Ensembles Learning ou Aprendizagem em Conjunto

Cristiane Neri Nobre

Fonte: jixta.files.wordpress.com

Ensembles

- Métodos que geram muitos classificadores e combinam os seus resultados

  "Consultar vários especialistas para tomada de decisão"

- É amplamente aceito que o desempenho de um conjunto de muitos classificadores fracos é geralmente melhor do que um único classificador, dada a mesma quantidade de informação de treinamento

Ensembles

`sklearn.ensemble`: Ensemble Methods

The `sklearn.ensemble` module includes ensemble-based methods for classification, regression and anomaly detection.

User guide: See the Ensemble methods section for further details.

`ensemble.AdaBoostClassifier`([...])	An AdaBoost classifier.
`ensemble.AdaBoostRegressor`([base_estimator, ...])	An AdaBoost regressor.
`ensemble.BaggingClassifier`([base_estimator, ...])	A Bagging classifier.
`ensemble.BaggingRegressor`([base_estimator, ...])	A Bagging regressor.
`ensemble.ExtraTreesClassifier`([...])	An extra-trees classifier.
`ensemble.ExtraTreesRegressor`([n_estimators, ...])	An extra-trees regressor.
`ensemble.GradientBoostingClassifier`(*[, ...])	Gradient Boosting for classification.
`ensemble.GradientBoostingRegressor`(*[, ...])	Gradient Boosting for regression.
`ensemble.IsolationForest`(*[, n_estimators, ...])	Isolation Forest Algorithm.
`ensemble.RandomForestClassifier`([...])	A random forest classifier.
`ensemble.RandomForestRegressor`([...])	A random forest regressor.
`ensemble.RandomTreesEmbedding`([...])	An ensemble of totally random trees.
`ensemble.StackingClassifier`(estimators[, ...])	Stack of estimators with a final classifier.
`ensemble.StackingRegressor`(estimators[, ...])	Stack of estimators with a final regressor.
`ensemble.VotingClassifier`(estimators, *[, ...])	Soft Voting/Majority Rule classifier for unfitted estimators.
`ensemble.VotingRegressor`(estimators, *[, ...])	Prediction voting regressor for unfitted estimators.
`ensemble.HistGradientBoostingRegressor`([...])	Histogram-based Gradient Boosting Regression Tree.
`ensemble.HistGradientBoostingClassifier`([...])	Histogram-based Gradient Boosting Classification Tree.

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble

Ensembles

Condições necessárias para um bom desempenho

- **Diversidade**
  - Classificadores base devem ser independentes
    - Ideal: cometer erros diferentes
- **Acurácia**
  - Desempenho dos classificadores base deve ser melhor que classificação aleatória

Ensembles

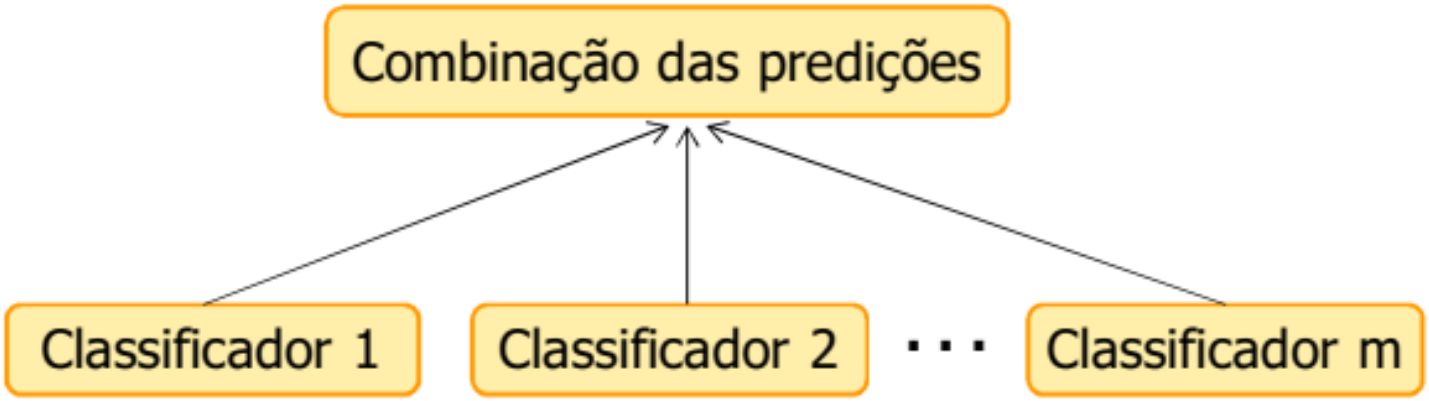
- Sejam 3 classificadores induzidos para os mesmos dados, com acurácia 0.6

- Se eles cometem os mesmos erros
  - Acurácia do ensemble será 0.6

- Se eles são completamente independentes
  - Ensembles erra classificação apenas se pelo menos 2 classificadores erram na predição

Ensembles

- Combinação de predições
  - Voto (média)
  - Voto (média) ponderado



Bagging

- Cada classificador é induzido por uma amostra diferente do conjunto de treinamento
  - Mesmo tamanho do conjunto original
  - Usa *bootstrap*
- Classe definida por votação
- Tende a reduzir variância associada com classificadores base

Técnica de amostragem Bootstrap

- No método **booststrap**, *r* subconjuntos de treinamento são gerados a partir do conjunto de exemplos original. Os exemplos são **amostrados aleatoriamente** desse conjunto, **com reposição**. Logo um exemplo pode estar presente em um determinado subconjunto de treinamento mais de uma vez.

- Os **subconjuntos de testes** são formados pelas sequências que **não estão no conjunto de treino**

  Cerca de um terço das instâncias são deixados de fora da amostra de bootstrap e não são usados na construção da k-árvore.

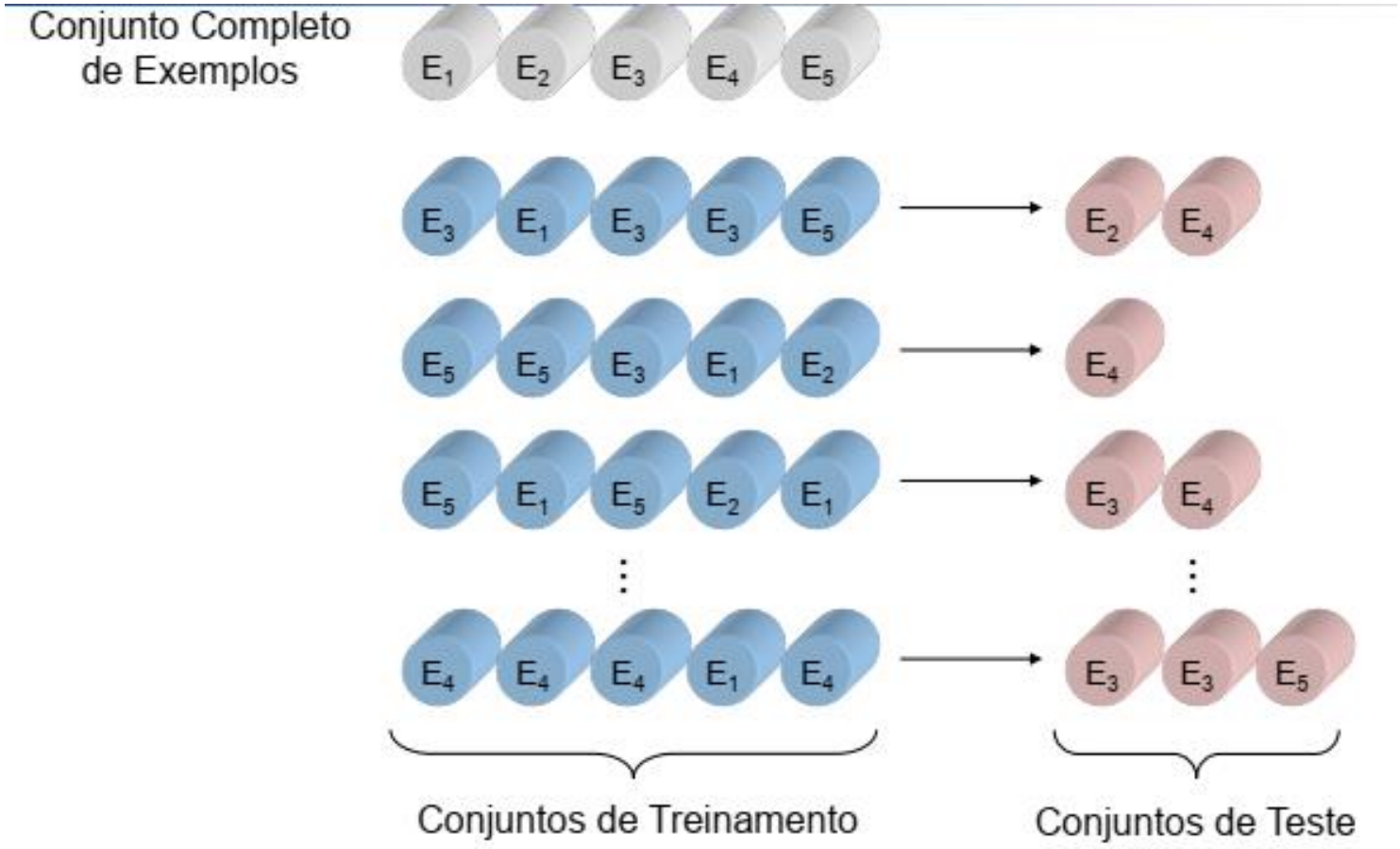
Técnica de amostragem Bootstrap

Normalmente adota-se $r \geq 100$. A ideia básica é repetir o experimento um número alto de vezes e estimar o desempenho nesses experimentos replicados.

Por este motivo, o bootstrap é um procedimento custoso.

Há vários estimadores bootstrap, e o mais comum é o e_0. Neste cada conjunto de treinamento tem *n* exemplos, amostrados com reposição do conjunto original, sendo *n* o número total de exemplos nesse conjunto

Técnica de amostragem Bootstrap



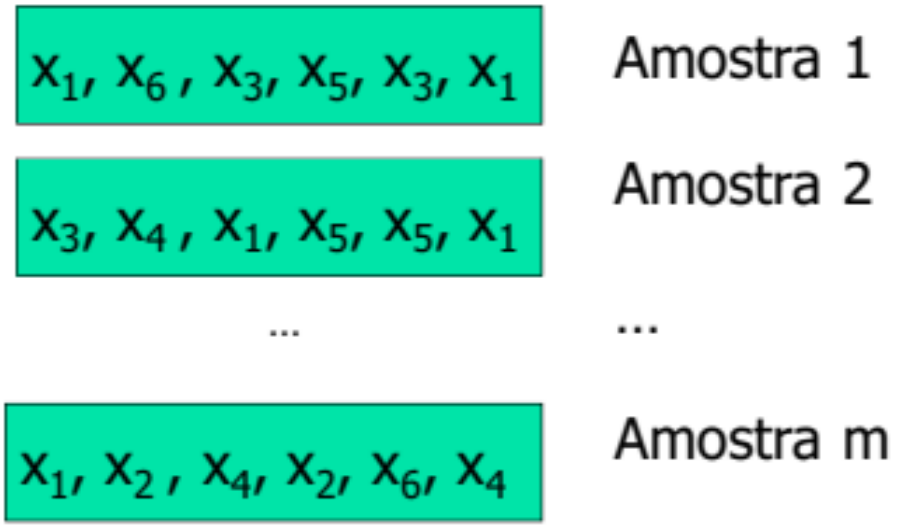
Técnica de amostragem Bootstrap

Como se avalia o resultado final?

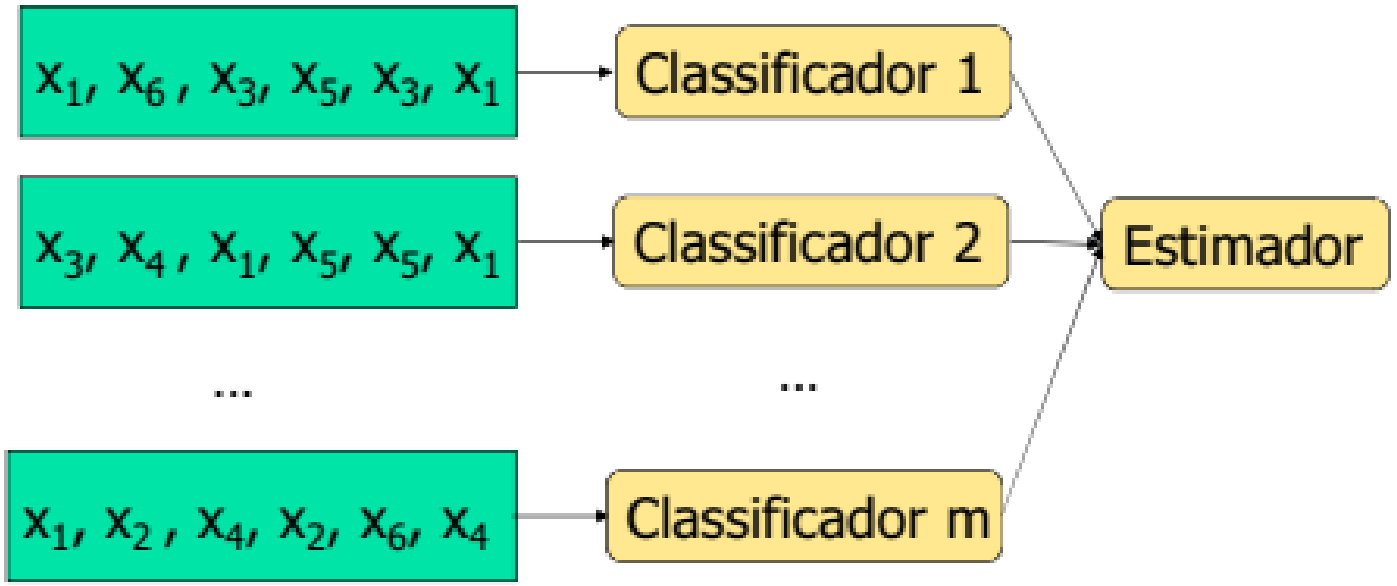
- O resultado final é dado pela média do desempenho observado em cada subconjunto de teste.

Bagging

- Seja o conjunto de dados de treinamento $\{x_1, x_2, x_3, x_4, x_5, x_6\}$



Bagging


$x_1, x_6, x_3, x_5, x_3, x_1$
Classificador 1
$x_3, x_4, x_1, x_5, x_5, x_1$
Classificador 2
Estimador
...
...
$x_1, x_2, x_4, x_2, x_6, x_4$
Classificador m

Boosting

- Conjunto de técnicas
  - Adaboost é uma das mais conhecidas
- A cada iteração
  - Induz classificador
  - Pondera cada exemplo do conjunto de dados completo pelo desempenho do classificador base
    - Quanto mais difícil de ser aprendido, maior o peso associado ao exemplo
- Boosting funciona de forma semelhante a minimização por gradiente descendente

Boosting

Seja o conjunto de dados de treinamento {x_1, x_2, x_3, x_4, x_5}

Exemplos:	x_1,	x_2,	x_3,	x_4,	x_5
Pesos atuais:	0.2	0.2	0.2	0.2	0.2
Classificação:	C	I	C	C	I
Novos pesos:	0.2	0.4	0.2	0.2	0.4

Soma dos pesos = 1.0

C: correta
I: incorreta

Exemplos:	x_1,	x_2,	x_3,	x_4,	x_5
Pesos atuais:	0.2	0.4	0.2	0.2	0.4
Classificação:	C	I	C	I	C
Novos pesos:	0.2	0.6	0.2	0.4	0.4

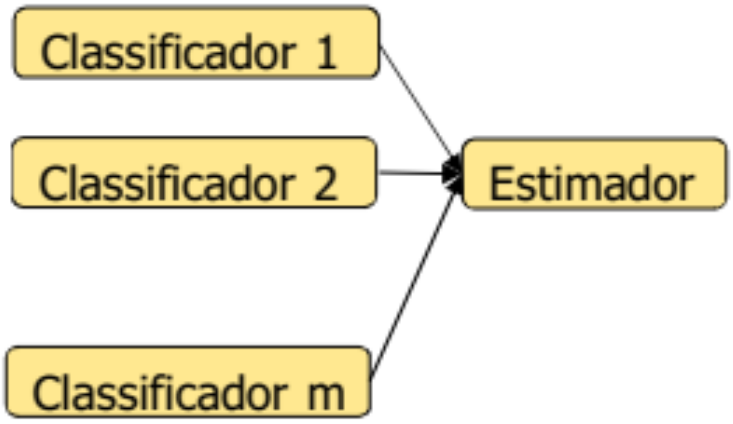
Boosting

- Indicado para classificadores base fracos
  - Acurácia ligeiramente melhor que palpite aleatório
- Convergência rápida
- Pouco indicado para dados com ruídos e pequenos conjuntos de dados
  - Por focar em exemplos difíceis de serem classificados

Stacking

- Um algoritmo estimador aprende a combinar predições de modelos base
  - Modelos gerados por algoritmos base
  - Saídas combinadas por algoritmo estimador
    - Algoritmo de AM

- Algoritmos base podem ser:
  - Homogêneos
  - Heterogêneos



Ensembles de Árvores de decisão

- Combina a predição de várias árvores de decisão

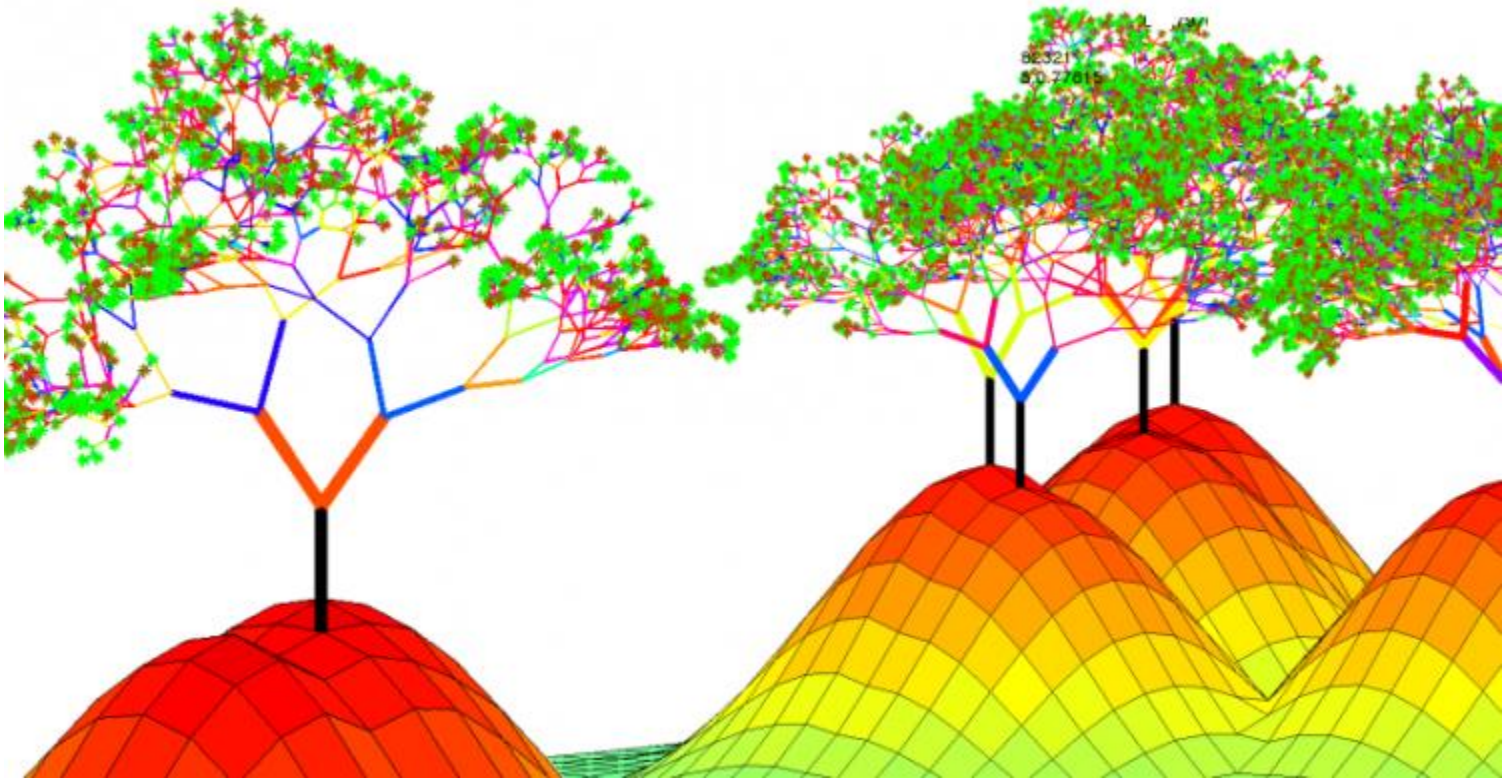
- Duas principais abordagens:
  - Extreme Gradient Boosting
  - Random Forest

Extreme Gradiente Boosting

- XGBoost
- Combina árvores geradas pelo algoritmo CART
- Pondera a resposta de cada árvore para reduzir complexidade do modelo final

Random Forest - Por quê Forests? Muitas árvores!

Por quê Random **Forests?**
Muitas árvores...



Random Forests

- É um algoritmo ensemble proposto por Breiman (veja o artigo **Random Forests)**[1] que constrói muitas árvores de decisão as quais são utilizadas para classificar um novo exemplo por meio do voto majoritário

- Ou seja, o algoritmo de Random Forest (RF) é um termo geral para métodos de ensemble utilizando classificadores do tipo árvore

- Utiliza a amostragem **Bootstrap**

Disponível em https://link.springer.com/content/pdf/10.1023%2FA%3A1010933404324.pdf

Machine Learning, 45, 5–32, 2001

Random Forests

LEO BREIMAN
Statistics Department, University of California, Berkeley, CA 94720

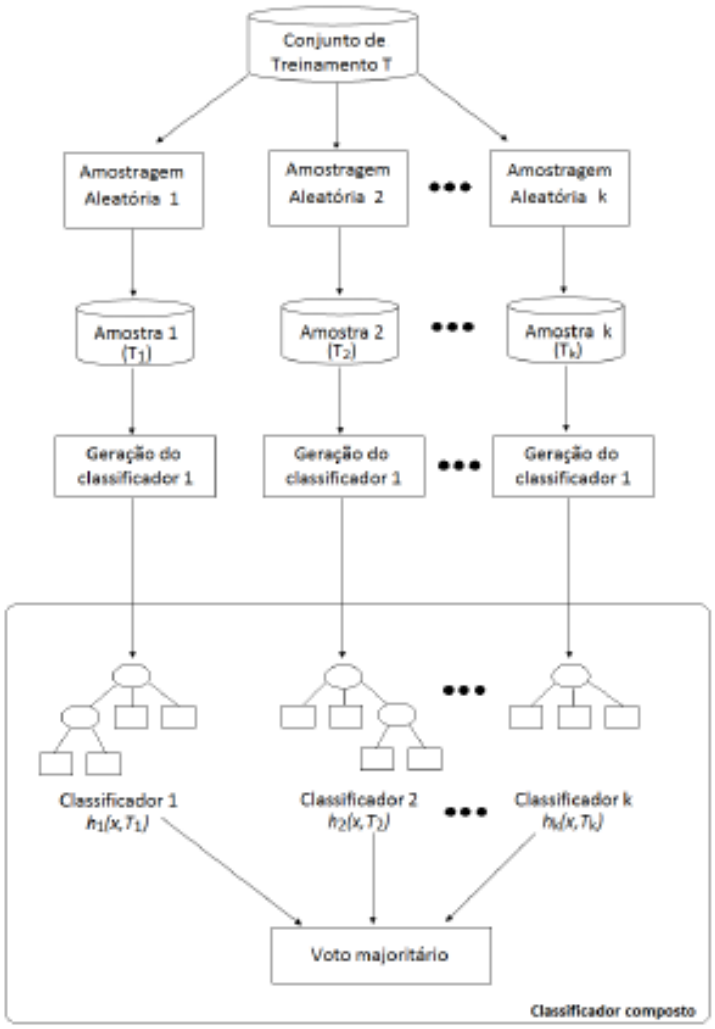
Editor: Robert E. Schapire

Abstract. Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The generalization error for forests converges a.s. to a limit as the number of trees in the forest becomes large. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them. Using a random selection of features to split each node yields error rates that compare favorably to Adaboost (Y. Freund & R. Schapire, *Machine Learning*: *Proceedings of the Thirteenth International conference*, ∗ ∗ ∗, 148–156), but are more robust with respect to noise. Internal estimates monitor error, strength, and correlation and these are used to show the response to increasing the number of features used in the splitting. Internal estimates are also used to measure variable importance. These ideas are also applicable to regression.

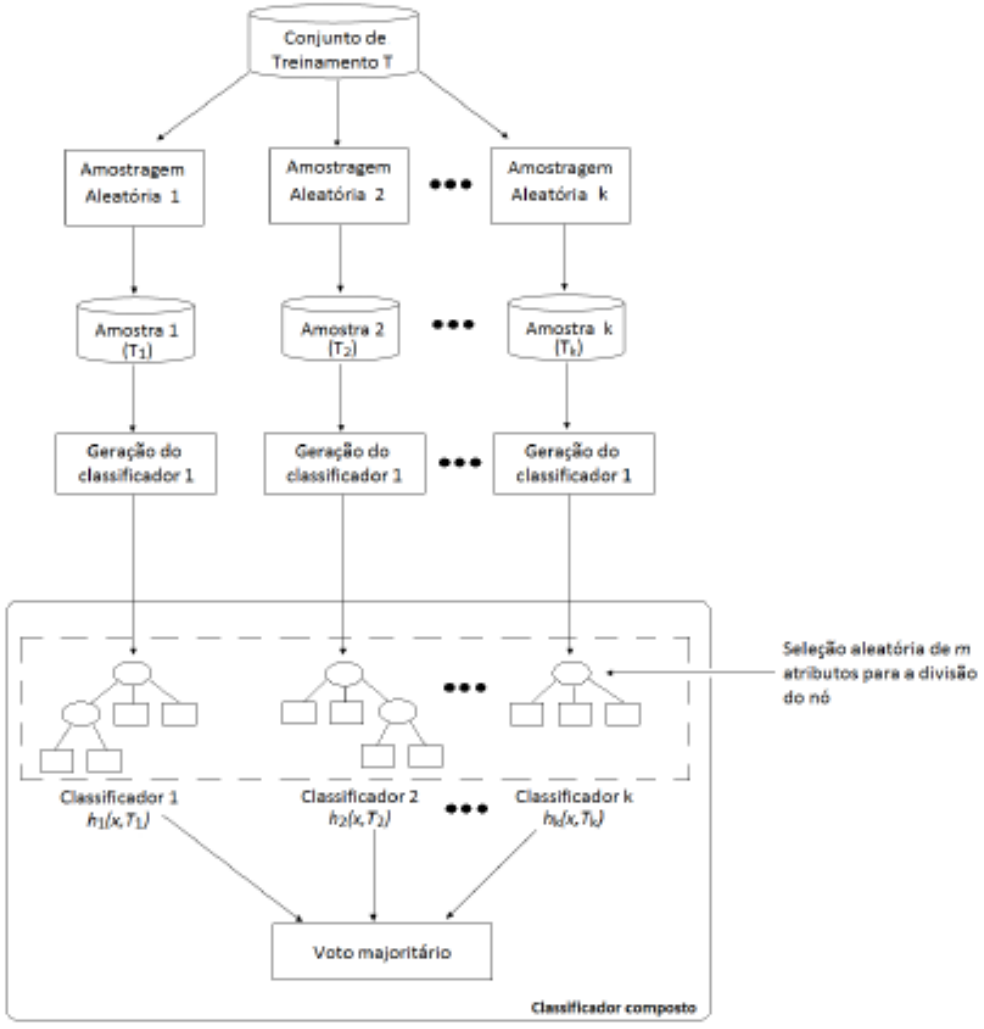
Keywords: classification, regression, ensemble

1. Random forests

1.1. Introduction



Funcionamento do Bagging



Funcionamento do Random Forest

The Random Forest Algorithm

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Algoritmo Random Forest

- Treina cada árvore com amostras geradas a partir do método de amostragem *bootstrap*

- Para cada conjunto de instâncias, o Random Forest considera somente *m* variáveis selecionadas aleatoriamente do conjunto de dados

- Random Forest não faz poda

O resultado final obtido a partir das árvores é dado por:

- Para problemas de **classificação**: voto majoritário
- Para problemas de **regressão**: Média dos valores preditos

Desvantagem da Random Forest

- Difícil extrair o conhecimento das árvores, apesar de o método exibir os atributos mais relevantes

Como definir o número de atributos a serem utilizados?

- Alguns métodos utilizam **raiz quadrada** da quantidade de atributos

- Outros utilizam o **log** da quantidade de atributos

Comandos para Random Forest, em Python

Random Forest: from sklearn.ensemble import RandomForestClassifier

```
modelo = RandomForestClassifier(n_estimators=10, max_features=3, criterion='gini', random_state = 0)
modelo.fit(X_treino, y_treino)
```

Veja os códigos abaixo que estão no CANVAS:

Lendo_e_tratando_arquivo_IRIS.ipynb
Random_Forest.ipynb

Comandos para Random Forest, em Python

Veja no código anterior, como imprimir os **atributos mais relevantes** que o Random Forest considerou para a classificação.

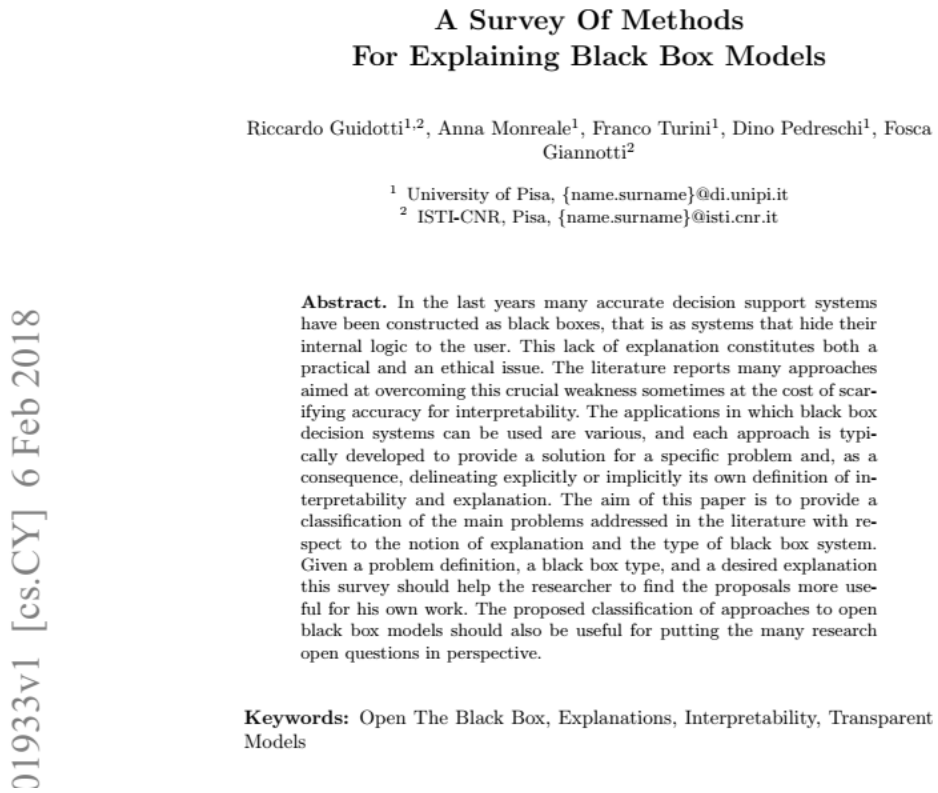
Além disso, veja algumas ferramentas que podem ser úteis para explicabilidade de Random Forest:

1) SHAP (SHapley Additive exPlanations)
2) LIME (Local Interpretable Model-agnostic Explanations)

Investiguem!

Comandos para Random Forest, em Python

Se você deseja maior explicabilidade do modelo, veja o artigo:



A Survey Of Methods
For Explaining Black Box Models

Riccardo Guidotti[1,2], Anna Monreale[1], Franco Turini[1], Dino Pedreschi[1], Fosca Giannotti[2]

[1] University of Pisa, {name.surname}@di.unipi.it
[2] ISTI-CNR, Pisa, {name.surname}@isti.cnr.it

Abstract. In the last years many accurate decision support systems have been constructed as black boxes, that is as systems that hide their internal logic to the user. This lack of explanation constitutes both a practical and an ethical issue. The literature reports many approaches aimed at overcoming this crucial weakness sometimes at the cost of scarifying accuracy for interpretability. The applications in which black box decision systems can be used are various, and each approach is typically developed to provide a solution for a specific problem and, as a consequence, delineating explicitly or implicitly its own definition of interpretability and explanation. The aim of this paper is to provide a classification of the main problems addressed in the literature with respect to the notion of explanation and the type of black box system. Given a problem definition, a black box type, and a desired explanation this survey should help the researcher to find the proposals more useful for his own work. The proposed classification of approaches to open black box models should also be useful for putting the many research open questions in perspective.

Keywords: Open The Black Box, Explanations, Interpretability, Transparent Models

Disponível em https://dl.acm.org/doi/10.1145/3236009

Comandos para Random Forest, em Python

Lembrar de ajustar os hyperparâmetros:
https://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble

Hyper-parameter optimizers

`model_selection.GridSearchCV`(estimator, ...)	Exhaustive search over specified parameter values for an estimator.
`model_selection.HalvingGridSearchCV`(...[, ...])	Search over specified parameter values with successive halving.
`model_selection.ParameterGrid`(param_grid)	Grid of parameters with a discrete number of values for each.
`model_selection.ParameterSampler`(...[, ...])	Generator on parameters sampled from given distributions.
`model_selection.RandomizedSearchCV`(...[, ...])	Randomized search on hyper parameters.
`model_selection.HalvingRandomSearchCV`(...[, ...])	Randomized search on hyper parameters.

Referências

Breiman, L. **Random Forest**. Machine Learning (2001) 45: 5. https://doi.org/10.1023/A:1010933404324
https://link.springer.com/article/10.1023%2FA%3A1010933404324

https://edisciplinas.usp.br/course/view.php?id=78145

https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#workings

Verificar sobre os parâmetros default
https://medium.com/turo-engineering/how-not-to-use-random-forest-265a19a68576