EST171 - Aprendizado de Máquina
Departamento de Estatística
Universidade Federal de Minas Gerais

Lista 3

Henrique Aparecido Laureano Magno Tairone de Freitas Severino

Outubro de 2016

Sumário

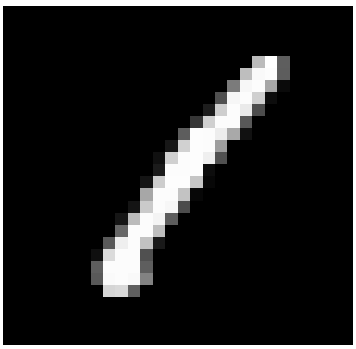
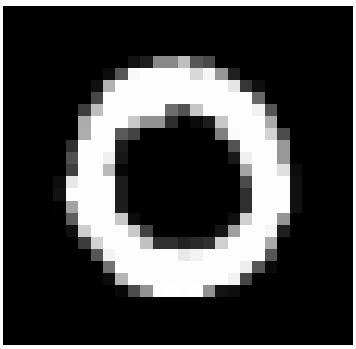
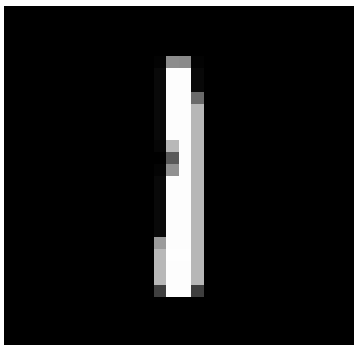
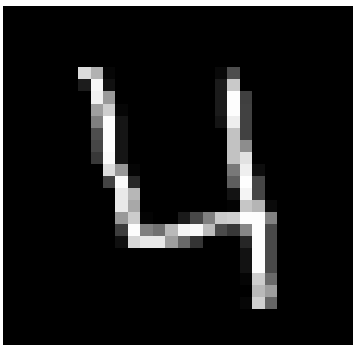
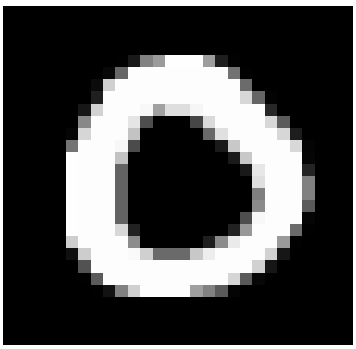
Exercício I

Seu objetivo é criar classificadores para predizer o dígito correspondente a uma imagem. O conjunto de dados está disponível em https://www.kaggle.com/c/digit-recognizer. Você deve usar o arquivo train.csv para criar os seus classificadores (incluindo validação), e deve fornecer ao site as predições encontradas para o conjunto test.csv. Note que os dígitos correspondentes no conjunto teste não estão indicados! O site irá rankear seu grupo de acordo com as predições fornecidas. Como parte do exercício, você deverá:

Inscrever seu time Kaggle. Qual o nome dele?

BetterThanYours

Plotar 5 imagens deste banco.

Você também deve implementar os seguintes classificadores, assim como estimar seus riscos via conjunto de teste (usando o conjunto de validação para estimar seus erros), mostrar o resultado de cada um.

Antes de implementar os classificadores, dividimos o banco de imagens em dois: 80% do banco para treino (33600 imagens) e 20% para validação (8400 imagens). O banco de treino foi utilizado para configurar o classificador e o de validação para escolha dos parâmetros de tunning. Para comparar os modelos, usamos o banco de testes disponível no site da competição. Vale ressaltar que em todos os modelos aqui ajustado, foram consideradas todas as 784 covariaveis, correspondentes à cada um dos pixels das imagens consideradas.

Bagging[1]

Bagging (**B**ootstrap **agg**rega**ting**) foi proposto por Leo Breiman em 1994 para melhorar a classificação pela combinação das classificações de conjuntos de dados de treino gerados aleatoriamente.

Dado um conjunto de treino padrão D de tamanho n, bagging gera m novos conjuntos de treino D_i, cada um de tamanho n', reamostrando de D uniformemente e com reposição. Pela amostragem com reposição algumas observações podem repetir em cada D_i. Se $n' = n$, para um grande n é esperado que o conjunto D_i tenha uma fração (1 - 1/e) ($\approx 63.2\%$) de valores únicos de D, com o resto sendo duplicações. Este tipo de amostra é chamada de amostra bootstrap. Os m modelos são ajustados usando as m amostras bootstrap e combinados pela média da resposta (para regressão) ou de votos (para classificação).

Bagging fornece uma melhora em procedimentos instáveis, como por exemplo, redes neurais artificiais e árvores de regressão e de classificação, na contramão, pode diminuir moderadamente a performance de métodos estáveis como o KNN.

A partir deste modelo, obtemos um classificador com acurácia de 91.66% no banco de validação.

Árvore de Classificação[2]

Florestas aleatórias é uma abordagem ensemble que pode ser pensada como uma forma de preditor de vizinhos mais próximos.

Ensemble é uma abordagem de "dividir para conquistar" utilizada para aumento de performance. O principal princípio dos métodos de ensemble é que métodos "fracos em aprendizagem" podem ser reunidos para formar um método "forte em aprendizagem".

[1] Breiman, Leo (1996). "Bagging predictors". *Machine Learning.* **24** (2): 123-140. https://dx.doi.org/10.1007%2FBF00058655.

[2] https://citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics/

Uma floresta aleatória começa com uma árvore de decisão que, em termos de ensemble, corresponde ao "fraco em aprendizagem". Numa árvore de decisão uma entrada é inserida no topo e conforme ela atravessa a árvore os dados são afunilados em conjuntos menores e menores. As florestas aleatórias combinam árvores com a percepção de um ensemble.

Para um dado número T de árvores, N amostras aleatórias com reposição são geradas para criar um subconjunto de dados. Esse subconjunto deve ter em torno de 66% dos dados totais. Para cada nó m variáveis explicativas são selecionadas ao acaso entre todas as variáveis disponíveis. As variáveis que fornecem a melhor classificação, de que acordo com alguma função objetivo, são utilizadas para fazer a classificação binária no nó. No nó seguinte outras m variáveis são selecionadas ao acaso e o mesmo é feito.

A partir deste modelo, obtemos um classificador com acurácia de 65.19% no banco de validação.

Boosting[3]

Boosting é um algoritmo que primariamente reduz o viés e a variância, e que também se trata de uma abordagem ensemble.

Boosting é fundamentado na observação de que encontrar vários métodos "fracos em aprendizagem" é muito mais fácil do que encontrar um único método de predição altamente preciso. Para aplicar a abordagem boosting nós começamos com um método (algoritmo) para encontrar regras fracas em aprendizado. O algoritmo boosting chama esse método "fraco" repetidamente, cada vez em um diferente subconjunto dos exemplos de treino (diferentes distribuições ou pesos). A cada chamada o algoritmo "fraco" gera uma nova regra fraca de predição que após várias chamadas são combinadas em uma única regra de predição que, se espera, ser muito mais acurada que qualquer uma das regras "fracas".

A partir deste modelo, obtemos um classificador com acurácia de 70.4% no banco de validação.

SVM

SVM (**S**upport **V**ector **M**achine) é uma representação das observações como pontos no espaço, mapeados de maneira que os exemplos de cada categoria sejam divididos por um espaço claro que seja tão amplo quanto possível. As novas observações são então mapeadas no mesmo espaço e preditos como pertencentes a uma categoria fundamentados em qual lado do espaço elas são alocados.

[3] https://www.cs.princeton.edu/picasso/mats/schapire02boosting_schapire.pdf

Em outras palavras, o que um SVM faz é encontrar uma linha de separação, mais comumente chamada de hiperplano entre dados de classes. Essa linha busca maximizar a distância entre os pontos mais próximos em relação a cada uma das classes.

O SVM é um classificador criado para fornecer separação linear.

A partir deste modelo, obtemos um classificador com acurácia de 89.13% no banco de validação, considerando o valor do parâmetro de custo igual a 1.

Na Tabela 1 são apresentados os tempos de processamento e execução dos classificadores mencionados acima, tendo o banco de testes como base para configurar os modelos. Nessa mesma tabela estão os respectivos riscos de cada classificador considerando o banco de testes. Estes riscos foram obtidos ao submeter o resultado da classificação de cada método para o site da competição.

Os classificadores foram processados numa máquina ubuntu com 8bg de memória ram. Entre os quatro classificadores propostos inicialmente o mais rápido foi a Árvore de Classificação e o mais lento foi o Boosting. Em relação ao risco, o melhor classificador foi o Bagging. Contudo, também foi utilizado o algoritmo KNN (**K** **N**earest **N**eighbor) considerando apenas um vizinho, $k = 1$. Com ele foi obtido o menor risco.

A partir deste modelo, KNN, obtemos um classificador com acurácia de 96.62% no banco de validação.

Tabela 1: Classificadores, tempos de processamento e riscos

Classificador	Tempo de processamento	Risco
Bagging	1.230357 horas	0.055
Árvore de Classificação	2.309469 minutos	0.375
Boosting	1.408647 horas	0.206
SVM	20.76242 minutos	0.099
KNN	12.86209 minutos	0.028

Na Figura 1 temos a árvore de classificação obtida.

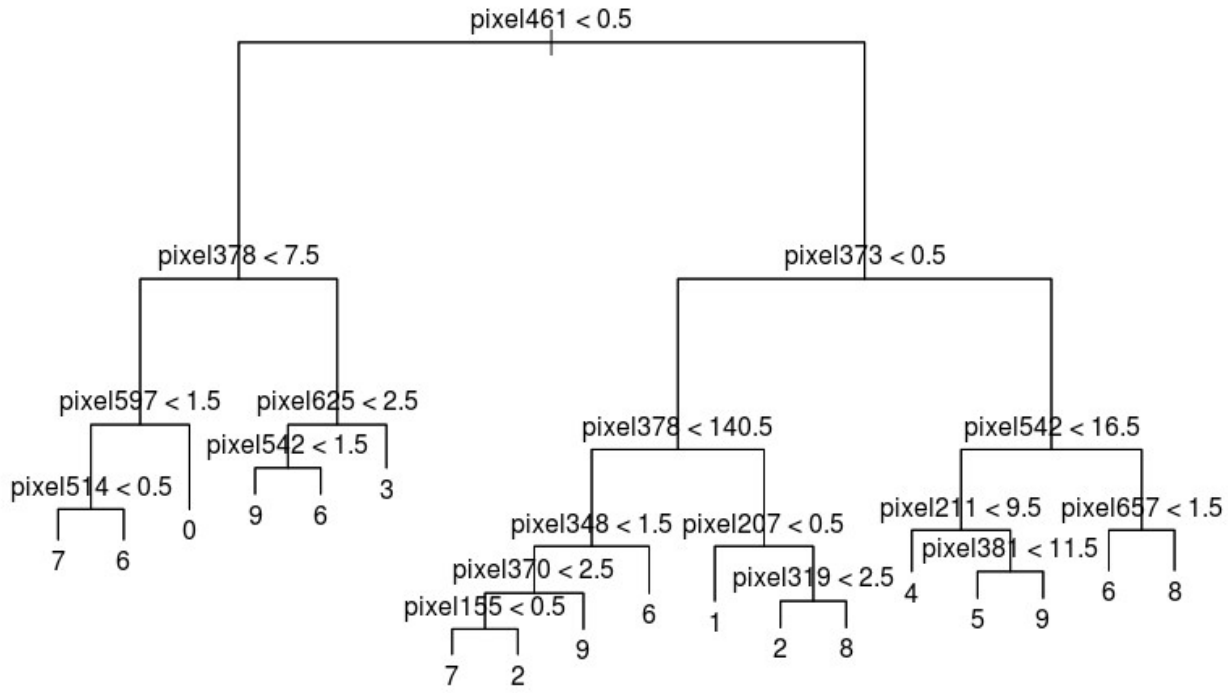


Figura 1: Árvore de Classificação obtida

Escolha um dos classificadores ajustados (o que achar melhor) e submeta suas respostas ao site

---

O classificador ajustado que obteve a menor porcentagem de erro foi o KNN.

Códigos: https://mynameislaure.github.io/ml-ufmg-list_3/code.R
