Relatório do trabalho 1
Espirais, Distância Euclidiana/Manhattan e Nearest Neighbour

João Paulo Pizani Flor
Mauricio Oliveira Haensch

March 30, 2011

1 Introdução

O primeiro trabalho prático da disciplina consistia em implementar um programa que pudesse gerar 4 tipos de conjuntos de dados:

- Um conjunto de dados 2D (duas variáveis apenas).
- Um conjunto de dados ND (com um grande número de variáveis)
- Conjuntos de dados de espiral simples.
- Conjuntos de dados de espiral dupla.

Além disso, deveriam ser implementados algoritmos para cálculo da Distância de Manhattan e da Distância Euclidiana entre padrões de dimensões arbitrárias, de maneira que possam ser usados para todos os conjuntos de dados gerados. Outro requisito era a implementação do algoritmo de NN (Nearest Neighbour) e kNN (k-Nearest Neighbours), testado para todos os 4 conjuntos de dados.

Observações:

- Os dados 2D e ND podem ser utilizados a partir de conjuntos de dados disponíveis nos sites sugeridos.
- Os dados em espiral devem ser gerados a partir de algoritmos que você mesmo vai implementar. Inclua no algoritmo a possibilidade de introduzir ruído na geração dos dados (através, por exemplo, de uma variável aleatória que modifica levemente o comprimento do raio gerado para o próximo ponto).
- Observe que na espiral dupla cada braço da espiral em seu total representa uma classe.
- O programa deve ser capaz de desenhar os padrões na tela com cores distintas para cada classe.
- Deve ser possível entrar com padrões arbitrários (2D e ND) para que o programa os classifique.

2 Ferramentas utilizadas

A linguagem utilizada para implementação do programa era de livre escolha dos alunos, assim como outras ferramentas que pudessem ser utilizadas para representação gráfica dos padrões, por exemplo. As ferramentas escolhidas para a implementação do trabalho foram:

Linguagem de programação - Python: escolhida por ser uma linguagem já bem conhecida pela equipe, com boas bibliotecas e documentação, além de agilizar o desenvolvimento.

Interface gráfica - PyQt: *binding* do framework gráfico Qt para a linguagem Python, escolhida por já ser conhecida pelos integrantes e por haver uma interface desenvolvida para um trabalho anterior que pode ser reaproveitada com poucas modificações.

O trabalho foi desenvolvido no ambiente Linux/Ubuntu e para executá-lo são necessários alguns pacotes[1] para a interface gráfica:

- pyqt4-dev-tools
- libqtgui4
- libqtcore4

[1] os nomes dos pacotes citados são específicos para a distribuição Ubuntu, para outras distribuições devem ser procurados os pacotes correspondentes.

3 Alguns resultados

Os conjuntos de dados utilizados no trabalho, tanto de 2 como mais dimensões, foram gerados aleatoriamente com distribuição uniforme dentro dos limites da janela. Optamos por não utilizar conjuntos de dados prontos dos sites sugeridos pois teríamos que adaptar os dados para utilizar como entrada em nosso programa, por exemplo, mapear para o domínio numérico os atributos descritivos.

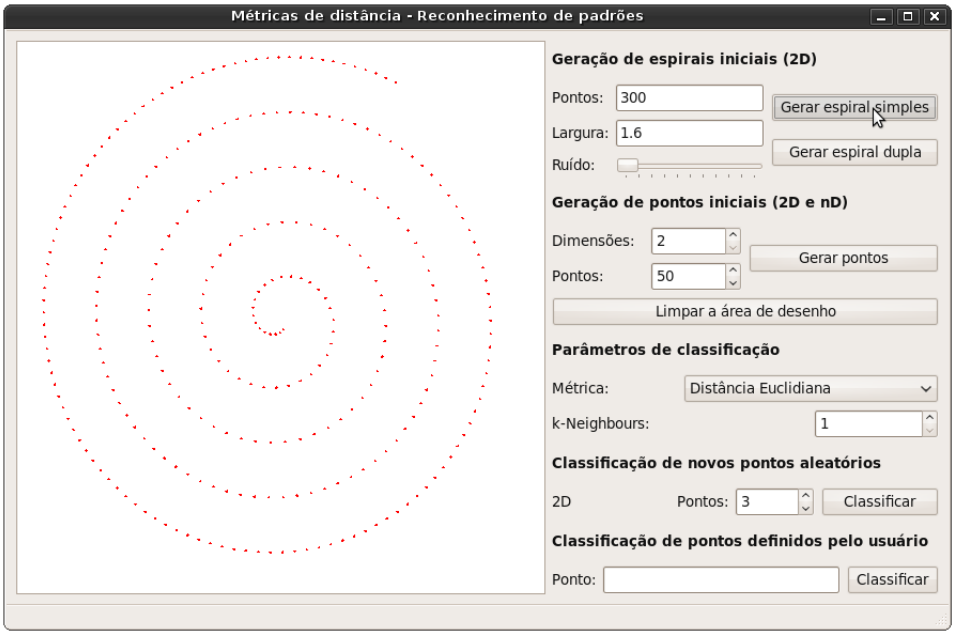
Iremos demonstrar alguns dos resultados alcançados com o trabalho para alguns dos requisitos. Primeiramente a geração de dados iniciais.

3.1 Geração de dados iniciais

A seguir, exemplos da geração de dados de 2 dimensões. A primeira imagem mostra a geração de 20 pontos iniciais distribuídos aleatoriamente no espaço, cada um representando uma classe.

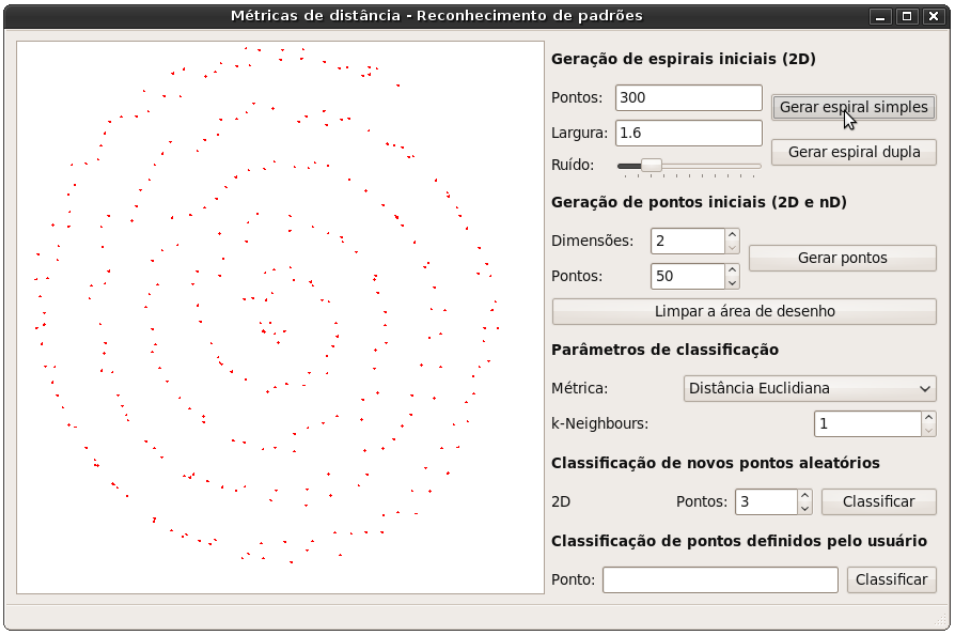
Conjunto inicial de dados com 2D, formado por 20 pontos.

Implementamos também a geração de dados em espiral. A função de geração de espiral foi implementada em coordenadas polares, com um passo posterior de transformação para coordenadas cartesianas. Alguns parâmetros da espiral podem ser ajustados pelo usuário, como a distância entre 2 voltas sucessivas da espiral, o número de pontos e o nível de ruído.

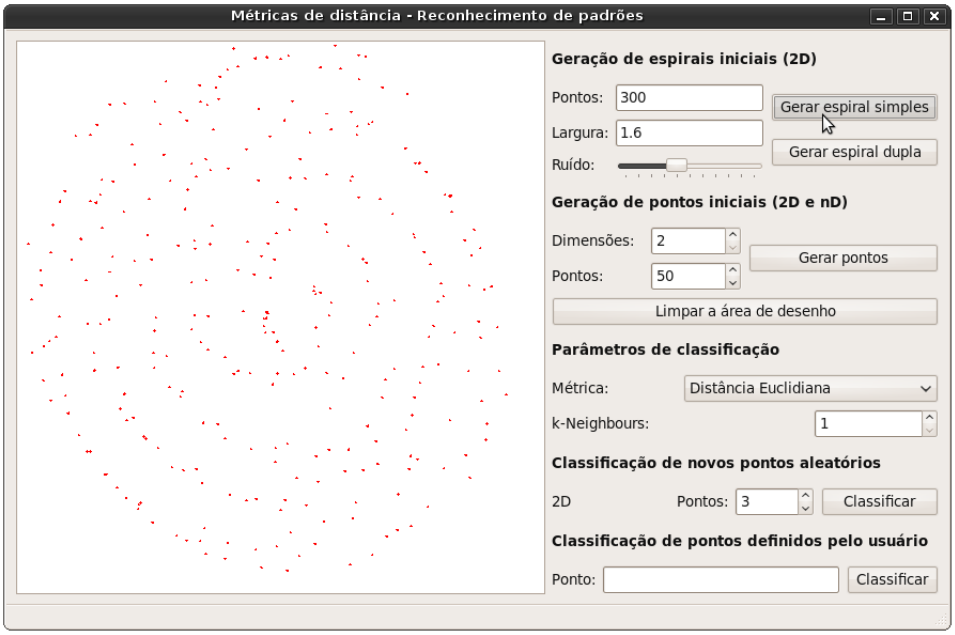


Espiral simples sem ruído.

Na geração de uma espiral com ruído, cada ponto sofre um deslocamento aleatório dentro de um quadrado cujo lado é determinado pelo nível de ruído, que varia de 0 a 2,5. A seguir, 2 imagens com nível crescente de ruído.

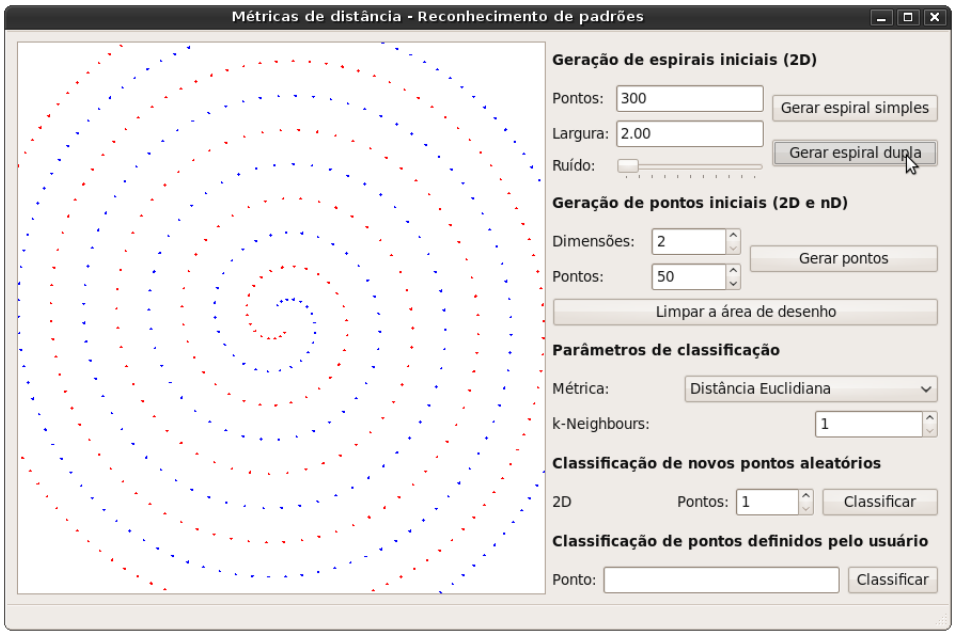


Espiral simples com um pouco de ruído.



Espiral simples com muito ruído.

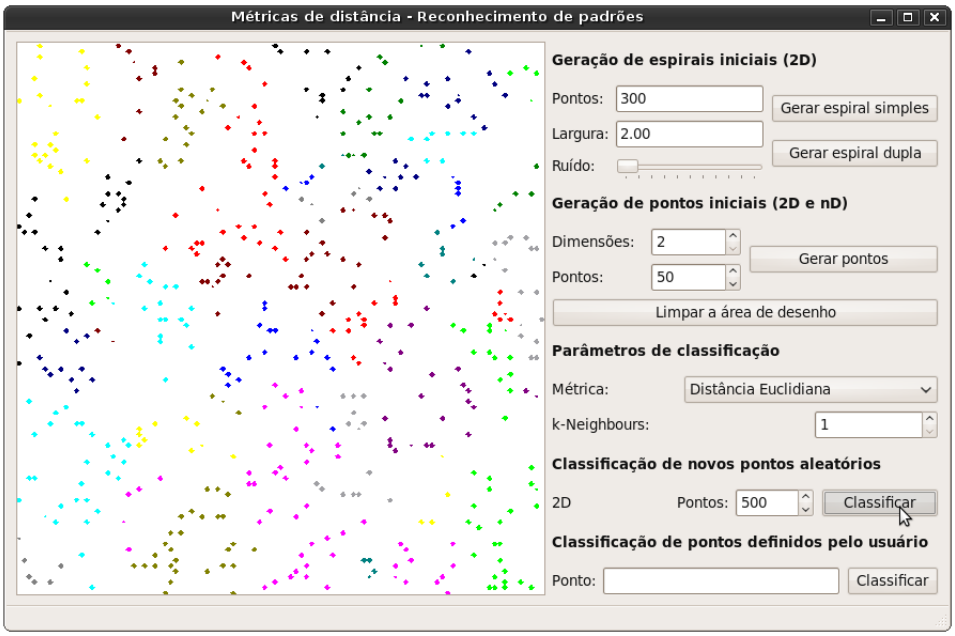
Outro requisito do trabalho era a geração de uma espiral dupla. Isos é feito através da geração de duas espirais simples, deslocando a posição inicial de uma delas em 180 graus com relação à outra, formando dois braços, cada um representando uma classe.



Espiral dupla.

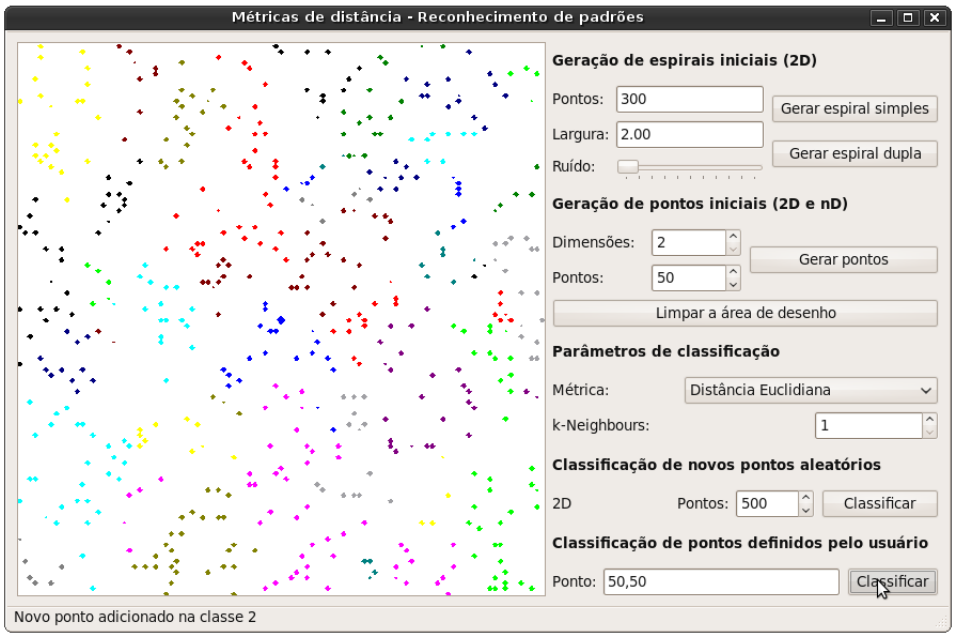
3.2 Classificação de novos pontos

A classificação de pontos deveria ser implementada seguindo o algoritmo kNearestNeighbour (incluindo o caso especial onde k=1) e utilizando 2 métricas de distância: distância euclidiana e distância de Manhattan. A primeira imagem demonstra a classificação de 500 pontos aleatórios sobre o conjunto de dados inicial de 2 dimensões já apresentado. Os novos padrões foram classificados com k=1 e utilizando a distância euclidiana.



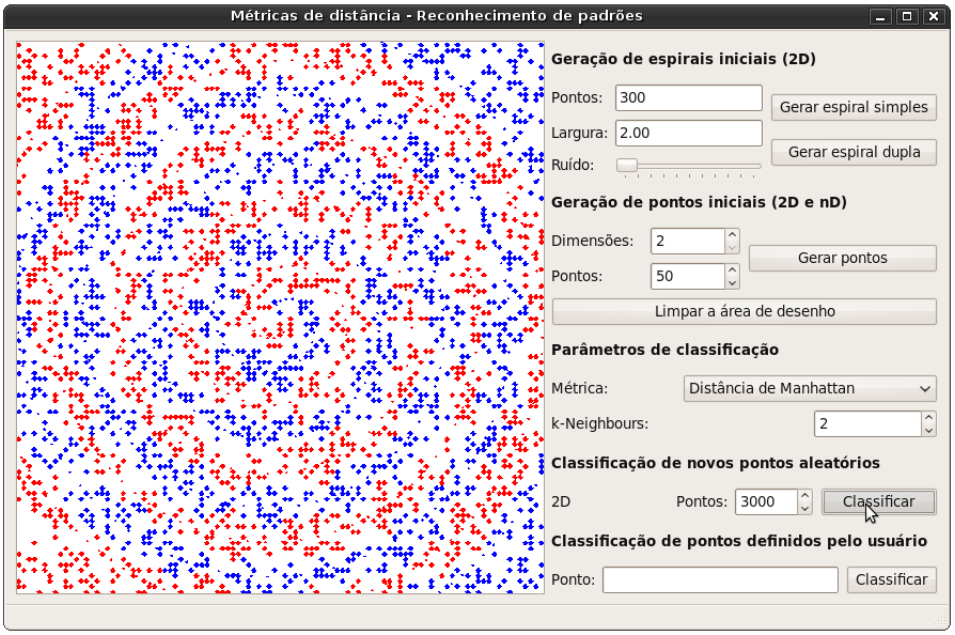
500 pontos classificados por Nearest Neighbour com distância Euclidiana.

Além da geração aleatório, é possível que o usuário determine um ponto arbitrário a ser classificado. A imagem a seguir mostra essa possibilidade, no caso o ponto central da área de desenho (coordenadas x=50 e y=50), também classificado com o mesmo algoritmo e métrica que o caso anterior.



Classificação de um ponto escolhido pelo usuário, através de Nearest Neighbour com distância Euclidiana.

A seguir, duas imagens de espirais duplas. Na primeira figura foram adicionados 3000 pontos aleatórios, classificados pelo algoritmo *kNearestNeighbour* com k=2 e utilizando a *distância de Manhattan*. A segunda figura demonstra uma espiral após a adição de um grande número de pontos, quase totalizando a área disponível da interface, mostrando as regiões cobertas por cada classe da espiral dupla.



Espiral dupla com 3000 pontos aleatórios (kNearestNeighour com k=2 e distância de Manhattan)

Espiral dupla com trocentos pontos.

4 Referências

- Python - http://www.python.org/
- Qt - http://doc.qt.nokia.com/
- pyQt - http://www.riverbankcomputing.co.uk/software/pyqt/intro