Universidade Federal de São Carlos
Pós-Graduação em Ciência da Computação
Aprendizado de Máquina
Prof. Tiago A. Almeida <talmeida@ufscar.br>

Exercício 8
Máquinas de Vetores de Suporte

Objetivo

O método de Máquinas de Vetores de Suporte, ou SVM (*Suppor Vector Machines*), foi apresentado por Boser *et al.*[1], em 1992, e desde então se consolidou como um dos mais robustos métodos de predição. Variações do modelo já foram aplicadas para problemas de regressão, classificação, agrupamento, detecção de anomalias, entre outros, além de ser um dos métodos mais populares em diferentes domínios de aplicação, de processamento de linguagem natural a recuperação de imagens.

O método funciona de forma semelhante a um modelo de regressão logística. Entretanto, para alcançar uma qualidade maior da classificação, seu principal objetivo é encontrar um hiperplano de separação que não apenas separe as diferentes classes, mas também maximize a margem entre elas. Isso é feito por meio de "vetores de suportes", isto é, pontos próximos da margem de separação.

Por se tratar de um dos métodos de classificação mais famosos da literatura, o SVM já foi implementado para diversas linguagens, em diversas bibliotecas. Em Python, uma das implementações mais populares é a da biblioteca `scikit-learn`. A biblioteca, também conhecida como `sklearn`, é a mais famosa para ciência de dados, IA e aprendizado de máquina em Python. Ela conta com uma documentação extremamente completa e diversas funções e objetos que ajudam desde o protocolo experimental até a predição em si.

Nesse exercício, você implementará o método de Máquinas de Vetores de Suporte através da biblioteca `scikit-learn`. Ao longo do exercício, você testará diferentes tipos de kernel, e observará como se comportam os limites de decisão construídos sobre diferentes conjuntos de dados. Em seguida, você fará um protocolo experimental completo, separando uma base de teste e encontrando os melhores parâmetros por meio de busca em grade, através de funções prontas do `scikit-learn`. Ao término do exercício, espera-se que você compreenda como funciona a interface da biblioteca, assim como aprenda a implementar o SVM.

[1] Bernhard E. Boser, Isabelle M. Guyon & Vladimir N. Vapinik. **A training algorithm for optimal margin classifiers**. *In:* COLT '92: Proceedings of the fifth annual workshop on Computational learning theory, pp. 144–152. 1992.

O exercício

Ao longo do exercício, você deverá completar 5 funções:

- `create_SVM_model`: responsável por construir um objeto do classificador SVM utilizando os parâmetros corretos;
- `fit_model_and_predict`: responsável por treinar o modelo sobre uma base de treinamento e fazer uma predição sobre uma base de teste;
- `holdout`: responsável por separar uma parte do conjunto de dados para treinamento e o restante para teste;
- `grid_search_cross_validation`: responsável por encontrar os melhores parâmetros e resultado atingido para uma determinada métrica, utilizando para isso busca em grade com K-fold;
- `calculate_metrics`: responsável por calcular as métricas de acurácia, precisão, revocação e f-medida.

Preencha o código apenas nos espaços delimitados por comentários, normalmente iniciados por um comentário "`COMPLETE O CÓDIGO AQUI`" e instruções para a implementação.

As implementações devem genéricas e funcionar para qualquer conjunto de dados. Na avaliação, as funções serão testadas em bases com quantidade de amostras e atributos diferentes das fornecidas com o exercício, tendo em comum apenas o nome da coluna que contém a classe das amostras. Não adicione comandos do tipo `print` ou `display` dentro das funções que serão completadas, apenas o código da função..

Os casos de teste

Este exercício possui **5 casos de teste** que buscam avaliar cada uma das funções implementadas. A distribuição de tarefas avaliadas por cada caso é feita da seguinte forma:

- **Casos de teste 1:** testa a função `create_SVM_model`;
- **Caso de teste 2:** testa a função `fit_model_and_predict`;
- **Caso de teste 3:** testa a função `holdout`;
- **Caso de teste 4:** testa a função `grid_search_cross_validation`;
- **Caso de teste 5:** testa a função `calculate_metrics`.