# Classificadores multi-classe

- Até aqui só lidámos com classificadores binários
  - *Cada exemplo de entrada é classificado numa de duas classes (uma considerada positiva, a outra negativa)*
  - *Exemplos: previsão da ocorrência ou não de precipitação ou deteção de determinada patologia*
- Mas muitos dos problemas com que lidamos são **multi-classe**
  - *Cada exemplo de entrada é classificado numa de várias classes*
  - *Exemplos: classificação duma planta iris (numa de 3 espécies) ou reconhecimento automático de dígitos manuscritos (ou de outros carateres)*
- Embora existam classificadores intrinsecamente preparados para a multi-classe, muitos outros foram idealizados especificamente para classificação binária
  - *Mas mesmo esses são facilmente adaptáveis a problemas multi-classe*

# Adaptação de classificadores binários para multi-classe

Os classificadores binários são convertidos em classificadores multi-classe, usando, por norma, uma de duas estratégias:

- a estratégia do **um-contra-todos** – é induzido um classificador binário para cada uma das n classes, encarando as restantes como a classe negativa (depois, para cada exemplo de entrada é escolhida a classe do modelo que a conseguir prever com maior probabilidade)
  - *por exemplo, para classificarmos as plantas iris nas 3 espécies, são criados 3 classificadores binários, para distinguir: versicolor das restantes, virginica das restantes, e setosa das restantes (já para reconhecer dígitos manuscritos são necessários n=10 classificadores binários)*
  - *Esta é a opção mais usada. Além de apresentar uma eficiência linear em relação ao número de classes (apenas n classificadores são necessários), tem também como vantagem a sua interpretabilidade. Como cada classe é representada por um classificador, é possível obter conhecimento sobre a classe analisando o classificador correspondente.*
- e a estratégia do **um-contra-um** – é induzido um classificador binário para cada par de classes, o que perfaz um total de n(n-1)/2 classificadores, em que n é o número de classes (depois, para cada exemplo de entrada é escolhida a classe mais votada, ou seja, aquela que foi mais vezes escolhida nos vários confrontos diretos com as restantes)
  - *por exemplo, para classificarmos as plantas iris nas 3 espécies, são criados 3 classificadores binários, para distinguir: versicolor da virginica, versicolor da setosa, e setosa da virginica (já para reconhecer dígitos manuscritos são necessários n(n-1)/2=10(9)/2=45 classificadores binários)*
  - *É uma abordagem menos eficiente, dada a sua complexidade em relação ao números de classes ser de ordem quadrática (são necessários n(n-1)/2 classificadores).*
  - *No entanto, note-se que cada um dos seus classificadores binários apenas envolve um pequeno subconjunto dos dados (apenas os exemplos classificados numa das duas classes – em média 2/n dos exemplos do dataset) – no um-contra-todos, cada classificador binário usa todo o dataset*

# Classificadores multi-classe e multi-*label*

- Todos os classificadores do Scikit-learn podem ser aplicados a problemas multi-classe
  - *enquanto que as random forests (RandomForestClassifier) e as redes neuronais (MLPClassifier) já estão intrinsecamente preparadas para multi-classe,*

    *as SVM (SVC) socorrem-se do esquema um-contra-um, por defeito*

    *e, por exemplo, a regressão logística (LogisticRegression) do um-contra-todos*

    Caso se pretenda, é possível alterar a estratégia mutli-classe desses algoritmos usando os meta estimadores OneVsRestClassifier e OneVsOneClassifier do módulo sklearn.multiclass – duas classes Wrapper

- Não confundir classificação multi-classe com classificação **multi-*label***
  - *Contrariamente à classificação multi-classe, em que cada exemplo de entrada apenas assume uma classe entre as n possíveis, na classificação multi-label cada exemplo pode assumir em simultâneo várias classes*

    Por exemplo, um mesmo filme pode ser classificado nos géneros histórico, biográfico e drama

# Matriz de confusão multi-classe

Como avaliar classificações multi-classe?

- O conceito de matriz de confusão usado em classificação binária é facilmente generalizado para o caso de termos n classes
  - *basta assumir que o valor da linha i e coluna k da matriz de confusão representa o número de exemplos do conjunto de teste que, pertencendo à classe $C_i$, foram classificadas pelo modelo como sendo da classe $C_k$*
- Por exemplo, se a matriz de confusão que se segue espelhar o resultado dum modelo na classificação do dataset iris, podemos concluir o seguinte:
  - *O modelo tem um bom desempenho, uma vez que em 38 plantas usadas para teste, apenas classificou mal uma delas: uma planta versicolor foi classificada como virginica*

| previsto / real | setosa | versicolor | virginica |
|---|---|---|---|
| setosa | 13 | 0 | 0 |
| versicolor | 0 | 12 | 1 |
| virginica | 0 | 0 | 12 |

# Métricas para classificação multi-classe

- A generalidade das restantes métricas são definidas para classificação binária
  - *Por isso, na avaliação da classificação multi-classe o que normalmente se faz é usar a média duma dessas métricas sobre as várias classes*
- No entanto, a média da métrica 'binária' a aplicar na avaliação multi-classe pode assumir formas diferentes:
  - *Macro – calcula-se a métrica binária respetiva para cada classe e considera-se depois a média aritmética desses valores*
  - *Micro – calcula-se a métrica 'binária' globalmente, contabilizando todos os verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos, independentemente das classes*
  - *Ponderada – idêntica à Macro, mas pondera a média em função do número de exemplos de cada classe*
- Um dos critérios para escolher o tipo de média a usar é o nível de desbalanceamento das classes
  - *Se as classes tiverem um número bastante diferente de ocorrências, poderá ser mais indicado a média Macro, uma vez que dá a mesma importância a todas as classes*
  - *Mas se quisermos, ainda assim, dar mais alguma importância às classes maioritárias, podemos sempre usar a média Ponderada*
  - *Também a média Micro é sensível às classes dominantes. Pela sua natureza, dá maior peso a classes com mais exemplos.*

# Exemplificação do cálculo das médias Macro e Micro

- Exemplifica-se o cálculo das médias **Macro** e **Micro**, como duas formas alternativas de transformar a métrica binária 'precisão' numa métrica multi-classe,
  - *o mesmo procedimento pode depois ser seguido para se calcular a média Macro e Micro de qualquer outra métrica de classificação binária*
  - *Refira-se, no entanto, que o Scikit-learn já põe ao nosso dispor funções que calculam todas essas médias (ver próximo diapositivo)*

- Exemplificação do cálculo da **precisão** Macro e Micro para avaliação da classificação das plantas iris

  *Como se sabe, a precisão dum classificador binário é dada pela taxa VP/(VP+FP)*

  - *Logo, para determinar a precisão Macro, começa-se por calcular essa mesma taxa para cada uma das espécies (por exemplo, para a espécie 'virginica', considera-se para VP a quantidade de plantas que foram classificadas corretamente como virginicas (12), e para FP a quantidade de plantas que foram classificadas erradamente como virginicas (1), a que corresponde uma precisão de 92.3%), e calcula-se, depois, a média de todos esses valores ( (92.3+100+100)/3=97.44% ).*
  - *Para determinar a precisão Micro, calcula-se globalmente VP/(VP+FP), considerando para VP a totalidade das plantas que foram classificadas corretamente independentemente da espécie (37 plantas iris), e para FP a totalidade das plantas que foram classificadas de forma errada (1 planta iris). No exemplo considerado a precisão Micro será então de 97.36%.*

    Repare-se que a média Micro da precisão acaba sempre por ser igual à média Micro, quer da sensibilidade, quer da acurácia (tx de acerto), quer da própria F1.

# Exemplos de métricas multi-classe

- No Scikit-learn, a avaliação de modelos multi-classe faz-se com recurso às funções que implementam as métricas conhecidas (do módulo *sklearn.metrics*), mas com o seu parâmetro *average* assumindo o valor '*macro*', '*micro*' ou '*weighted*', *em concordância com o tipo de média que se pretende calcular. Seguem-se alguns exemplos:*
  - *roc_auc_score(..., average='macro') – média aritmética dos AUCs das classes*
  - *roc_auc_score(..., average='micro') – AUC global, contabilizando todos os verdadeiros positivos e todos os falsos positivos, independentemente da classe*
  - *roc_auc_score (..., average='weighted') – média dos AUCs das classes, ponderada pelo número de ocorrências de cada classe*
  - *f1_score(..., average='macro') – média aritmética dos valores F1 das classes*
  - *f1_score(..., average='micro') – F1 global, contabilizando todos os verdadeiros positivos, falsos negativos e falsos positivos, independentemente da classe*
  - *f1_score(..., average='weighted') – média dos F1, ponderada pelo número de ocorrências de cada classe*
  - *precision_score(..., average='macro') – média aritmética das precisões das classes*
  - *precision_score(..., average='micro') – precisão global, contabilizando todos os verdadeiros positivos e todos os falsos positivos, independentemente da classe*
  - *precision_score(..., average='weighted') – média das precisões, ponderada pelo número de casos verdadeiros de cada classe.*
- Para se perceber que interpretação dar aos conceitos 'classe positiva' e 'classe negativa' no cálculo das métricas multi-classe, recuperemos o exemplo mostrado da matriz confusão
  - *O exemplo mal classificado, representa, ao mesmo tempo, um falso positivo (FP) relativamente á espécie virginica, e um falso negativo (FN) relativamente à espécie versicolor.*

# Aprendizagem não supervisionada

- Todos os métodos de ML que estudámos até este momento tinham por base técnicas de aprendizagem supervisionada
- Existem outros métodos, com proeminência noutras tarefas específicas de análise e exploração de dados, que se suportam em técnicas de aprendizagem não supervisionada
  - *Esses métodos recebem como entrada dados não classificados (datasets sem a variável alvo)*
  - *Entre eles, destaca-se o método de clustering*

- O método de clustering, também conhecido por método de agrupamento ou de segmentação de dados, usa-se quando se pretende criar um conjunto finito de categorias, para segmentar uma população heterogénea em grupos mais homogéneos.
  - *Trata-se de uma técnica que permite agrupar objetos (observações) em grupos homogéneos relativamente a uma ou mais características comuns.*
    - Na situação ideal, os objetos de cada grupo devem ter a maior similaridade entre eles, e a menor entre eles e os objetos pertencentes a outros grupos, de forma a garantir homogeneidade dentro de cada cluster e heterogeneidade entre clusters.

# Clustering

- A construção de *clusters* envolve a seleção de características e de medidas de similaridade entre os objetos
  - *Normalmente, as medidas de similaridade usadas são as de distância tradicionais como, por exemplo, a euclidiana.*
- O *clustering*, por vezes, é uma das primeiras tarefas a efetuar quando se pretende realizar um estudo de *data mining*, uma vez que permite identificar grupos, a partir dos quais poderão ser efetuadas outras análises mais orientadas e detalhadas, como é o caso de certas previsões.
- Tratando-se de um método não supervisionado, não é exigido que sejam definidos *clusters iniciais* no momento em que o algoritmo começa,
  - *pois não se pretende associar a um cluster específico cada uma das instâncias.*
  - *Pretende-se, pelo contrário, identificar, a partir do conjunto inicial de instâncias, clusters que traduzam grupos de instâncias similares.*

# Exemplos de aplicação do método de Clustering

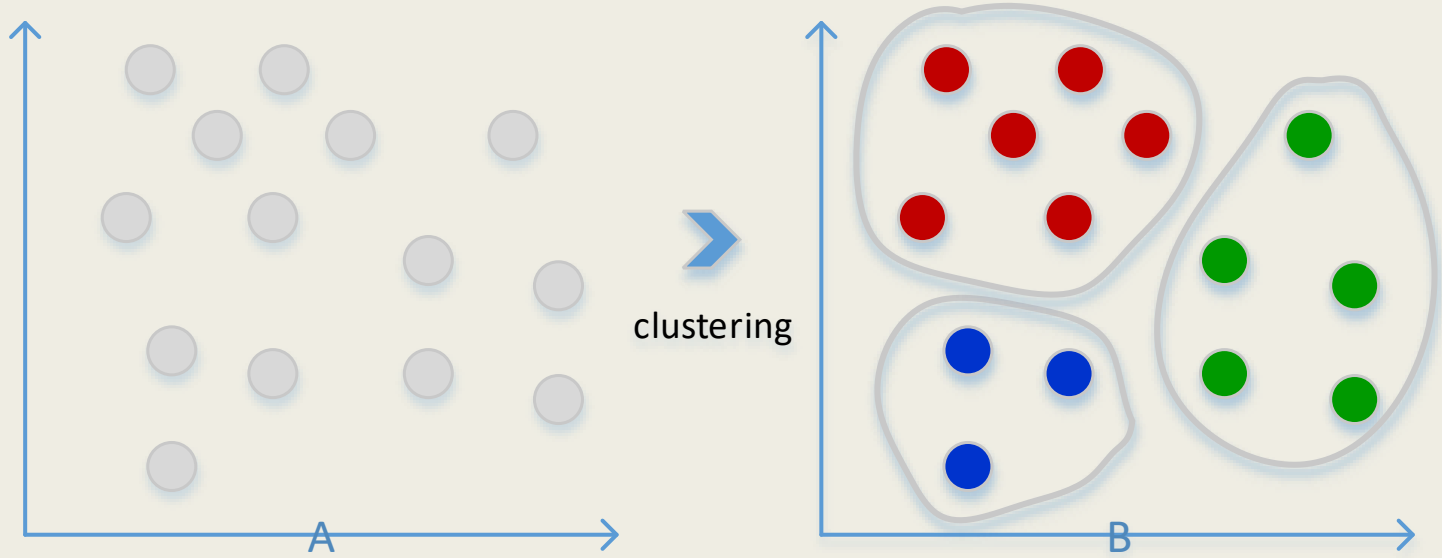
1. Saber que diferentes medidas devem ser consideradas para uma nova peça de vestuário que se pretenda produzir (exercício 25)
   - *o método tratará de segmentar os indivíduos de uma dada amostra populacional em vários subgrupos, cada um dos quais com indivíduos com caraterísticas especificas similares (como por exemplo, altura e medida de cintura)*
   - *depois serão as caraterísticas médias (altura e medida da cintura médias) de cada grupo encontrado que servirão de referência para os vários tamanhos da peça de roupa a produzir*
2. Segmentar os alunos com comportamentos e níveis de aproveitamento escolar semelhantes, com o objetivo de prever e analisar os respetivos processos de aprendizagem em grupos mais homogéneos.
3. Segmentar os alunos de acordo com as suas características pessoais e de aprendizagem para serem aplicadas metodologias de ensino mais personalizadas.

# O algoritmo K-Means

- O K-Means (k médias) é uma das técnicas de clustering mais conhecidas
- Sendo aplicada a dados não classificados ou rotulados (método não supervisionado),
  - *tem por objetivo encontrar K subgrupos nos dados analisados, para um valor de K fornecido*
- Para formar esses subgrupos (os *clusters*), tenta encontrar k centroides que representem adequadamente o centro de cada um deles:
  - *começa por "plantar" K centróides, distribuídos aleatoriamente no espaço de caraterísticas*
  - *depois, iterativamente,*
    - associa a cada um dos centróides as instâncias que lhes são mais próximas,
    - reajusta cada um dos centróides ao seu novo grupo de instâncias
  - *concluindo esse processo quando todos os grupos (clusters) estabilizarem, i. e., não sofrerem alterações (ou, equivalentemente, quando os centróides não mudarem de posição)*

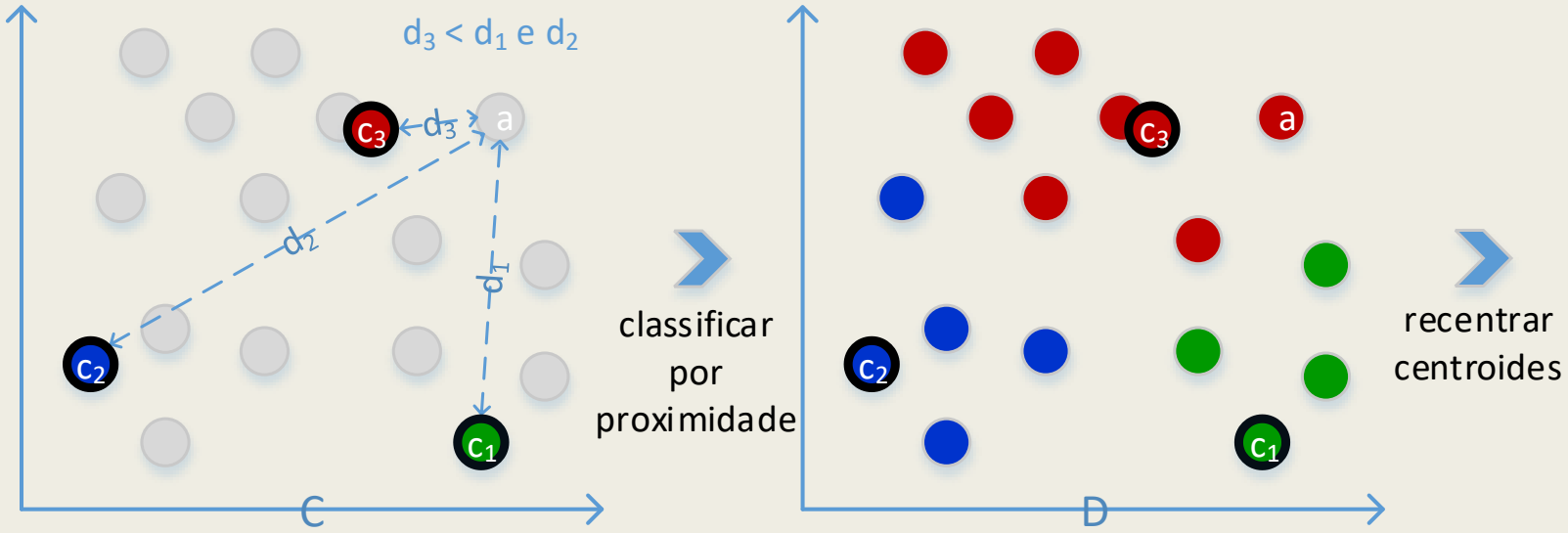
# K-Means – Exemplificando para K=3

- Suponha que tem um conjunto de pontos 2D (representado observações/instâncias com dois atributos)
  - *e que o objetivo é agrupar esses pontos em 3 diferentes clusters que permitam descobrir, dentro de cada um deles, um padrão que os caraterize*
- A figura que se segue ilustra um possível resultado quando o objetivo for então separar os dados em 3 grupos (k=3)



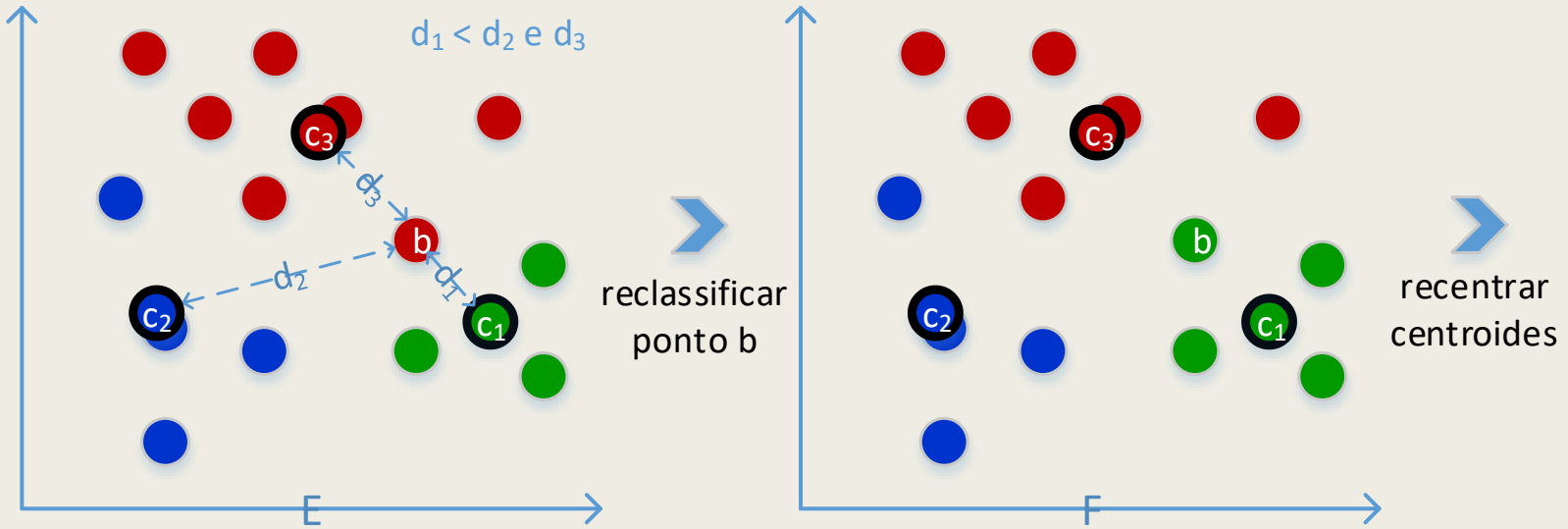
# K-Means – Exemplificando para K=3

- O algoritmo começa por escolher aleatoriamente 3 centróides ($C_1$, $C_2$ e $C_3$, no Gráfico C)
- Depois, mede a distância de cada ponto (instância) a cada um dos 3 centróides, de forma a associá-lo ao que estiver mais próximo
  - *por exemplo, como ilustrado no Gráfico C, a distância $d_3$, do ponto (a) ao centróide $C_3$, é menor que as distâncias $d_1$ e $d_2$, desde esse ponto aos centróides $C_1$ e $C_2$, respetivamente*
    - logo, o ponto (a) será classificado como pertencendo ao Cluster 3
- No Gráfico D representam-se os pontos depois de associados aos respetivos centróides



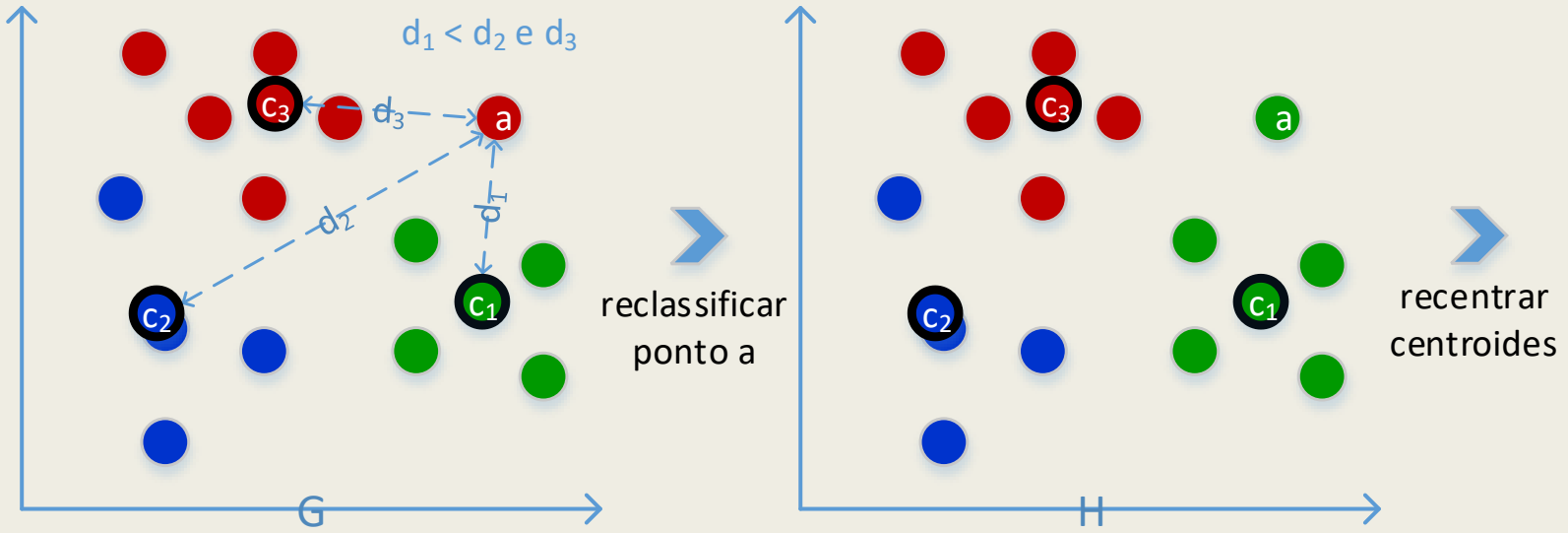

- Seguidamente todos os centróides vão ser reposicionados, passando a refletir cada um deles a média (o centro geométrico) de todos os pontos incluídos no seu cluster, resultando a ilustração do Gráfico E

# K-Means – Exemplificando para K=3



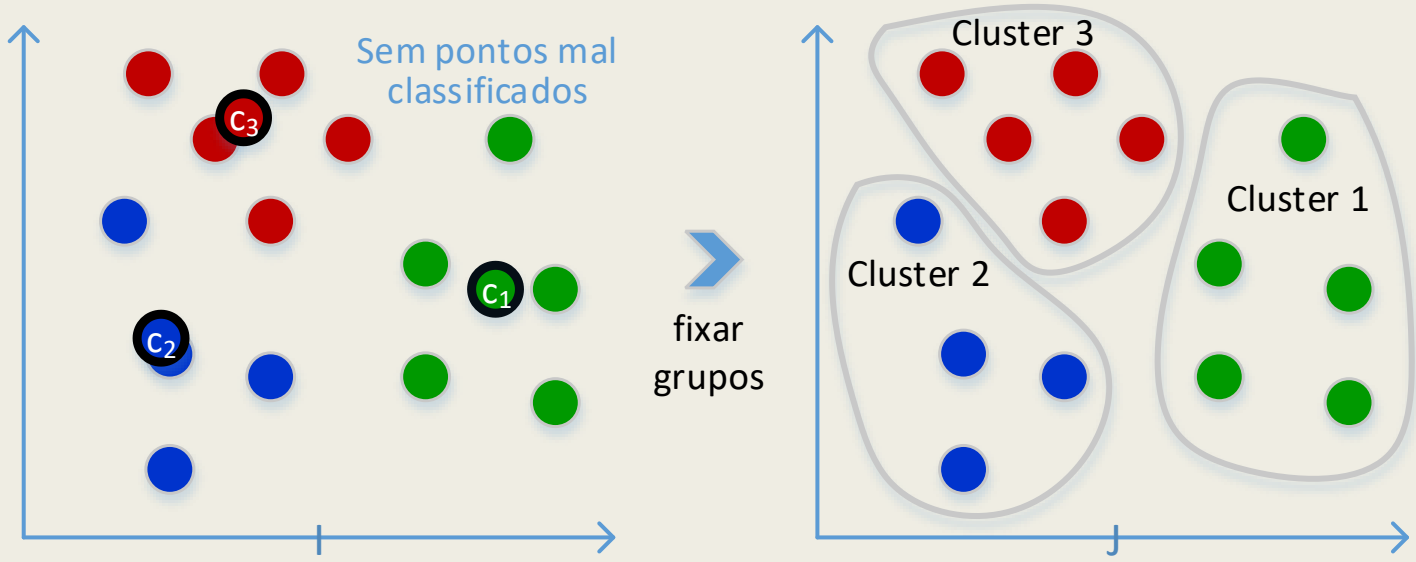

- Novamente, é medida a distância de cada ponto do gráfico (instância) a cada um dos 3 novos centróides, de forma a reclassificar algum que tenha ficado mais próximo do centróide de outro cluster do que do seu
  - *no exemplo ilustrado no Gráfico E, a distância $d_1$, do ponto (b) ao centróide $C_1$, passou a ser menor que as distâncias $d_2$ e $d_3$, desde esse ponto aos centróides $C_2$ e $C_3$, respetivamente*
    - logo, o ponto (b) é reclassificado, passando do Cluster 3 para o Cluster 1 (Gráfico F)
- Como os clusters 1 e 3 foram alterados, os seus centróides voltam a ser recalculados, resultando a ilustração do Gráfico G

# K-Means – Exemplificando para K=3




- Agora é o ponto (a), que  devido ao deslocamento dos centróides, passou a estar mais próximo do centróide de outro cluster do que do seu
  - *Daí transitar do Cluster 3 (Gráfico G) para o Cluster 1 (Gráfico H)*
- Volta-se a recentrar os centróides $C_1$ e $C_3$, tal como ilustrado no Gráfico I

# K-Means – Exemplificando para K=3




- Como (apesar dos centróides se terem movido um pouco) todos os pontos continuam associados efetivamente aos centróides mais próximos, concluiu-se a procura dos 3 clusters, ilustrados no Gráfico J.

# Redução de dimensionalidade

- O objetivo da redução da dimensionalidade de um dataset é passar a representá-lo de uma forma mais compacta, reduzindo o número de variáveis preditivas
  - *num dataset com menos colunas passa a ser mais fácil aplicar algoritmos sofisticados, que envolvem, normalmente, grande esforço computacional*
  - *acresce que os algoritmos, quando aplicados a datasets de elevada dimensionalidade, têm tendência a se ajustar demasiado aos dados de treino (overfiting), comprometendo assim a sua capacidade de generalização*
  - *e se a redução resultar em 2 ou 3 variáveis, nomeadamente passa a ser possível representar graficamente o dataset*
- Se a informação mais importante intrínseca aos dados poder ser representada por menos dimensões, por que não tentar excluir as dimensões em excesso?...
  - *A redução de dados leva, no entanto, a alguma perda de informação,*
  - *mas o uso de algoritmos mais evoluídos pode compensar em grande medida essa perda*
- A redução de dimensionalidade pode ser conseguida de diferentes formas
  - *e a redução do dataset em si pode também ser conseguida reduzindo, por amostragem, o número de observações (i.e., diminuindo a sua dimensão), em vez de se reduzir a dimensionalidade – no fundo, eliminando linhas do dataset, em vez de colunas.*

# Métodos de Redução de dimensionalidade

- **Seleção** de caraterísticas (Feature Selection)

  *apenas algumas variáveis preditivas são escolhidas, sem envolver qualquer transformação das mesmas*

  - *a informação retida pelas colunas não selecionadas é totalmente perdida*
  - *existem vários métodos para fazer essa seleção, quer manuais, quer automáticos ou semiautomáticos*

- Redução por **transformação** de tipo

  *os dados passam a ser representados noutros formatos*

  - *por exemplo, séries temporais podem ser convertidas em dados multidimensionais de menor tamanho e complexidade, usando transformações wavelet*

- **Projeção** de caraterísticas (ou extração de caraterísticas)

  *projetam os dados num novo espaço de menor dimensão*

  - *Redução por transformação rotacional*

    as correlações nos dados são aproveitadas para representá-los num menor número de dimensões

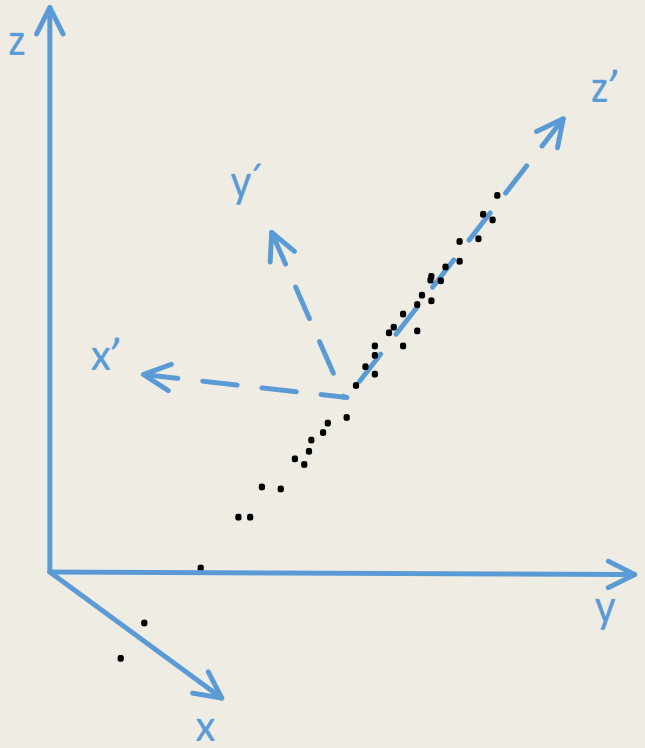
    - a  informação de todas as colunas iniciais vai estar de alguma forma representada nas novas colunas, ainda que sejam em menor número
    - dois métodos importantes que fazem este tipo de redução, e a que daremos mais atenção, são o PCA (*principal component analysis*) e o SVD (*singular value decomposition*)

  - *Redução com manifolds (método t-SNE)*

# Transformação Rotacional

- Em datasets reais existe sempre alguma correlação entre as suas variáveis preditivas, que pode, inclusive, passar despercebida ao analista de dados
  - *essa correlação traduz-se em informação redundante que vai penalizar a eficiência do algoritmo de aprendizagem*

- Para se perceber melhor o efeito da rotação nos dados, repare-se nos dados 3D (3 variáveis explicativas, x , y e z) da figura
  - *neste caso, se for aplicada uma rotação aos dados que os transfira para as novas coordenadas* x'y'z', há uma clara vantagem, na medida em que os dados passam a poder ser (aproximadamente) representados unicamente por uma das novas coordenadas: a variável z'
- Agora a questão importante a saber é de que forma se poderá determinar a rotação ideal
  - *a que conduza à maior remoção de redundâncias (correlações entre variáveis) – que transfira os dados para o novo sistema de eixos* x'y'z', *no exemplo da figura*




Os métodos que perseguem esse objetivo são, precisamente, o PCA e o SVD

- *Ambos integram o módulo decomposition do Scikit-learn* (*PCA* e *TruncatedSVD*)

# PCA – análise de componentes principais

- O PCA é o principal método de redução de dimensionalidade
  - *Tem por base fundamentação que combina cálculo estatístico com álgebra linear*
- Deixando a sua fundamentação matemática para outros estudos mais especializados, foquemo-nos no essencial do seu funcionamento:
  - *não precisamos indicar de quantos atributos será a redução*
  - *antes de aplicar a transformação de rotação propriamente dita, o método começa por centrar em torno da origem do sistema de eixos o conjunto de dados, subtraindo a cada uma das observações o valor médio de todo o conjunto*
  - *procura o conjunto de direções ortogonais que represente o conjunto de dados inicial*
  - *em termos gerais, o PCA procura mapear o dataset original para um novo espaço de atributos em que os vetores coluna do novo dataset passem a ser ortogonais entre si*
    - cada uma das novas colunas passa a representar, por ordem decrescente, uma certa percentagem da variância dos dados (o máximo que conseguir)

      *(a rotação perfeita seria a que conseguisse representar 100% da variância numa só coluna)*

# Uso de funções de Kernel no método PCA

- O PCA é, pela sua natureza, um método linear,
  - *o que significa que tem dificuldade em lidar com dados que não sejam linearmente separáveis*
  - *e esse é o caso da maior parte dos conjuntos de dados com que lidamos no mundo real*
- Uma forma de se conseguir capturar a não linearidade dos dados, é aplicar algum tipo de transformações não lineares a esses dados e só depois reduzir a sua dimensionalidade pelo método PCA
  - *essas transformações são asseguradas por funções Kernel, que projetam os dados noutro espaço dimensional*
- Exemplos de funções de Kernel usadas:
  - *polinomial*
  - *função de base radial*
  - *sigmoidal*
  - *cosseno*
- As transformações com funções de Kernel podem ser efetivamente úteis na separação dos dados
  - *mas podem também contribuir para o overfitting*

    *(esse efeito pode ser controlado e minimizado com treino-validação)*
- É recomendável a normalização dos dados antes da aplicação do PCA

# SVD – decomposição em valores singulares

- No método SVD, a matriz de dados (que representa o dataset), é decomposta no produto de 3 matrizes com caraterísticas especiais que as tornam mais fáceis de analisar e manipular.
- O método SVD é bastante similar ao PCA,
  - *excetuando o facto da factorização no SVD ser feita sobre a matriz de dados e não sobre a matriz de covariância*
  - *podendo, por isso, e contrariamente ao PCA, poder ser aplicado em matrizes esparsas*

- SVD truncado

  *A diferença do SVD truncado em relação ao SVD normal está no número de colunas resultantes*
  - *enquanto que o SVD normal produz matrizes com n colunas, sendo n o número de atributos do problema,*
  - *o SVD truncado produz matrizes com um número específico de colunas, de modo a se conseguir reduzir a dimensionalidade.*

# Redução de dimensionalidade com *manifolds* – t-SNE

- Trata-se de um conjunto de algoritmos que permitem mapear os dados multidimensionais para um espaço com menos dimensões (normalmente 2 ou 3), com o objetivo de poderem ser facilmente visualizados graficamente
- Um desses algoritmos, particularmente interessante, e que tem ganho cada vez mais popularidade, é o t-SNE (*t-distributed stochastic neighbor embedding*)
  - *calcula uma nova representação dos dados de treino (apenas pode transformar os dados para os quais é treinado, não os de teste)*
  - *é útil para análise exploratória de dados, e raramente é usado em aprendizagem supervisionada*
  - *tenta encontrar uma representação bidimensional (ou tridimensional) dos dados que preserve as distâncias entre os pontos*
  - *começa com uma representação aleatória para cada pontos de dados (instâncias, observações do dataset) e vai depois tentado aproximar os pontos que estão próximos no espaço original e afastar os pontos que se encontrem mais distantes nesse espaço*
- É computacionalmente muito exigente, comparativamente com os outros métodos
  - *Por isso, quando o dataset atinge uma dimensionalidade elevada (acima de algumas dezenas de colunas) é recomendável que se comece primeiro pela redução PCA ou SVD*