scikit-learn cuida de procurar os coeficientes que melhores se ajustam à forma genérica do SVM.

```
from sklearn.svm import SVC

svm_clf = OneVsOneClassifier(SVC(random_state=0), n_jobs=-1)
svm_clf.fit(X_train, y_train)
```

7.2.3.2 Espiar uma predição

Escolha uma instância no conjunto de teste e veja a predição que seu modelo fará. Observe se funcionará como esperado ou cometerá erro de predição. Abaixo escolhemos a instância `n=1687`, ano da publicação de *Principia*, principal obra de Isaac Newton. Inicialmente a imagem da instância, em seguida da predição:

```
n = 1687
some_digit = X_test[n]
some_digit_image = some_digit.reshape(28, 28)
plt.imshow(some_digit_image, cmap="binary")
plt.axis("off")
plt.show()

svm_clf.predict([X_test[n]])    #predição
```

Apesar de o algarismo não estar bem legível, o modelo acertou!!! Se houver dúvida, use o comando `print(y_test[n])` para saber qual é o algarismo representado. O modelo retornou um `array([8], dtype=uint8)`. O funcionamento do modelo OvO (*one-versus-one*) consiste em treinar 45 classificadores binários (zero contra um, zero contra dois...) e obter o *score* de decisão para a imagem a fim de selecionar a classe vencedora como predição. Veja o *score* por classe:

```
>>> svm_clf.decision_function([X_test[n]])
array([[ 0.69936312, 8.28966103, 3.76575871, 7.2792078, 3.7287,
1.7583041, -0.3054874, 3.757808, 9.315962, 6.2295871]])
```

De fato, o *score* mais alto corresponde à classe correta. Quanto menor o *score*, menor a correlação com a classe. O classificador ficou com pequena dúvida em relação à classe 1. Por outra lado, está bastante confiante que não é classe 6, pois obteve *score* de aproximadamente -0,3.

7.2.3.3 Avaliar o modelo

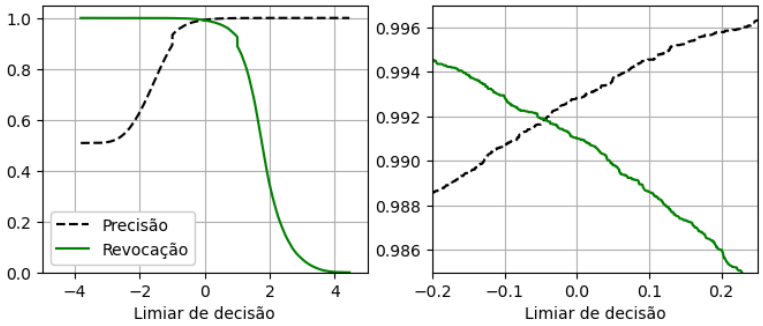
Como saber se o modelo está bem ajustado, sem sobreajuste ou subajuste? Uma maneira de avaliar é observar o percentual de predições erradas nos conjuntos. Na etapa I, em avaliar o modelo, definimos as funções `test_pred`, `train_pred` e `loss`, por isso é essencial que haja as funções no código a fim de evitar problema. Destarte, resta apenas aplicar as funções:

```
>>> test_pred = test_pre(svm_clf)
>>> train_pred = train_pre(svm_clf)
>>> loss(svm_clf, test_pred, train_pred)
Erros no conj de treinamento:  618
Percentual:  0.0103
Erros no conj de teste:  204
Percentual:  0.0204
Acurácia:  0.9796
```

Observe que 97% de acurácia é consideravelmente superior à 91% obtido no modelo linear. Isso se deve ao fato de o modelo não linear ter mais coeficientes para serem ajustados aos dados, permitindo melhor separação das classes.

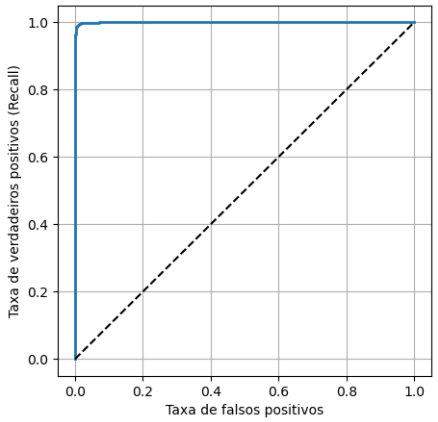
É mister a adaptação para classificador binário, com o propósito de avaliar o modelo a partir da curva de característica de operação (ROC), da precisão e revocação. Ou seja, considere todos os rótulos na dicotomia `verdadeiro` para ímpares e `false` para outras classes. Lembre-se de que anteriormente importamos as bibliotecas e criamos etapas binárias e funções, isso é essencial para correto funcionamento. Fazer o retreinamento, encontrar os *scores* para obter os valores e o gráfico de revocação e precisão:

```
svm_clf_imp = SVC(random_state=0)
svm_clf_imp.fit(X_train, y_train_imp)
y_scores = svm_clf_imp.decision_function(X_train)
train_pred_imp = (y_scores > 0)

r1 = [-5, 5, 0, 1.05]
r2 = [-0.2, 0.25, 0.985, 0.997]
plot_precision_recall(y_train_imp, y_scores, r1, r2)
```



Com o limiar em zero, a precisão é aproximadamente $0,9928$ e revocação $0,9909$. Plotar a curva ROC, revocação em função da taxa de falsos positivos:

```
plot_roc_curve(y_train_imp, y_scores)
```



A linha contínua representa a curva ROC, a segmentada é referência. Em resumo, quanto mais próxima a curva é da borda superior esquerda, melhor o modelo. Uma forma de quantificar a curva ROC é calcular a área sob a curva. Um classificador perfeito tem área igual a 1, um classificador exclusivamente aleatório tem área igual a 0,5.

```
>>> roc_auc_score(y_train_imp, y_scores)
 0.9994203237849304
```

A área abaixo do gráfico maior que 0,9994u.a. revela maior assertividade do modelo não linear no conjunto de treinamento. Não obstante, de forma análoga, é possível você fazer para o conjunto de teste.

7.2.3.4 Analisar o erro

No item acima avaliamos o modelo na ótica binária, doravante analisaremos na perspectiva multiclasse. Construir a matriz de confusão associada é simples. Note que a matriz será construida no conjunto de teste, por isso os números são menores.

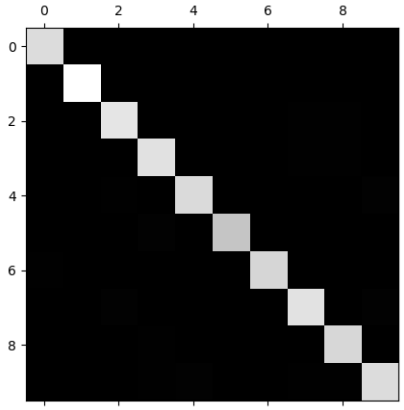
```
#Matriz de confusão para o Conjunto de Teste

conf_mx = confusion_matrix(y_test, test_pred)
conf_mx
```

```
[[ 972,    0,    2,    0,    0,    2,    1,    1,    2,    0]
[   0, 1126,    3,    1,    0,    1,    1,    0,    2,    1]
[   4,    0, 1008,    2,    1,    0,    2,    7,    7,    1]
[   0,    0,    1,  994,    0,    2,    0,    5,    6,    2]
[   0,    0,    5,    0,  961,    0,    3,    0,    2,   11]
[   2,    0,    0,    9,    0,  870,    4,    1,    4,    2]
[   5,    2,    0,    0,    2,    3,  945,    0,    1,    0]
[   0,    4,   10,    1,    1,    0,    0,  998,    3,   11]
[   3,    0,    2,    5,    3,    3,    2,    3,  950,    3]
[   3,    4,    1,    5,   10,    2,    1,    6,    5,  972]]
```

Observe que as linhas correspondem classes reais, as colunas classes previstas. Assim, os maiores números na diagonal principal mostram que os acertos são maiores que os erros. O maior elemento fora da diagonal principal é o 11. Localiza-se esse elemento em 5x10. Isso significa que há bastante erro absoluto de predição quando trata-se do número 4 e o modelo acredita ser 9. Provavelmente existem muitas imagens dessa classe ou nosso modelo não funciona tão bem nessa classe. Talvez, visualizar a matriz de confusão em cores seja mais conveniente. Não deixe de comparar com a matriz da etapa anterior.

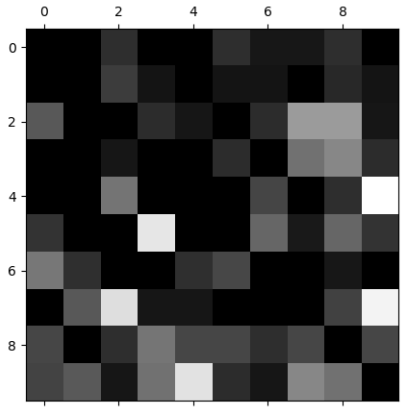
```
plt.matshow(conf_mx, cmap=plt.cm.gray)
plt.show()
```

Na figura abaixo, as linhas correspondem classes reais, as colunas classes previstas pelo modelo. Quanto mais claro a imagem na diagonal principal, mais acertos absolutos.



Por fim, relativizar o erro torna-se indispensável para a comparação. Destarte, vamos dividir cada valor na matriz pelo número de imagens da classe afim.

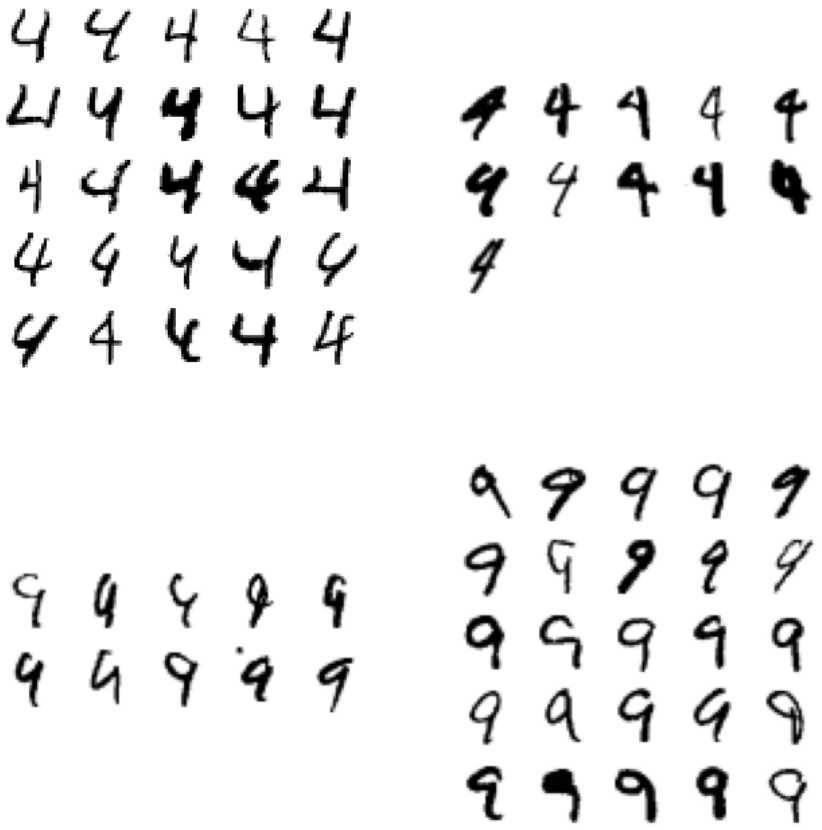
```
row_sums = conf_mx.sum(axis=1, keepdims=True)
norm_conf_mx = conf_mx / row_sums
np.fill_diagonal(norm_conf_mx, 0)
plt.matshow(norm_conf_mx, cmap=plt.cm.gray)
plt.show()
```



Igualmente, as linhas correspondem classes reais e colunas as previstas pelo modelo. Cor clara indica maior erro de predição. Não obstante, o final da coluna da classe 4 e o final da linha da classe 4 são tendentes ao branco. Dessa forma, é imperioso notar que ocorre bastante equívoco quando envolvem as classes 4 e 9 na predição e no rótulo mutuamente. Para análise minuciosa, plotaremos algumas imagens dessas classes:

```
show_digits(4, 9)
```

A seguir há quatro blocos de imagens, em todos há somente algarismos quatro ou nove. O bloco superior esquerdo e inferior direito foram preditos corretamente pelo modelo. O bloco superior direito foi erroneamente classificado pelo modelo como nove. O



bloco inferior esquerdo foi erroneamente classificado pelo modelo como quatro. De fato, há notória promiscuidade no manuscrito, onde até mesmo um humano teria dificuldade em reconhecer. Caso queria retrabalhar nos dados, é preciso eliminar alguns ruídos, trataremos disso na próxima etapa.

7.3 Etapa III: Expansão do conjunto de treinamento

7.3.1 Apresentação

Manipule o conjunto de dados MNIST a fim de adequá-lo a sua necessidade. Elimine os dados de baixa qualidade. Para cada imagem do conjunto de treinamento, crie quatro novas imagens a partir das existentes, deslocando o algarismo em 1 pixel para direita, para esquerda, para cima e para baixo. Dessas quatro novas imagens, crie uma para cada direção. Organize os rótulos dos novos dados criado artificialmente.

7.3.2 Duração

sugere-se quatro tempos de no mínimo 45 minutos cada

7.3.3 Objetivo

Apresentar aos alunos uma forma de uso de programação para ampliar artificialmente o conjunto de dados. Melhorar a performance do modelo sem a coleta de novos dados. Manipular computacionalmente os dados para transladar os algarismos. Será de suma importância durante a atividade noções básicas em matriz e operações matriciais.

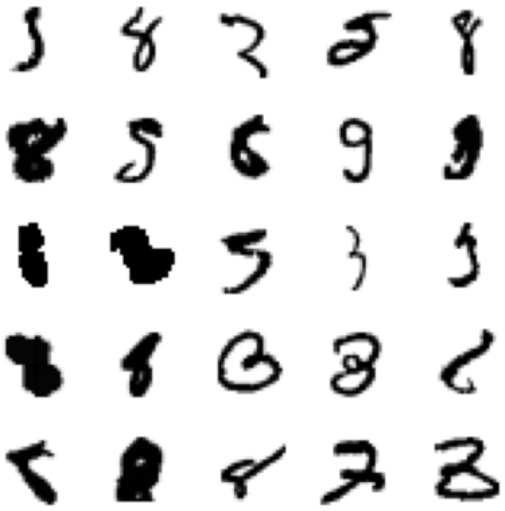
7.3.4 Sugestão de resolução e considerações

7.3.4.1 Eliminar dados de baixa qualidade

De início, faz-se necessário adequar os dados ao contexto da aplicação da inteligência artificial. Os dados devem ser representativos, portanto devem guardar íntima relação com a realidade. Assim, é imperioso conhecer a realidade de onde se quer implementar.

Além disso, os dados não devem ser de baixa qualidade. Neste sentido, é oportuno eliminar ruido, mormente se for implementado com pessoas de boa grafia. Em etapas anteriores, observaram-se dados de baixa qualidade. Não obstante, convidamos o leitor a apreciar alguns dados por meio do seguinte código:

```
i = 0

ruido = (132,160,500,635,720,756,1032,1404,10994,11781,16376,
16676,16678,18814,24059,25018,26852,32276,33412,34328,
34758,37198,39355,50340,56228)

for x in ruido:
some_digit = X_train[x]
some_digit_image = some_digit.reshape(28, 28)
plt.subplot(5, 5, i+1 )
plt.imshow(some_digit_image, cmap="binary")
plt.axis("off")
plt.rcParams["figure.figsize"] = [5, 5]
i = i+1

plt.show()
```



Tente identificar quais são os algarismos da figura acima. Provavelmente o único algarismo que o leitor está convencido a acertar é o localizado na segunda linha e quarta coluna. Infortunadamente, o que parece ser um nove é um três (e o algarismo central não é um cinco!). Com isso, torna-se patente que existem ruídos. No próximo código,vamos eliminar esses e outros dados de baixa qualidade.

```
X_train = np.delete(X_train, [137,160,228,368,376,456,500,
503,517,528,635,720,756,817,935,1024,1032,1120,1207,1239,
1276,1325,1328,1329,1339,1341,1384,1402,1509,1598,1670,1702,
1758,1873,1948,2000,2098,2150,2250,2444,2554,2622,2636,2707,
2720,2818,2901,2933,2946,3205,3219,3220,3225,3243,3271,3275,
3289,3535,3570,3634,3682,3810,3907,3928,4068,4077,4136,4148,
4167,4480,4506,4562,4638,4652,4692,4742,4908,5034,5148,5158,
5177,5332,5346,5408,5440,5562,5598,5616,5799,5896,5971,5972,
6026,6066,6195,6418,6506,6610,6636,6746,6808,6816,6905,7042,
7101,7190,7300,7308,7606,7642,7768,7851,7909,7995,8029,8031,
8033,8047,8210,8465,8666,8690,8701,8719,8730,8761,8785,8859,
8867,8889,9220,9952,10064,10281,10319,10800,11210,11351,
11482,1080,2823,3430,3456,3897,3898,6226,7599,7898,8219,8689,
12078,12113,12207,12805,13259,15730,16130,17005,17129,17130,
17209,17213,17706,18293,19782,20322,21944,23665,23693,23703,
23727,23740,23749,23751,25397,25414,25416,25421,25428,25434,
27068,27071,27091,27104,27111,28828,28831,28834,28836,28837,
28857,32353,32358,32365,32367,32705,32710,32718,32721,35360,
35820,35826,35829,35831,35856,35862,36744,37725,37728,37732,
37741,37742,37761,37782,39403,39417,39437,39441,39444,39451,
41150,41193,41195,41205,41209,42655,42657,42666,42673,42834,
42837,42846,42852,44648,44650,44661,44677,44679,44680,44682,
```

```
44693,46452,46461,46468,46470,46481,46496,46499,48232,48236,
48253,48254,48261,48266,50139,50150,50158,50190,50194,50207,
50219,50572,51901,51905,51908,51932,51936,228,1068,1077,
1097,1244,1341,1357,1378,1476,1634,2023,2200,2580,2652,2720,
2774,2803,2818,2960,3034,3220,3358,3415,3435,3462,3471,3491,
3756,3810,4068,4156,4506,4638,4694,4742,4762,5086,5148,5238,
5332,5408,5562,5616,5738,5770,5787,5798,5799,5831,5923,6251,
6269,6291,6297,6347,6362,6920,7080,7192,7584,7784,7803,9078,
9324,9516,9602,9932,10091,10282,10470,10984,11097,11104,
11781,11949,12153,12183,12416,12967,13109,13376,14692,15774,
16530,19215,20016,20150,22204,22643,22675,23962,25034,26629,
27172,28374,28392,31962,34660,38520,38990,43562,43658,43865,
44456,46246,46689,46857,47519,47689,50369,50431,50500,54782,
54858,14896,16658,17244,17602,17756,19124,26865,29026,29179,
29259,31800,32018,35246,35916,39291,39307,41218,44806,47034,
47634,48824,49088,51315,51576,53236,53854,54362,56286,16376,
500,10994,16676,16678,18814,24059,32276,33412,50340,56228,
132,1404,25018,26852,34328,34758,37198,39355,], axis = 0)

#Não se esqueça de excluir os rótulos
y_train = np.delete(y_train, [137,160,228,368,376,456,500,
503,517,528,635,720,756,817,935,1024,1032,1120,1207,1239,
1276,1325,1328,1329,1339,1341,1384,1402,1509,1598,1670,1702,
1758,1873,1948,2000,2098,2150,2250,2444,2554,2622,2636,2707,
2720,2818,2901,2933,2946,3205,3219,3220,3225,3243,3271,3275,
3289,3535,3570,3634,3682,3810,3907,3928,4068,4077,4136,4148,
4167,4480,4506,4562,4638,4652,4692,4742,4908,5034,5148,5158,
5177,5332,5346,5408,5440,5562,5598,5616,5799,5896,5971,5972,
6026,6066,6195,6418,6506,6610,6636,6746,6808,6816,6905,7042,
7101,7190,7300,7308,7606,7642,7768,7851,7909,7995,8029,8031,
8033,8047,8210,8465,8666,8690,8701,8719,8730,8761,8785,8859,
8867,8889,9220,9952,10064,10281,10319,10800,11210,11351,
11482,1080,2823,3430,3456,3897,3898,6226,7599,7898,8219,8689,
12078,12113,12207,12805,13259,15730,16130,17005,17129,17130,
17209,17213,17706,18293,19782,20322,21944,23665,23693,23703,
23727,23740,23749,23751,25397,25414,25416,25421,25428,25434,
27068,27071,27091,27104,27111,28828,28831,28834,28836,28837,
28857,32353,32358,32365,32367,32705,32710,32718,32721,35360,
35820,35826,35829,35831,35856,35862,36744,37725,37728,37732,
37741,37742,37761,37782,39403,39417,39437,39441,39444,39451,
41150,41193,41195,41205,41209,42655,42657,42666,42673,42834,
42837,42846,42852,44648,44650,44661,44677,44679,44680,44682,
```

```
44693,46452,46461,46468,46470,46481,46496,46499,48232,48236,
48253,48254,48261,48266,50139,50150,50158,50190,50194,50207,
50219,50572,51901,51905,51908,51932,51936,228,1068,1077,
1097,1244,1341,1357,1378,1476,1634,2023,2200,2580,2652,2720,
2774,2803,2818,2960,3034,3220,3358,3415,3435,3462,3471,3491,
3756,3810,4068,4156,4506,4638,4694,4742,4762,5086,5148,5238,
5332,5408,5562,5616,5738,5770,5787,5798,5799,5831,5923,6251,
6269,6291,6297,6347,6362,6920,7080,7192,7584,7784,7803,9078,
9324,9516,9602,9932,10091,10282,10470,10984,11097,11104,
11781,11949,12153,12183,12416,12967,13109,13376,14692,15774,
16530,19215,20016,20150,22204,22643,22675,23962,25034,26629,
27172,28374,28392,31962,34660,38520,38990,43562,43658,43865,
44456,46246,46689,46857,47519,47689,50369,50431,50500,54782,
54858,14896,16658,17244,17602,17756,19124,26865,29026,29179,
29259,31800,32018,35246,35916,39291,39307,41218,44806,47034,
47634,48824,49088,51315,51576,53236,53854,54362,56286,16376,
500,10994,16676,16678,18814,24059,32276,33412,50340,56228,
132,1404,25018,26852,34328,34758,37198,39355,], axis = 0)
```

Impende destacar que procurou-se manter a probabilidade de cada classe, com fito de diminuir *viés*.

7.3.4.2 Expansão do conjunto de treinamento

O objetivo aqui é inserir novos dados no conjunto de treinamento sem coletar novas imagens. Manipular os dados artificialmente para criar cópias alteradas é uma boa forma de cumprir o objetivo.

Inicialmente, usar `reshape` para representação matricial 28x28. Além disso, para cada instância no conjunto de treinamento, mover as imagens com a função `shift` do módulo `scipy.ndimage` para deslocar em 1 pixel para cada direção. Após, retornar para o formato original e usar `append` para adicionar cada cópia.

```
from scipy.ndimage import shift
XX_train = [] #criar lista vazia
for x in range(len(y_train)):
deslocar = X_train[x].reshape(28,28) #matriz 28x28
deslocar_d = shift(deslocar, [1,0])   #deslocar 1 baixo
deslocar_u = shift(deslocar, [-1,0])  #deslocar 1 cima
deslocar_r = shift(deslocar, [0,1])   #deslocar 1 dir
deslocar_l = shift(deslocar, [0,-1])  #deslocar 1 esq

deslocar_d = deslocar_d.reshape(784) #volta ao formato
```

```
deslocar_u = deslocar_u.reshape(784)
deslocar_r = deslocar_r.reshape(784)
deslocar_l = deslocar_l.reshape(784)

XX_train.append(X_train[x])           #adicionar o orig
XX_train.append(deslocar_d.flatten())
#array.flatten() Transforma para array de 1 dimensão!
XX_train.append(deslocar_u.flatten())
XX_train.append(deslocar_r.flatten())
XX_train.append(deslocar_l.flatten())

X_train = np.array(XX_train)
```

É mister observar que doravante `X_train` é o conjunto de treinamento expandido artificialmente. Assim, esse conjunto é consideravelmente maior que o anterior.

7.3.4.3 Organizar os rótulos

Com efeito da mudança no conjunto de treinamento, é mister organizar os rótulos a fim de manter a aprendizagem supervisionada.

```
yy_train = []

for x in range(len(y_train)):
yy_train.insert(x*5, y_train[x])
yy_train.insert(x*5, y_train[x])
yy_train.insert(x*5, y_train[x])
yy_train.insert(x*5, y_train[x])
yy_train.insert(x*5, y_train[x])

y_train = np.array(yy_train)
```

Note que, doravante, `y_train` armazena os rótulos correspondentes ao conjunto de treinamento expandido artificialmente.

7.4 Etapa IV: SVM no conjunto expandido

7.4.1 Apresentação

Treine um modelo supervisionado de classificação não linear multiclasse (os rótulos devem ser os inteiros entre 0 e 9, inclusive) do SVM para reconhecer um número, de apenas um algarismo, escrito à mão. Use os dados importados do conjunto de dados

MNIST e expandido artificialmente, como realizado na etapa anterior. O programa deverá receber uma imagem e classificar qual número é representado na imagem. O cálculo do desempenho deve ser feito considerando o número de acertos e erros nos conjuntos. Para o gráfico da precisão, revocação e curva ROC, considere um modelo de classificação binária com os seguintes rótulos: número ímpar ou número não ímpar. A análise de erro pode ser realizada com a matriz de confusão para o conjunto de teste.

7.4.2 Duração

Sugere-se cinco tempos de, no mínimo, 45 minutos cada.

7.4.3 Objetivo

Apresentar aos alunos uma forma de uso de programação para o reconhecimento de número manuscrito. Será de suma importância durante a atividade o conhecimento em probabilidade de eventos equiprováveis, taxas de falsos positivos, de verdadeiros positivos, revocação, precisão, noções básicas de matriz e matriz de confusão.

7.4.4 Sugestão de resolução e considerações

Como informado anteriormente, a etapa anterior é essencial por importar os dados, remover ruídos, expandir artificialmente e importar as bibliotecas utilizadas doravante.

7.4.4.1 Treinar o modelo

Os dados foram exaustivamente trabalhados, destarte para ajusta-los ao modelo use o comando `SVC`. Ao treinar um modelo não linear com extenso conjunto de dados, como o apresentado, há severamente maior tempo de processamento. Tenha em mente que essa etapa consome bastante tempo de processamento. Para uma estimativa do tempo, no meu caso, demorou cerca de uma hora, com o atual processador i7 e 32GB de RAM. O módulo do Scikit-learn cuida de procurar os coeficientes que melhores se ajustam à forma genérica do SVM.

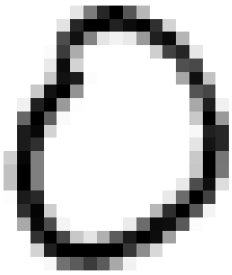
```
from sklearn.svm import SVC
svm_clf = SVC(random_state=0)
svm_clf.fit(X_train, y_train)
```

7.4.4.2 Espiar uma predição

Escolha uma instância no conjunto de teste e veja a predição que seu modelo fará. Observe se funcionará como esperado ou cometerá erro de predição. Abaixo escolhemos a

instância `n=1777`, ano de nascimento do *Príncipe dos Matemáticos*, Carl Friedrich Gauss. Inicialmente a imagem da instância, em seguida da predição:

```
n = 1777
some_digit = X_test[n]
some_digit_image = some_digit.reshape(28, 28)
plt.imshow(some_digit_image, cmap="binary")
plt.axis("off")
plt.show()

svm_clf.predict([X_test[n]])    #predição
```



O modelo acertou!!! Se houver dúvida, use o comando `print(y_test[n])` para saber qual é o algarismo representado. O modelo retornou um `array([0], dtype=uint8)`. O funcionamento do modelo OvR (*one-versus-rest*) consiste em treinar 10 classificadores binários (zero contra não zero, um contra não um...) e obter o *score* de decisão para a imagem a fim de selecionar a classe vencedora como predição. Veja o *score* por classe:

```
>>> svm_clf.decision_function([X_test[n]])
array([[ 9.3151376, -0.3054926, 5.2532182, 2.7421357, 0.697794,
7.2869575, 8.2946583, 1.7154973, 3.7349534, 5.891824]])
```

De fato, o *score* mais alto corresponde à classe correta. Quanto menor o *score*, menor a correlação com a classe. O classificador ficou com pequena dúvida em relação à classe 6. Por outra lado, está bastante confiante que não é classe 1, pois obteve *score* de aproximadamente -0,3.

7.4.4.3 Avaliar o modelo

Como saber se o modelo está bem ajustado, sem sobreajuste ou subajuste? Uma maneira de avaliar é observar o percentual de predições erradas nos conjuntos. Já criamos uma função para essa tarefa na etapa I, por isso é essencial que haja as funções no código a fim de evitar erro. Destarte, resta apenas aplicar as funções no modelo. Novamente, é

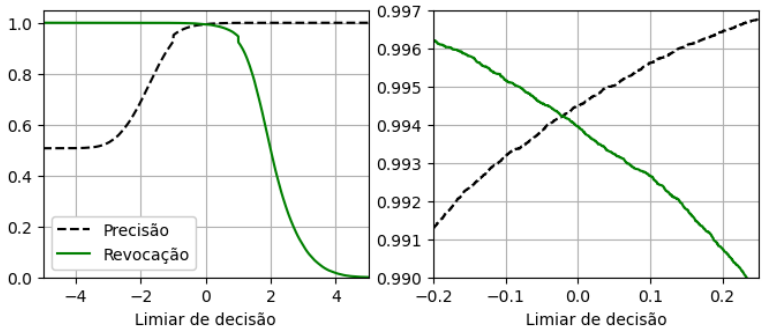
necessária muita paciência ao lidar com grande volume de dados, no meu caso, quase duas horas de processamento.

```
>>> test_pred = test_pre(svm_clf)
>>> train_pred = train_pre(svm_clf)
>>> loss(svm_clf, test_pred, train_pred)
número de erros no conj de Treinam:  1643
percentual:  0.005514440584671668
número de erros no conj TESTE:  118
percentual:  0.0118
Acurácia:  0.9882
```

Observe que 98,82% de acurácia, consideravelmente superior à 91% obtido no modelo linear, é ligeiramente acima de 97% obtido com o conjunto padrão. Destarte, é importante avaliar em cada caso se há interesse no aumento de menos de 1% na acurácia, a custo de horas a mais no tempo de processamento.

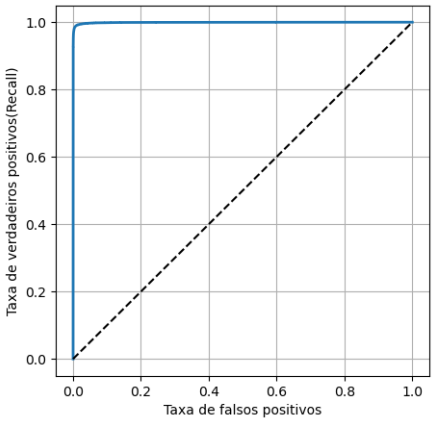
A fim de avaliar o modelo a partir da precisão, revocação e curva de característica de operação (ROC), adaptar o modelo para classificador binário. Ou seja, considere todos os rótulos na dicotomia `verdadeiro` para classes ímpares e `false` para outras classes. Lembre-se de que anteriormente importamos as bibliotecas, criamos etapas binárias e funções. Isso é essencial para correto funcionamento. Para plotar a precisão e revocação iremos chamar a função `plot_precision_recall`. Tenha paciência com o processamento nesta etapa pois há grande volume de dados, no meu caso, quase duas horas e meia. A notícia boa é que está é a última etapa demorada.

```
y_train_imp = [x % 2 == 1 for x in y_train]
svm_clf_imp = SVC(random_state=0)
svm_clf_imp.fit(X_train, y_train_imp)
y_scores = svm_clf_imp.decision_function(X_train)
train_pred_imp = (y_scores > 0)
r1 = [-5, 5, 0, 1.05]
r2 = [-0.2, 0.25, 0.99, 0.997]
plot_precision_recall(y_train_imp,y_scores,r1,r2)
```



Com o limiar em zero a precisão é aproximadamente $0,9944$ e revocação $0,9939$. Plotar a revocação em função de falsos positivos é fácil.

```
plot_roc_curve(y_train_imp, y_scores)
```



A linha contínua representa a curva ROC, a segmentada é referência. Em resumo, quanto mais próxima a curva é da borda superior esquerda, melhor o modelo. Uma forma de quantificar a curva ROC é calcular a área sob a curva. Um classificador perfeito tem área igual a 1, um classificador exclusivamente aleatório tem área igual a 0,5.

```
>>> roc_auc_score(y_train_imp, y_scores)
 0.9996410677747907
```

Essa é uma assertividade muito boa da aplicação proposta, a área abaixo do gráfico é muito próxima a 1. Caso queira, você pode construir a curva para o conjunto de teste.

7.4.4.4 Analisar o erro

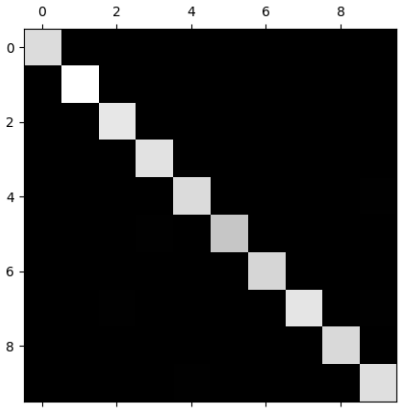
No item acima avaliamos o modelo na ótica binária, doravante analisaremos na perspectiva multiclasse. Construir a matriz de confusão no conjunto de treinamento é simples.

```
#Matriz de confusão

conf_mx = confusion_matrix(y_test, test_pred)
conf_mx
```

```
[[ 975,    0,    0,    0,    0,    1,    2,    1,    1,    0]
[   0, 1131,    2,    0,    0,    1,    0,    0,    1,    0]
[   1,    1, 1021,    0,    1,    0,    0,    4,    3,    1]
[   0,    0,    1, 1000,    0,    1,    0,    2,    3,    3]
[   0,    1,    1,    0,  969,    0,    3,    0,    0,    8]
[   2,    0,    0,    7,    1,  875,    3,    1,    0,    3]
[   4,    2,    0,    0,    1,    2,  948,    0,    1,    0]
[   0,    3,    5,    2,    0,    0,    0, 1013,    0,    5]
[   2,    0,    2,    1,    1,    2,    1,    2,  961,    2]
[   3,    4,    0,    2,    5,    2,    0,    3,    1,  989]]
```

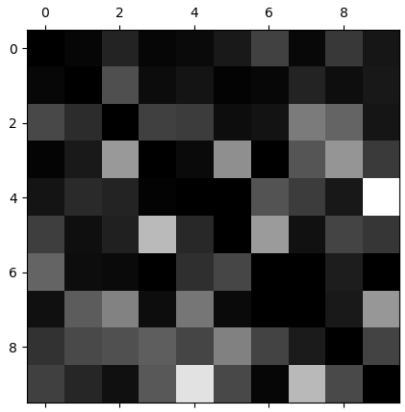
Observe que as linhas correspondem classes reais, as colunas classes previstas. Assim, a diagonal principal mostra que os acertos são maiores que os erros. O maior elemento fora da diagonal principal localiza-se em `5x10`. Isso significa que há bastante erro absoluto de predição quando trata-se do número 4 e o modelo acredita ser 9. Provavelmente existem muitas imagens dessa classe ou nosso modelo não funciona tão bem nessa classe. Não deixe de comparar com as matrizes anteriores. Talvez, visualizar a matriz de confusão em cores seja mais conveniente.

```
plt.matshow(conf_mx, cmap=plt.cm.gray)
plt.show()
```



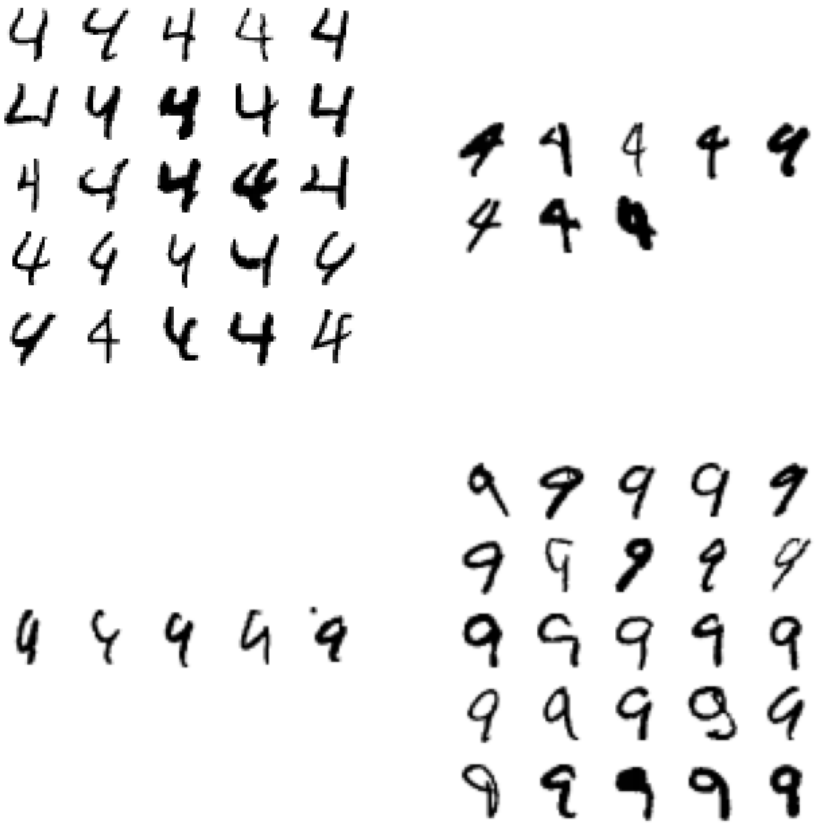
As linhas correspondem classes reais, as colunas classes previstas pelo modelo. Quanto mais claro a imagem na diagonal principal, mais acertos absolutos. Por fim, relativizar o erro torna-se indispensável para a comparação. Destarte, vamos dividir cada valor na matriz pelo número de imagens da classe afim.

```
row_sums = conf_mx.sum(axis=1, keepdims=True)
norm_conf_mx = conf_mx / row_sums
np.fill_diagonal(norm_conf_mx, 0)
plt.matshow(norm_conf_mx, cmap=plt.cm.gray)
plt.show()
```



Igualmente, as linhas correspondem classes reais e colunas as previstas pelo modelo. Cor clara indica maior erro de predição. Não obstante, o final da coluna da classe 4 e o final da linha da classe 4 são tendentes ao branco. Dessa forma, é imperioso notar que ocorre bastante equívoco quando envolvem as classes 4 e 9 na predição e no rótulo mutuamente. Para análise minuciosa, plotaremos algumas imagens dessas classes. Na etapa I, importamos as bibliotecas e definimos a função `show_digits`.

```
show_digits(4, 9)
```



Acima tem-se quatro blocos de imagens, em todos há somente algarismos quatro ou nove. O bloco superior esquerdo e inferior direito foram preditos corretamente pelo

modelo. O bloco superior direito foi erroneamente classificado pelo modelo como nove. O bloco inferior esquerdo foi erroneamente classificado pelo modelo como quatro. De fato, há alguma promiscuidade no manuscrito, onde até mesmo um humano teria dificuldade em reconhecer.

7.5 Etapa V: SVM no conjunto expandido com redução de dimensionalidade

7.5.1 Apresentação

Use o algoritmo de redução de dimensionalidade PCA nos dados importados do conjunto de dados MNIST expandido artificialmente da etapa anterior, certifique-se de preservar 95% de sua variância. Treine um modelo supervisionado de classificação não linear multiclasse (os rótulos devem ser os inteiros entre 0 e 9, inclusive) do SVM para reconhecer um número, de apenas um algarismo, escrito à mão. O programa deverá receber uma imagem e classificar qual número é representado na imagem. O cálculo do desempenho deve ser feito considerando o número de acertos e erros nos conjuntos. Para o gráfico da precisão, revocação e curva ROC, considere um modelo de classificação binária com os seguintes rótulos: número ímpar ou número não ímpar. A análise de erro pode ser realizada com a matriz de confusão para o conjunto de teste.

7.5.2 Duração

Sugere-se três tempos de, no mínimo, 45 minutos cada.

7.5.3 Objetivo

Apresentar aos alunos uma forma de uso de programação para o reconhecimento de número manuscrito. Será de suma importância durante a atividade o conhecimento em probabilidade de eventos equiprováveis, taxas de falsos positivos, de verdadeiros positivos, revocação, precisão, noções básicas de matriz e matriz de confusão.

7.5.4 Sugestão de resolução e considerações

Como informado anteriormente, as etapas I e III são essenciais por importar os dados, remover ruídos, expandir artificialmente e importar as bibliotecas utilizadas.

7.5.4.1 Redução de dimensionalidade

O objeto `pca` recebe a configuração da `PCA` ajustada para preservar 95% da variância. `X_train` recebe o conjunto de treinamento com redução de dimensionalidade. `X_test` garante a compactação também do conjunto de teste, afinal o modelo processará apenas instâncias compactadas.

```
from sklearn.decomposition import PCA

pca = PCA(n_components=0.95)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

É mister observar que doravante, os conjuntos de treinamento `X_train` e de teste `X_test` estão expandido artificialmente e compactados. Além disso, se usar o comando `.shape` verás que cada instância foi compactada para 160 dimensões, em vez das 784 características originais, uma diferença considerável. Resta descobrir acerca do desempenho!

7.5.4.2 Treinar o modelo

O conjunto de dados agora possui consideravelmente menos características e preserva boa parte da variância. Os dados foram exaustivamente trabalhados, destarte para ajusta-los ao modelo `svm_clf_reduced` use o comando `SVC`. Ao treinar o modelo, o Scikit-learn cuida de procurar os coeficientes que melhores se ajustam à forma genérica do SVM. No meu computador, demorou cerca de 20 minutos.

```
from sklearn.svm import SVC

svm_clf_reduced = SVC(random_state=0)
svm_clf_reduced.fit(X_train, y_train)
```

7.5.4.3 Espiar uma predição

Escolha uma instância no conjunto de teste e veja a predição que seu modelo fará. Observe se funcionará como esperado ou cometerá erro de predição. Abaixo escolhemos novamente a instância `n=1777`, ano de nascimento de Carl Friedrich Gauss. Inicialmente a imagem da instância, em seguida da predição. Para a visualização, é fundamental que `inverse_transform` garanta a descompactação da instância, nos termos do ajuste feito.

```
n = 1777

X_inverse = pca.inverse_transform([X_test[n]])

some_digit = X_inverse
some_digit_image = some_digit.reshape(28, 28)
plt.imshow(some_digit_image, cmap="binary")
plt.axis("off")
plt.show()

svm_clf_reduced.predict([X_test[n]])    #predição
```

O modelo acertou!!! Se houver dúvida, use o comando `print(y_test[n])` para saber qual é o algarismo representado. O modelo retornou um `array([0], dtype=uint8)`. Note que, apesar de ter sido compactado e descompactado, não há severa perda na qualidade da imagem. Provavelmente o erro de reconstrução é aceitável. O funcionamento do modelo OvR consiste em treinar 10 classificadores binários (zero contra não zero, um contra não um...) e obter o *score* de decisão para a imagem a fim de selecionar a classe vencedora como predição. Veja o *score* por classe:

```
>>> svm_clf_reduced.decision_function([X_test[n]])
 array([[9.3140252, -0.3017417, 5.2397645, 3.7390169, 0.7018689,
 7.2797272, 8.29281059, 1.7213442, 2.7295436, 5.9343933]]|
```

De fato, o *score* mais alto corresponde à classe correta. Quanto menor o *score*, menor a correlação com a classe. O classificador ficou com pequena dúvida em relação à classe 6. Por outra lado, está bastante confiante que não é classe 1, pois obteve *score* de aproximadamente -0,3.

7.5.4.4 Avaliar o modelo

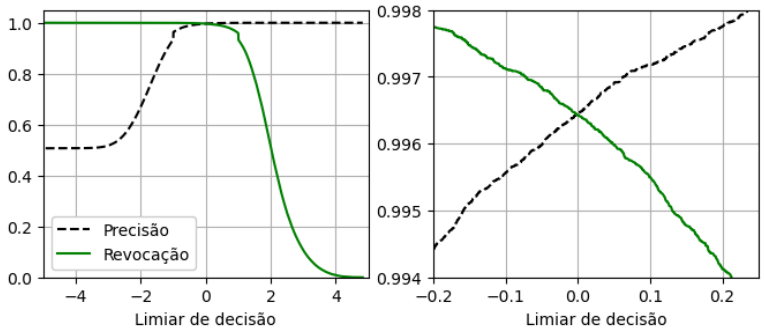
Como saber se o modelo está bem ajustado, sem sobreajuste ou subajuste? Uma maneira de avaliar é observar o percentual de predições erradas nos conjuntos. A etapa I define as funções, por isso é essencial que haja aquela etapa no código a fim de evitar erro. Destarte, resta apenas aplicar as funções no modelo.

```
>>> test_pred = test_pre(svm_clf_reduced)
>>> train_pred = train_pre(svm_clf_reduced)
>>> loss(svm_clf_reduced, test_pred, train_pred)
número de erros no conj de Treinam:  945
percentual:  0.0031717263253285004
número de erros no conj TESTE:  107
percentual:  0.0107
Acurácia:  0.9893
```

É incrível! Com muito menos tempo de processamento, conseguimos 98,93% de acurácia! Para o nosso caso, a redução foi duplamente eficaz, provavelmente por filtrar alguns ruídos e detalhes desnecessários. Compare o resultado com os anteriores.

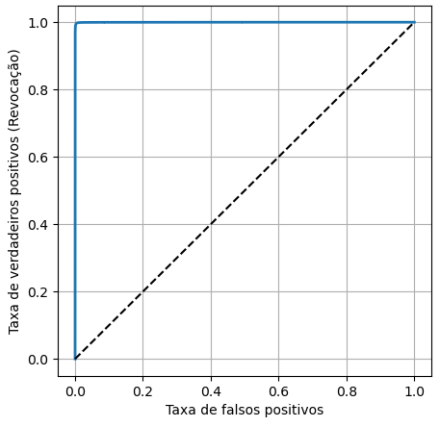
A fim de avaliar o modelo a partir da precisão, revocação e curva de característica de operação (ROC), vamos adaptar para classificador binário. Ou seja, considere todos os rótulos na dicotomia `verdadeiro` para classes ímpares e `false` para outras classes. Para plotar a precisão e revocação, basta chamar a função. No meu caso, o processamento levou 16 minutos.

```
y_train_imp = [x % 2 == 1 for x in y_train]

svm_clf_imp = SVC(random_state=0)
svm_clf_imp.fit(X_train, y_train_imp)
y_scores = svm_clf_imp.decision_function(X_train)
train_pred_imp = (y_scores > 0)

r1 = [-5, 5, 0, 1.05]
r2 = [-0.2, 0.25, 0.994, 0.998]
plot_precision_recall(y_train_imp,y_scores,r1,r2)
```



Com o limiar em zero a precisão é aproximadamente $0,9964$ e revocação $0,9964$. Plotar a revocação em função de falsos positivos é fácil.

```
plot_roc_curve(y_train_imp, y_scores)
```



A linha contínua representa a curva ROC, a segmentada é referência. Em resumo, quanto mais próxima a curva é da borda superior esquerda, melhor o modelo. Uma forma de quantificar a curva ROC é calcular a área sob a curva. Um classificador perfeito tem área igual a 1, um classificador exclusivamente aleatório tem área igual a 0,5.

```
>>> roc_auc_score(y_train_imp, y_scores)
 0.9998428313941465
```

Essa é a maior assertividade da aplicação proposta, a área abaixo do gráfico é muito próxima a 1. Você pode construir a curva, também para o conjunto de teste.

7.5.4.5 Analisar o erro

No item acima avaliamos o modelo na ótica binária, doravante analisaremos na perspectiva multiclasse. Construir a matriz de confusão no conjunto de treinamento é simples.

```
#Matriz de confusão

conf_mx = confusion_matrix(y_test, test_pred)
conf_mx
```

```
[[ 976,    0,    0,    0,    0,    1,    2,    1,    0,    0]
[   0, 1131,    2,    0,    0,    0,    0,    0,    1,    1]
[   1,    1, 1021,    0,    1,    0,    0,    4,    3,    1]
[   0,    0,    1, 1001,    0,    1,    0,    2,    3,    2]
[   0,    0,    1,    0,  972,    0,    2,    0,    0,    7]
[   2,    0,    0,    6,    1,  877,    2,    1,    0,    3]
[   3,    2,    0,    0,    1,    1,  950,    0,    1,    0]
[   0,    3,    4,    1,    1,    0,    0, 1014,    1,    4]
[   1,    0,    2,    1,    2,    2,    0,    2,  963,    1]
[   1,    3,    0,    2,    7,    3,    1,    3,    1,  988]]
```

Observe que as linhas correspondem classes reais, as colunas classes previstas. Assim, a diagonal principal mostra que os acertos são maiores que os erros. O maior elemento fora da diagonal principal localiza-se em `5x10`. Isso significa que há alguns erro absoluto de predição quando trata-se do número 4 e o modelo acredita ser 9. Provavelmente existem muitas imagens dessa classe ou nosso modelo não funciona tão bem nessa classe. Ainda sim, esse é um modelo promissor. Não deixe de comparar com as matrizes anteriores.

7.6 Etapa VI: Enviar imagem do seu manuscrito para o modelo

7.6.1 Apresentação

Interagir com o modelo usando sua própria escrita. Importar para a máquina uma imagem PNG com 28x28 pixels contendo um dígito escrito à mão. Tratar a imagem para que se assemelhe ao conjunto de treinamento, a fim de o modelo predizer corretamente. Realizar a predição da imagem. O cálculo do desempenho deve ser feito considerando o grau de semelhança com as classes. A predição da imagem pode ser feita usando os modelos das etapas IV e V.

7.6.2 Duração

Sugere-se dois tempos de, no mínimo, 45 minutos cada.

7.6.3 Objetivo

Apresentar aos alunos uma forma de uso de programação para a predição de número manuscrito pelo próprio aluno. Efetuar o protagonismo do aluno no projeto. Será de suma importância durante a atividade noções básicas de matriz.

7.6.4 Sugestão de resolução e considerações

Essa etapa do projeto é valiosa, pois concatena a teoria e traz sentido prático. Novamente, as etapas anteriores são essenciais por importar os dados, remover ruídos, expandir artificialmente e importar as bibliotecas utilizadas doravante. Não obstante, algumas considerações para a produção da imagem se faz necessária: procure um ambiente bem iluminado; use fundo branco; escreva com caneta preta de ponta grossa (preferencialmente caneta com a ponta *brush*). Se possível, use um equipamento eletrônico próprio que permite a escrita diretamente na tela.

7.6.4.1 Importar e tratar a imagem

De início, usar o comando `cv2.imread` para importar, aplicar tons de cinza e unificar camadas da imagem. Aqui, consideramos o nome do arquivo de imagem como `student.png`. Ademais, é necessário que a imagem esteja no mesmo diretório, caso contrário, acrescente o endereço completo do diretório no nome do arquivo (se for programar online, isso não se aplica!). É importante destacar que o código do retângulo abaixo não funciona no Google Colab, porque não foi projetado para a programação online. Caso esteja programando online com Google Colab, se faz necessário adequar o código, a fim do correto *upload* do arquivo. Para o Google Colab, substitua o código do retângulo pelo código das seis linhas a seguir. Além disso, armazene a imagem do seu manuscrito com nome de `student.png` na pasta raiz do seu Google Drive.

```
#FUNCIONA NO GOOGLE COLAB
import cv2
from google.colab import drive
drive.mount("/content/drive")
frame = "/content/drive/My Drive/student.png"
my_image = cv2.imread (frame,0)
```

```
import cv2
#NãO FUNCIONA NO GOOGLE COLAB
my_image = cv2.imread("student.png", 0)
```

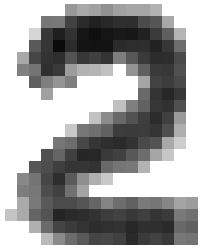
Note que a intensidade dos pixels da imagem está invertida. Abaixo, a primeira linha do comando faz a correção. Não obstante, usar o comando `reshape` para remodelar as características em apenas uma linha.

```
my_image = 255 - my_image
my_image = my_image.reshape(784)
```

7.6.4.2 Predizer e avaliar

Como exemplo, abaixo tem-se a imagem do manuscrito produzido pelo autor.

```
some_digit_image = my_image.reshape(28, 28)
plt.imshow(some_digit_image, cmap="binary")
plt.axis("off")
plt.show()
```



Para a predição, usar a sintaxe *"nome do modelo"*+`.predict`+*(["nome da instância"])* exibirá a predição. Por exemplo, usar `lin_clf.predict([my_image])` para o modelo linear predizer a instância `my_image`.

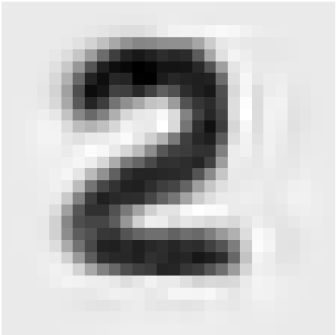
Usar o comando a seguir para predizer e avaliar segundo o modelo `svm_clf` do capítulo IV (ou II se pulou aquele). Na primeira linha, a predição segundo o modelo não linear, na segunda, a correlação do manuscrito com as classes do modelo, isto é, o *score* de decisão:

```
>>> print(svm_clf.predict([my_image]))
>>> svm_clf.decision_function([my_image])
 [2]
 array([[2.7368049, 5.754726, 9.3150874, 8.3085217, 0.6919741,
 4.778347, 3.8654007, -0.306809, 7.3091502, 1.6982652]])
```

Avaliar o manuscrito com o modelo produzido na etapa V exige a compactação da instância. Use o código abaixo para compactar, predizer e avaliar a correlação do

manuscrito com os *scores* de decisão. Por fim, tem-se a imagem do manuscrito com o erro de reconstrução.

```
my_image = pca.transform([my_image])
my_image = my_image[0]      #extrair my_image do conj criado
print(svm_clf_reduced.predict([my_image]))
print(svm_clf_reduced.decision_function([my_image]))

some_digit = pca.inverse_transform([my_image])

some_digit_image = some_digit.reshape(28, 28)
plt.imshow(some_digit_image, cmap="binary")
plt.axis("off")
plt.show()
```



```
[2]
[[3.72689705 2.71023004 9.31617513 7.30697049 -0.31119923
4.7126231  3.72142907 1.70623264 8.3177109  2.71172789]]
```

8 CONSIDERAÇÕES FINAIS

Diante de todo o exposto, o Aprendizado de Máquina, da teoria à implementação, como recurso didático e tecnológico, favorece a plena formação educacional. Seu uso estruturado nas aulas de matemática do Ensino Médio permite a abordagem de conceitos, categorias, métodos avaliativos, adversidades do aprendizado, além de táticas para resolução de problemas.

Destarte, depreende-se que a utilização do Aprendizado de Máquina, apresentado da teoria à implementação, auxilia na visualização da utilidade prática da matemática, agregando maior valor ao conhecimento. Isso é devido ao uso das habilidades – adquiridas no pensar do raciocínio lógico-matemático na busca pela resolução – de raciocinar, interpretar, reorganizar, argumentar e avaliar na implementação.

Insta salientar que o Aprendizado de Máquina encontrasse disseminado na cultura atual. O SVM é a entrada perfeita para compreender a temática, devido sua simples e clara organização estrutural. Por essa razão, esse algoritmo tem grande poder pedagógico.

A abordagem holística deste trabalho traz um produto final completo aos docentes, sendo um manual abrangente ou uma codificação jurídica, já que oferece legislações pertinentes, exercícios teóricos, tutorial e aplicações relevantes.

Por fim, buscou-se auxiliar os docentes nas estratégias pedagógicas que envolvam o Aprendizado de Máquina, contribuindo para as mudanças e formulações das novas diretrizes pedagógicas. Há a esperança de que o usufruto deste trabalho contribua para a melhoria da aprendizagem em matemática.

Referências

BANKO, M.; BRILL, E. *Scaling to Very Very Large Corpora for Natural Language Disambiguation*. Redmond, WA: Microsoft Research, 2001. <https://dl.acm.org/doi/pdf/10.3115/1073012.1073017>. [Acesso em: 13 jun. 2023]. Citado na página 32.

BINMORE, K. G. *The Foundations of Topological Analysis: A Straightforward Introduction*. Cambridge, UK: Cambridge University Press, 1980. ISBN 0-521-29930-6. Citado na página 35.

BRASIL. *Base Nacional Comum Curricular*. Brasília: Ministério da Educação, 2018. Citado na página 20.

BRASÍLIA. *Currículo em movimento do novo Ensino Médio*. Brasília: Secretaria de Estado de Educação do Distrito Federal, 2020. Citado na página 21.

BURKOV, A. *The Hundred-Page Machine Learning Book*. [S.l.]: Andriy Burkov, 2019. ISBN 978-1-9995795-0-0. Citado 6 vezes nas páginas 22, 24, 26, 34, 40 e 42.

DSTA. *DSTA Study Programme Chapter 21: Support Vector Machines*. Government of Singapore: Defence Science and Technology Agency Study Programme, 2023. <https://www.dcs.bbk.ac.uk/~ale/dsta+dsat/dsta+dsat-6/dsta-ZM-21-SVMs-excerpts-v2.pdf>. [Acesso em: 13 jun. 2023]. Citado na página 38.

FMI. *GDP per capita*. 2023. <https://www.imf.org/external/datamapper/NGDPDPC@WEO/OEMDC/ADVEC/WEOWORLD>. [Acesso em: 13 jun. 2023]. Citado na página 25.

GRUS, J. *Data Science from Scratch: First Principles with Python*. 2. ed. [S.l.]: O'Reilly Media, 2019. ISBN 978-14-920-4113-9. Citado 5 vezes nas páginas 56, 57, 58, 59 e 61.

GÉRON, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 2. ed. [S.l.]: O'Reilly Media, 2019. ISBN 978-1-492-03264-9. Citado 12 vezes nas páginas 26, 28, 30, 32, 34, 37, 39, 40, 41, 43, 44 e 45.

KINDERSLEY, D. *O livro da matemática*. Rio de Janeiro: Globo livros, 2020. ISBN 9786555670233. Citado na página 22.

MITCHELL, T. M. *Machine Learning*. New York: McGraw-Hill Education, 1997. ISBN 0070428077. Citado na página 23.

NORVIG, P. et al. *The Unreasonable Effectiveness of Data*. Washington, DC: iEEE intelligent Systems, 2009. <https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/35179.pdf>. [Acesso em: 13 jun. 2023]. Citado na página 31.

OCDE. *Better Life Index*. 2023. <https://stats.oecd.org/Index.aspx?DataSetCode=BLI>. [Acesso em: 13 jun. 2023]. Citado na página 25.

RAHMAN, W. *Inteligência Artificial e Aprendizado de Máquina*. [S.l.]: Editora Senac São Paulo, 2022. ISBN 978-85-396-3278-7. Citado na página 23.

RUSSELL, S.; NORVIG, P. *Inteligência Artificial*. Rio de Janeiro: Elsevier, 2013. ISBN 978-85-352-3701-6. Citado 4 vezes nas páginas 34, 41, 44 e 45.

SCHÖLKOPF, B.; SMOLA, A. J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. [S.l.]: The MIT Press, 2001. ISBN 0-262-19475-9. Citado 3 vezes nas páginas 38, 40 e 41.

TURING, A. M. *Computing Machinery and Intelligence*. Oxford: MIND, 1950. Citado na página 22.

VAPNIK, V. *Estimation of Dependences Based on Empirical Data*. Berlin: Springer-Verlag, 1982. Citado na página 34.

VAPNIK, V. *The Nature of Statistical Learning Theory*. 2. ed. [S.l.]: Springer, 1999. ISBN 0-387-98780-0. Citado na página 40.

VEISDAL, J. *Richard Feynman on Artificial General Intelligence*. Norway: Cantor's Paradise, 2019. <https://www.cantorsparadise.com/richard-feynman-on-artificial-general-intelligence-2c1b9d8aae31>. [Acesso em: 28 jun. 2023]. Citado na página 17.

WILMOTT, P. *Machine Learning: An Applied Mathematics Introduction*. [S.l.]: Panda Ohana Publishing, 2019. ISBN 978-1-9160816-0-4. Citado 6 vezes nas páginas 23, 40, 41, 43, 44 e 45.

WING, J. M. Computational thinking. *Communications of the ACM*, v. 49, p. 33 – 35, March 2006. Citado na página 31.