02

Boosting

Transcrição

[00:00] Nessa última aula nós aprendemos como nós combinamos essas árvores, combinando diferentes métodos e quais são os tipos de resultados que nós podemos obter com eles.

[00:16] No caso nós vimos AdaBoost, o random forest, que são algoritmos bem poderosos, principalmente o random forest, pra aplicações um pouco mais próximas da realidade. Mas eles levam em consideração a média entre as árvores e a maioria dos votos entre esse conjunto de árvores.

[00:30] E se nós, ao invés de considerarmos essa forma, construíssemos as árvores de uma forma um pouco diferente?

[00:37] Ao invés de nós pegarmos várias árvores aleatórias, nós construímos uma árvore de decisão super simples e com base na resposta que nós temos, nós vamos ter um erro, nós pegamos esse erro e construímos outra árvore em cima disso, meio que sequencialmente, depois pegamos outro erro, calculamos o erro em cima disso e depois construímos mais uma por cima da árvore e assim por diante, até nós termos uma quantidade aceitável ou chegarmos até um determinado limite.

[01:06] Esse processo mais sequencial, que leva em consideração esse erro residual, que é focada mais pra você reduzir o viés da árvore, é o que nós podemos chamar de boosting, são os algoritmos de boosting.

[01:20] Basicamente, quando você trabalha com esses métodos de ensemble, você tem dois tipos, o método de bagging – ensemble eu digo de juntar, combinar vários métodos ou várias árvores, no caso que nós estamos trabalhando aqui – você tem o caso de bagging ou você tem o caso de boosting, e são algoritmos diferentes. O caso de boosting é um pouco mais complexo, mas o ponto que eu quero dizer é que ele funciona tanto quanto ou só uma alternativa justamente para o caso de bagging.

[01:48] Então você tem o random forest pra um lado e você tem o AdaBoost, que é adaptative boosting ou gradient boosting. A diferença entre os algoritmos é a forma que você leva em consideração esses erros pra ir construindo a sua árvore, ir expandindo a sua árvore de forma sequencial.

[02:05] Eu não vou entrar nos detalhes, nem pro caso de regressão nem pro caso de classificação, porque esses métodos são de fato mais complexos de se entender. O legal é que, novamente, o sklearn provê toda essa metodologia pra nós já implementada e nós vamos ver como é fácil implementar e ver que eles funcionam, e funcionam bem.

[02:26] Basicamente, de forma geral, eles vão pegar uma árvore que tem nossos dados, eles vão construir uma árvore pequena e nós reduzimos essa árvore, então ela está numa forma simples. Nós pegamos os erros e vamos repetindo o processo, construímos uma árvore sequencialmente, depois olhamos de novo o erro que nós tivemos, com certeza o erro mudou com relação ao nosso erro inicial, e esse processo é construído de uma forma um pouco mais interativa. A ideia geral dos algoritmos de boosting é essa. Vamos ver como que nós fazemos?

[03:05] No caso, eu citei dois, existem mais, mas os dois que eu vou trabalhar nessa aula são o adaptative boosting e o gradient boosting. Nós vamos ver que eles funcionam tanto pra classificação quanto pra regressão. Eu vou já importar os dois, então nós vamos ter novamente aqui dentro do caso do ensemble, nós vamos ter "AdaBoostRegressor" e como eles são parecidos eu já vou fazer aqui do "Classifier", já vou importar e vamos ver o que acontece.

[03:47] Os nossos dados são praticamente os mesmos, então estou aqui novamente só recriando nossos dados de treinamento, até aqui. Lembrando que pro nosso caso da regressão deu mais ou menos 81%, então agora nós só

precisamos fazer o modelo, ele recebe “AdaBoostRegressor”. “modelo.fit(treino, treino_labels)”. Mesmo problema que nós já tivemos, então passando aqui, limpei os dados, “.ravel”. Criamos o nosso regressor.

[04:38] Ele pode ser usado em combinação com outros métodos, exatamente como nós vimos com outros casos de ensemble, caso base são as árvores. E no caso aqui, o estimador base é o que define justamente pra eu ir crescendo esse modelo, podem ser usados outros se eu não passo nenhum é usado como base uma árvore de decisão.

[05:00] Agora vamos ver como que ele se portou? A ideia geral, na verdade, desse tipo de modelo é que ao invés dele tentar overfitar os dados, ou seja, construir uma árvore super grande, ele vai fazendo um processo muito mais gradual, ele vai olhando os erros, vai corrigindo. E a ideia é que eu aprenda o formato geral, genérico da minha árvore de uma forma muito mais lenta, digamos assim.

[05:27] Agora novamente aqui, modelo.predição, 80% pra treino 77%, 78% pra teste também funcionou super bem. Não tão bem, novamente, perde ainda para a nossa regressão linear, mas também já funcionou superbem.

[05:42] No caso, vamos já fazer pra mostrar como é fácil, vamos fazer também do gradient boosting já. Nós temos o “sklearn.ensemble import GradientBoostingRegressor”. Criamos o método. Subimos, então aqui é basicamente “GradientBoostingRegressor”, ele não está definido porque é boosting e não boost. Criamos o modelo e agora o resto todo nós já estamos mais do que careca de saber. Criamos o modelo.

[06:29] Tem aqui o critério, taxa de aprendizado pra eu ir crescendo. Novamente, se nada é passado eu uso uma árvore como base. Agora é só fazer o score, no caso de treino deu 84% ali e no caso de teste, olha só, 81, 82%. Valendo só a curiosidade, vamos ver como o nosso modelinho de regressão linear se portou? Só pra nós relembrarmos? Eu estou copiando aqui, estou colando, 82, ou seja, por meio ponto não é melhor, mas quase a mesma coisa.

[07:14] É um modelo diferente, foi construído de forma diferente, com aprendizado um pouco diferente e olha só, o último modelo que nós vimos funcionou praticamente a mesma coisa, indo melhor, inclusive, nos dados de treinamento.

[07:28] Agora, da mesma forma como nós vimos para os dados de regressão, vamos ver como funciona pra classificação? Vamos primeiro no caso aqui, vamos copiar os nossos dados porque nós vamos fazer o mesmo processo. Vou até já fazendo a regressão logística aqui pra nós darmos uma olhada, vou até copiar tudo aqui, na verdade.

[07:51] Novamente, não estou fazendo tudo porque nós já fizemos isso e já estamos mais do que carecas de saber que eu copiei isso de outras aulas. Então aqui o paste, eu basicamente copiei todo o processo de treino e de classificação, a acurácia aqui é do nosso modelo da árvore de decisão e eu quero parar aqui, pra nós lembrarmos quanto que foi no caso da previsões e aqui a nossa querida acurácia, no caso da regressão linear. Acurácia, que foi o 75%.

[08:41] O que eu quero fazer agora? Vamos fazer a mesma coisa, “from sklearn.ensemble import AdaBoostClassifier” e já vou importar do gradient boosting também. Eu vou só mandar pra cima porque é uma coisa super simples. Nós vimos lá que é 75%, nós vamos ter agora modelo, ele recebe “AdaBoostClassifier”. Ada boost classifier não está definido, eu acredito que eu tenha escrito errado, vamos copiar. É porque tem dois S aqui. Modelo.fit que nós também já estamos acostumados, “modelo.fit(treino, treino_labels)” criamos aqui.

[09:46] Novamente, o que muda é o algoritmo de visão da árvore do classificador pra regressão, fizemos a nossa previsão, vai ser “modelo.predict”. E fizemos a nossa acurácia. A acurácia pra esse cara deu 74. Foi melhor do que a nossa regressão linear que nós vimos.

[10:19] Dando uma relembrada ali, quanto que deu pra nossa decision tree? Nós vamos ter o sklearn. Nós vamos ter modelo, ele recebe “LogisticRegressor”. Logistic regressor is not defined.

[10:46] Nós podemos só importar aqui, from linear model, eu estou só copiando pra ter certeza que existe. Novamente nós vamos ter o "treino.head". Existe? Não, porque é um numpy array. Então nós queremos o "treino_labels", é 1, 0, 1, 0, quer dizer se gostou ou não gostou. Nós estamos até treinando de novo pra termos certeza do que acabou de acontecer.

[11:23] Então nesse cara, vou até copiar tudo de novo, pra nós termos certeza do que está acontecendo, novamente, prova real. Estou vindo aqui, "Command + V", a acurácia desse cara deu 78%. Agora porque aqui nós fizemos outra divisão de treino teste, então se eu copiar de novo, nós vamos ter outra acurácia, porque nós estamos sempre tendo a divisão. Só pra garantir que nós estamos fazendo com os dados bonitinhos, felizes e contentes. 76%, pra aquele caso nós vimos, deu 74%.

[12:00] Pra esse conjunto de dados nós queremos ver quanto que vai dar, eu vou revisar exatamente outros classificadores que nós vimos. Nós vamos ter o "sklearn.ensemble import RandomForestClassifier". O modelo recebe "RandomForestClassifier", "modelo.fit(treino, treino_labels)". Criamos nosso modelo.

[12:36] Agora nós já estamos acostumados com o resto, então aqui vai ser "modelo.predict(teste)" e a acurácia vai ser exatamente essa acurácia que nós vimos. Aqui é tudo igual. Pra acurácia pro caso da random forest foi um pouco pior.

[12:51] Agora nós vamos querer, o modelo vai receber ada boost classifier, depois o gradient boost. "AdaBoostClassifier", "modelo.fit(treino, treino_labels)" e criamos aqui. Agora novamente, fazendo as previsões em cima do modelo e novamente a acurácia em cima das nossas previsões. Acurácia, 74%. Foi um pouco pior, mas novamente, foi muito perto do 76 da nossa regressão.

[13:25] Dependendo da nossa distribuição, no caso de treino e teste, poderia dar outro resultado. Mas novamente, muito próximo, é exatamente isso que nós queremos focar, os dados são próximos, são parecidos, então o modelo não foi necessariamente pior.

[13:36] E aqui nós vamos importar o último caso, que é "from sklearn.ensemble import GradienteBoostingClassifier", porque nós queremos fazer o classificador. Modelo vai receber o "GradientBoostingClassifier", estou apertando pra cima, "modelo.fit(treino, treino_labels)".

[14:11] Criamos novamente critério próprio. A previsão, como nós estamos acostumados. A acurácia, como nós estamos acostumados. A acurácia desse cara, 74%, também foi muito boa. Novamente parecido. Pra esse conjunto de treino e teste os dados foram parecidos.

[14:28] E a grande lição que nós podemos ter nesses contos de árvores é porque as árvores são construídas de formas diferentes. No caso do random forest e do bagging, eles usam princípios de bagging, e no caso do gradient boosting e do adaptative boosting, o AdaBoost, eles consideram o boosting, ou esse processo de boosting, e vão gerar em árvores criadas sobre estratégias diferentes.

[14:54] E dependendo do seu conjunto de dado, uma pode se sobressair a outra, nós vimos comparações aqui, como elas foram parecidas, em alguns casos umas foram melhores do que as outras.

[15:03] Pra esse conjunto final de testes que eu fiz, o modelo melhor foi o linear, mas não necessariamente isso pode acontecer e pra um caso mais real, pra um caso de conjunto de dados mais reais, as árvores tendem a funcionar melhor. Árvores, eu falo, essas combinações de métodos.

[15:18] E o legal é justamente essa infinidade de métodos, que cada um, dependendo da forma como nós usarmos, nós podemos encontrar uma variação melhor em cima dos nossos dados.