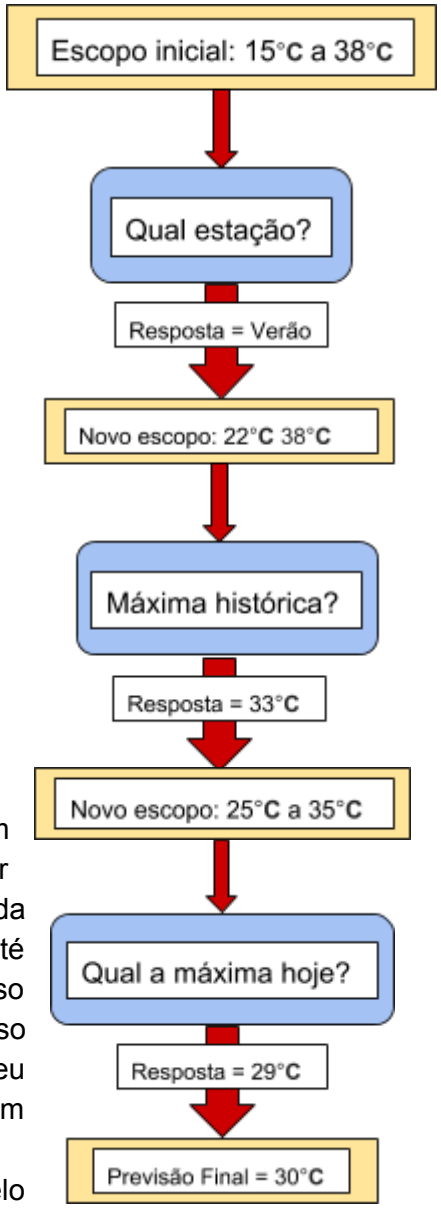
Relatório Projeto 3

Gabriel Zezze, Pedro Luiz da Costa, Rafael Libertini Argondizo

Em nosso projeto exploramos uma base de dados com mais de 50M de arquivos vetoriais representando desenhos em 350 diferentes categorias. Frente ao desafio de criar um classificador para reconhecer desenhos dentro dessas categorias, usamos alguns métodos de machine learning oferecidos pela biblioteca scikit-learn do python.

O primeiro método avaliado foi o Random Forest, um método baseado em árvores de decisão. Para entendermos como esse método funciona, precisamos entender o conceito de uma árvore de decisão.

Um bom exemplo para abstrair esse conceito conceito é a previsão da temperatura máxima de amanhã na cidade de São Paulo, dia 15 de Janeiro. Se não soubermos em qual mês estamos, é razoável definir um escopo de estimativas de 15°C a 38°C, sabendo que essas são as mínimas e máximas anuais. Podemos fazer alguma pergunta para melhorar essa estimativa, como por exemplo, “ Em que estação estamos? ”. Se estamos no verão, nosso escopo diminui para 22°C a 38°C. Perguntando, “Qual a temperatura máxima histórica nos dia 15 de Janeiro” (31°C) diminuímos para 27°C a 37°C. Estamos chegando perto, porém, para fazer uma melhor previsão, podemos nos perguntar “Qual foi a temperatura máxima hoje?”, sabendo que hoje a máxima foi de 29°C podemos pensar que o ano foi um pouco mais frio que a média histórica, finalizando nossa predição dizendo que amanhã a máxima será de 30°C.



Poderíamos seguir fazendo questões ad infinitum, porém, a partir desse ponto, o retorno só irá diminuir, portanto podemos seguir com nossa estimativa atual sem grandes perdas de confiança no resultado e sem desperdiçar esforço. Resumindo o exemplo, para chegar em uma predição fizemos uma série de perguntas, cada uma reduzindo o escopo de possíveis resultados, até chegarmos em um resultado que estávamos confiantes(isso tudo sem gastar esforço desnecessário). Esse processo ainda não é a árvore de decisão, porém, nos auxiliará no seu entendimento. Podemos resumir nossa predição em um gráfico.

Fizemos nossa predição com base em um modelo mental de pergunta-e-resposta, que nada mais é que uma árvore de decisões simplificada.

As duas principais diferenças entre nosso modelo e a arvore de decisoes sao, primeiramente, o fato de termos negligenciado a os ramos alternativos, ou seja, a predição

que teríamos chegado se a resposta às questões fosse diferente (ex: Se a resposta às estações fosse inverno.), e por último, uma característica muito diferente é que a arvore de decisoes só aceita perguntas de sim e não (True ou False), portanto para a pergunta da temperatura hoje, o método perguntara todas as possíveis temperaturas, e alguma delas terá resposta True, enquanto as outras tem False.
Portanto um diagrama de árvore de decisões bem simplificado ficará parecido com a figura abaixo:

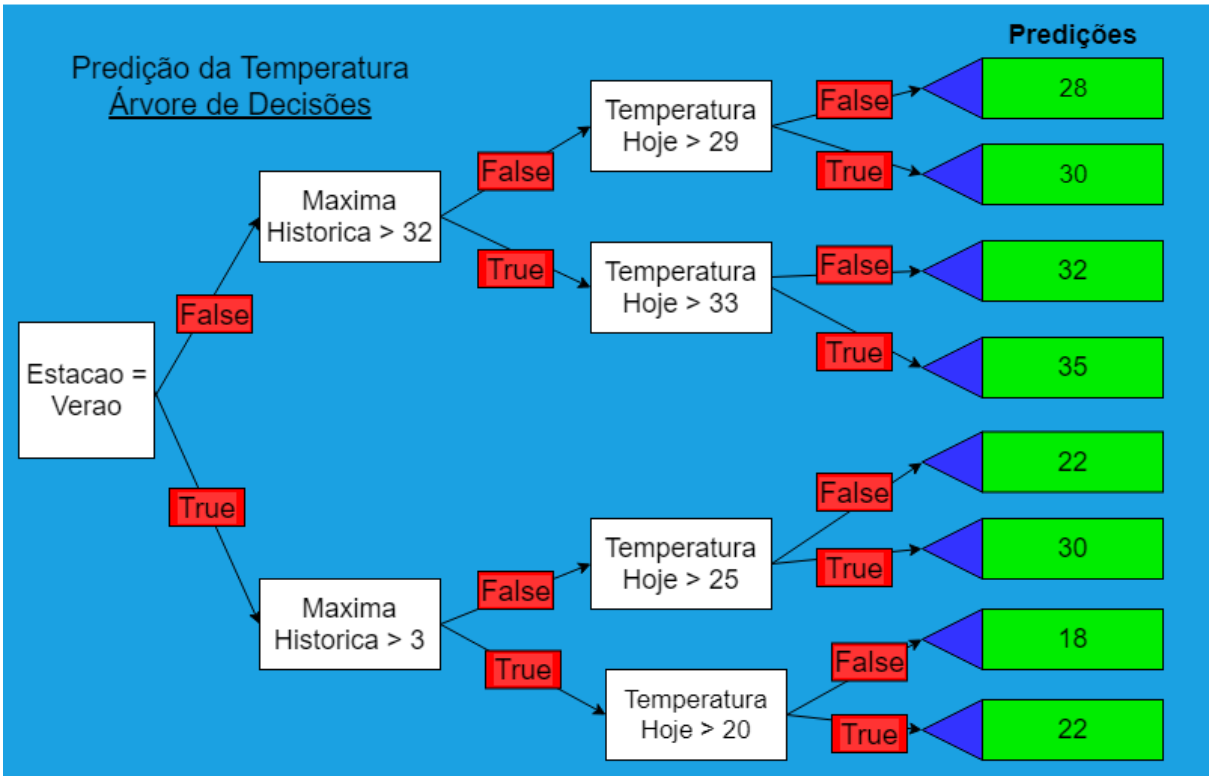


fig. 2 Arvore de Decisoes (incompleta)

Esse diagrama pode parecer muito diferente do nosso modelo mental, porém ele é apenas uma expansão dele. Nosso modelo está contido nessa árvore, basta tomar o caminho mais abaixo e seguir por nossas respostas e chegamos a mesma predição.

O modelo Random Forest, em sua essência, é uma árvore de decisões, que consegue decidir as “perguntas” que irá fazer, partindo dos dados que colocamos, e assim, montar uma árvore que consiga englobar o máximo de casos para fazer uma previsão coerente .

Apesar de simples nosso modelo já serve para fazer algumas predições, porém ele não tem nenhum conhecimento prévio do assunto, assim ele não sabe que se Inverno = Mais Frio, ele tem que aprender todas as relações com os dados do problema que jogamos nele. Chegamos agora a parte de Machine Learning, onde vamos treinar nosso Random Forest para que ele possa efetivamente fazer predições. Temos que então separar um pedaço de nossos dados, geralmente 80%, para alimentar o programa. Damos ao Machine Learning todos os dados relevantes ao problema, e dizemos qual o resultado desses dados.

Seguindo nosso exemplo da temperatura, precisamos de uma base de dados com informações relevantes para a previsão de tempo (conhecidas como “Features”) e dos valores da temperatura naqueles dias (valor que nós queremos prever “target”). Assim, ele “entenderá” que fatores levam a um específico resultado. Essa é chamada a fase de treino, que deve ser feita antes de qualquer previsão.

Com nosso Random Forest treinado, podemos agora usar o que sobrou dos nossos dados como base de Teste. Utilizando os mesmos tipos de variáveis que o modelo treinou, nós damos um cenário para o modelo e ele nos retorna um valor de temperatura, baseado no que ele sabe sobre dias com essas característica. Um exemplo muito simplificado é o de treinarmos com uma base que em janeiro a temperatura é sempre maior que 30 graus, portanto, se nós tentarmos prever a temperatura em um dia de janeiro, ele nos retorna uma temperatura maior que 30 graus.

Para nosso trabalho utilizamos a biblioteca Scikit Learn do Python, que conta com vários métodos de machine learning. Aplicamos nossas imagens como treino, e como target demos as suas categorias. O random forest nos retorna um modelo treinado, e quando colocamos uma foto nova ele nos retorna a probabilidade dela se encaixar em algumas das categorias de target.

Stochastic Gradient Descent

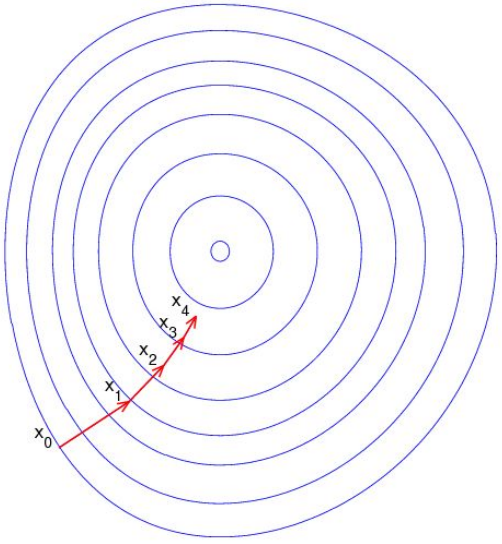
Outro método que testamos foi o SGD ("Stochastic gradient descent"), método muito comum no mundo do machine learning, por ser muito eficaz, relativamente simples de ser implementado.

O SGD é um algoritmo de otimização baseado em funções convexas; ele pode minimizar uma função para seu mínimo, a partir de mudanças em seus parâmetros, que o método aplica de maneira iterativa.

Para entendermos melhor o método precisamos definir um importante aspecto utilizado, o gradiente. Ele pode ser entendido como a taxa de variação de uma dada função, ou seja, quanto a saída dessa função mudará, dado um pequeno incremento em sua entrada.Ou seja, quanto maior o gradiente, maior a derivada (taxa de variação) e portanto, o modelo irá aprender mais rapidamente e se o gradiente for zero, o modelo não aprenderá mais. Uma definição matemática do gradiente é a *derivada parcial, com respeito a entrada da função.*

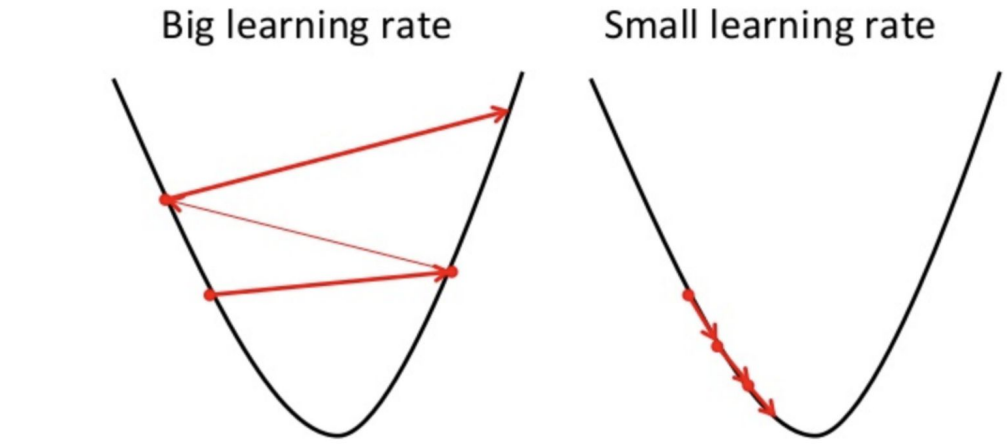
Podemos imaginar o SGD como alguém vendado tentando subir uma montanha com menor número de passos possíveis. No começo ele dará largos passos na parte mais inclinada da montanha, porém ao chegar perto do topo, ele diminui seus passos, para não arriscar passar o cume. Esse processo pode ser descrito a partir de um gradiente.

Se visualizarmos a figura abaixo, como uma vista superior da montanha, com as linhas representando uma mudança de altitude (curva de nível). As setas vermelhas representam os passos do nosso homem hipotético. A partir da imagem podemos abstrair o gradiente como um vetor, que indica a direção do passo na direção mais íngreme.

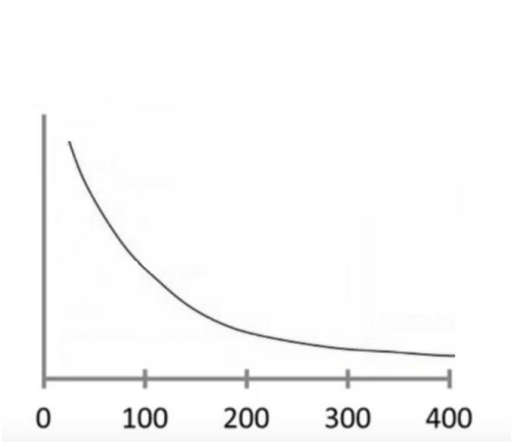


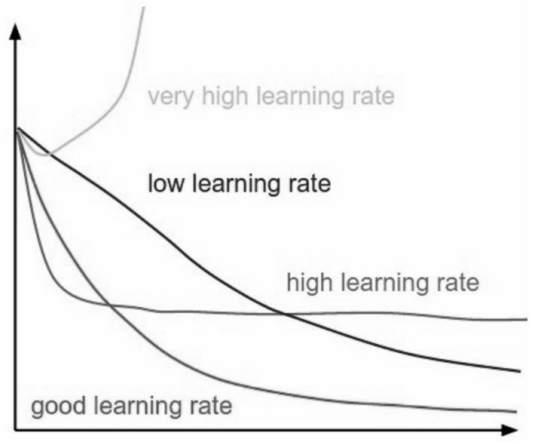
Podemos notar que o gradiente de X0 a X1 é muito mais longo que o de X3 a X4, isso ocorre pois a colina fica menos íngreme conforme nosso amigo sobe a montanha. Isso se relaciona com a taxa de variação diminuindo em um função convexa, conforme chegamos em seu máximo (“cume da montanha”), a taxa de variação diminui.

O SGD pode ser programado também para chegar no fundo de um vale, ou o mínimo de uma função. O que queremos , basicamente, é chegar lá o mais rápido possível, sem correr o risco de passar reto e ter que fazer meia volta. O caminho que tomamos até chegar ao nosso mínimo ou máximo é chamado de taxa de aprendizagem, e pode ser resumido em duas imagens:



Na imagem vemos os dois extremos da taxa de aprendizagem, uma taxa alta, nos dá um caminho errado, onde passamos muitas vezes do objetivo e temos que dar meia volta, pois tomamos passos muito grandes, a segunda nos mostra uma abordagem muito conservadora, onde, para não correr o risco de não errar, romanos passos muito pequenos, e acabamos demorando uma eternidade para chegar ao destino. Nosso trabalho é otimizar essa taxa de modo a chegar rápida e eficientemente a nosso mínimo. Se plantarmos diferentes taxas de aprendizado podemos padronizar o que seria uma boa ou ruim taxa:





A esquerda temos a taxa ideal, onde sempre diminuirmos nossa função até chegarmos no mínimo absoluto, e a esquerda alguns exemplos de taxas incorretas, onde a função sempre cresce ou estabiliza num valor maior que o mínimo real.

Entender melhor as técnicas de machine learning nos fez refletir sobre como melhorar a performance de nosso classificador, permitindo-nos obter um resultado mais próximo da realidade.