1 Sobre os microprojetos

1.1 Objetivo

O objetivo do microprojeto é realizar um trabalho de casa sobre a análise e implementação de um tema abordado nas aulas com uma parte de programação e de experiência numérica.

- Serão criados grupos de 4-6 alunos;
- Devem produzir um relatório de 15 páginas e um conjunto de slides para apresentar publicamente o vosso trabalho.
- A classificação deste trabalho será: 50% do relatório, 50% da apresentação oral.

1.2 Procedimentos

1. Até quinta-feira 13/05, divulgação dos projetos e formação dos grupos, onde cada grupo deve designar um coordenador.
2. **Dia 14/05, a partir de 17h00**, cada coordenador de grupo envia-me um email com a equipa e a seriação por ordem de preferência de todos os temas que gostariam realizar. Vou atribuir os projetos e tutor consoante a hora de chegada dos emails (excluindo os emails que chegarem mais cedo).
3. A atribuição dos temas será realizada em função da ordem de chegada da seriação. Emails que cheguem antes do horário indicado não serão considerados.
4. O coordenador de grupo contactará o seu tutor para a a organização de sessões de trabalho.
5. O coordenador deve enviar por email, no formatado pdf, o relatório e os slides ao seu tutor **antes de segunda-feira 07/06, 18h00**.
6. As apresentações serão realizadas durante a segunda semana de junho (horário a definir com o tutor).

2 Temas

Os temas, detalhados no resto do documento, são os seguintes:

T1 — *Soft margins* SVM com o SMO

T2 — *Método SAGA*

T3 — *Sparsy Connected Logistic Classifiers*

T4 — *Shallow Logistic classifiers*

T5 — OvA vs ECOC

T6 — OvO vs ECOC

2.1 Soft margins SVM com o SMO

Descrição. Propõe-se neste trabalho a implementação de uma versão simplificada do algoritmo SMO (Sequential Minimal Optimization algorithm) para treinar Soft Margins SVM (C-SVM) na versão dual, sem e com *kernel*. Para as bases de dados a disponibilizar, deverá considerar dois valores possíveis para o parâmetro C e algumas funções para o *kernel*.
Trabalhos a realizar.

1. Descrição e implementação em Python/MatLab da versão simplificada do algoritmo SMO;
2. Descrição e implementação em Python/MatLab da versão dual do algoritmo C-SVM;
3. Validação com as bases de dados do algoritmo C-SVM dual com o SMO.
4. Descrição e implementação em Python/MatLab da versão dual do algoritmo C-SVM com função *kernel*;
5. Validação com as bases de dados do algoritmo C-SVM dual com *kernel* com o SMO.

Os documentos de referências são: http://fourier.eng.hmc.edu/e176/lectures/ch9/node8.html e http://cs229.stanford.edu/materials/smo.pdf

2.2 Método SAGA

Descrição. Os chamados métodos de redução de ruído (do inglês *noise reduction methods*) conseguem redução de ruído, aumentando gradualmente o tamanho do mini-batch usado no cálculo do gradiente estocástico, usando assim, estimativas do gradiente cada vez mais precisas à medida que o processo de otimização prossegue. Uma classe destes métodos é a dos métodos de agregação do gradiente (do inglês *gradient aggregation methods*). Neste trabalho pretende-se implementar o método SAGA, e analisar o seu desempenho na resolução de um dos problemas de machine learning estudados nesta UC.

Trabalhos a realizar.

1. Implementação do método SAGA;
2. Análise do desempenho em termos de eficiência e qualidade das soluções produzidas.
3. Análise dos desempenhos do método SAGA *versus* método do gradiente estocático mini-batch.

Os documentos de referências são: [1] L. Bottou, F. E. Curtis, J. Nocedal. Optimization Methods for Large-Scale Machine Learning. Northwestern University, 2018.

2.3 Sparsy Connected Logistic classifiers

Descrição. O objetivo deste trabalho é a realização de um *logistic classifier* com uma conectividade reduzida afim de melhorar o desempenho computacional e reduzir o risco de *overfitting*. A arquitectura habitual do Logistic classifier será alterada para introduzir a construção de *sparse vectors*. A determinação das conexões na fase do *training* é um dos objetivos principais deste projeto.
Trabalhos a realizar.

1. Implementar the *sparsy connected logistic classifier*;
2. Desenvolvimento de differentes estrategia de aprendizagem e implementação;
3. Avaliação da eficiência dos *logistic classifiers* em função da tecnica de *training*.

Os documentos de referências são `document3-1.pdf` e `document3-2.pdf`.

2.4 Shallow Logistic classifiers

Descrição. A introdução de uma camada oculta numa rede neuronal associada a uma função de ativação não linear é um dos princípios fundamentais em Machine Learning para criar uma hierarquia de entre pequenas estruturas (linhas, arestas, pontos,...) e grandes estruturas complexas (carro, animal, casa,...). Neste trabalho, pretendemos analisar a introdução de apenas uma camada para perceber a hierarquização dos dados e a capacidade de melhorar a predicção com dados mais complexos tais como os digitos ou as letras.

Trabalhos a realizar.

1. Implementação do Shallow Layer Logistic classifier (SLL);
2. Implementação de tecnica de aprendizagem;
3. Experimentação e avaliação do SLL com base de dados de complexidades crescentes;
4. Extensão ao Shallow Layer Softmax classifier.

Os documentos de referências são `document4-1.pdf` e `document4-2.pdf`.

2.5 OvA vs ECOC

Descrição. Propõe-se neste trabalho comparar (tempo computacional e eficiência de aprendizagem) as versões *one-vs-all* (OvA) e *Error-Correcting Output Codes* (ECOC) do classificador *Perceptron* multi-classe/*single-layer* nas versões primal e dual com *kernel*. Deve-se considerar uma base de dados com 4 classes (por exemplo um subconjunto da base de dados MNIST), várias regras de aprendizagem e diferentes funções para o *kernel*.

Trabalhos a realizar.

1. Descrição e implementação do método primal com as versões *one-vs.-all* e *Error-Correcting Output Codes*.
2. Validação e avaliação com a base de dados.
3. Descrição e implementação do método dual com as versões *one-vs.-all* e *Error-Correcting Output Codes*.
4. Validação com a base de dados.
5. Descrição e implementação do método dual com *kernel* com as versões *one-vs.-all* e *Error-Correcting Output Codes*.
6. Validação e avaliação com a base de dados.

Os documentos de referências são `http://ciml.info/dl/v0_99/ciml-v0_99-all.pdf`, nomeadamente as secções 4, 6.2 e 11, e `http://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/lecture_notes/ecoc/ecoc.pdf`.

2.6 OvO vs ECOC

Descrição. Propõe-se neste trabalho comparar (tempo computacional e eficiência de aprendizagem) as versões *one-vs-one* (OvO) e *Error-Correcting Output Codes* (ECOC) do classificador *Perceptron* multi-classe/*single-layer* nas versões primal e dual com *kernel*. Deve-se considerar uma base de dados com 4 classes (por exemplo um subconjunto da base de dados MNIST), várias regras de aprendizagem e diferentes funções para o *kernel*.

Trabalhos a realizar.

1. Descrição e implementação do método primal com as versões *one-vs.-one* e *Error-Correcting Output Codes*.

2. Validação e avaliação com a base de dados.

3. Descrição e implementação do método dual com as versões *one-vs.-one* e *Error-Correcting Output Codes*.

4. Validação com a base de dados.

5. Descrição e implementação do método dual com *kernel* com as versões *one-vs.-one* e *Error-Correcting Output Codes*.

6. Validação e avaliação com a base de dados.

Os documentos de referências são `http://ciml.info/dl/v0_99/ciml-v0_99-all.pdf`, nomeadamente as secções 4, 6.2 e 11, e `http://www.ccs.neu.edu/home/vip/teach/MLcourse/4_boosting/lecture_notes/ecoc/ecoc.pdf`.