Inteligência Computacional

Luís A. Alexandre

UBI

Ano lectivo 2019-20

Conteúdo

Objectivo da aprendizagem

- Quando nos é dado um conjunto e dados e treinamos o nosso classificador (a RN neste caso) nesses dados, temos de ter em atenção o seguinte: o nosso objectivo não é que o classificador tenha o menor erro possível nestes dados! É sim que ele tenha o menor erro possível em novos dados que provenham duma distribuição idêntica à que gerou estes dados.
- Para que o classificador tenha erro zero no conjunto de treino, basta usar uma tabela em que ele simplesmente guarde a classe de cada um dos pontos do conjunto de treino: decorar os dados.
- É óbvio que se o classificador for apenas uma tabela, não conseguirá dizer nada relativamente a um padrão nunca visto: não consegue generalizar.

Conjuntos de treino e de teste

- Como saber então qual o erro de generalização (o erro em dados nunca vistos que obedecem à mesma distribuição dos dados que possuímos)?
- Não é possível conhecer o verdadeiro erro de generalização: o que se faz é tentar estimá-lo.
- Para estimar o erro de generalização divide-se o conjunto de dados que possuímos em 2 subconjuntos disjuntos: o conjunto de treino e o de teste.
- O conjunto de treino é usado para treinar o classificador. O de teste para estimar o seu erro de generalização.

Erro de treino e de teste

- O erro medido em cada um dos conjuntos anteriores é chamado de erro de treino e de teste, respectivamente.
- O erro de treino é usado para guiar o processo de aprendizagem.
- O erro de teste é a nossa estimativa do erro de generalização.
- Quando iniciamos o treino, ambos os erros decrescem, mas após algum tempo, o erro de teste poderá começar a aumentar, mesmo que o erro de treino continue a diminuir: estamos perante o overfitting.

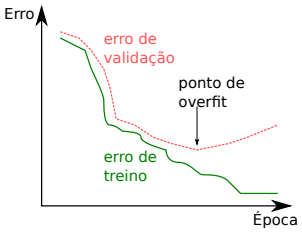
O que é o overfitting?

- O overfitting (em português talvez o melhor termo seja sobre-ajuste) acontece quando a RN memoriza os padrões de treino e perde assim a capacidade de generalizar.
- Quando existe overfitting a RN deixa de conseguir prever correctamente as saídas relativas a pontos que não se encontrem no conjunto de treino.
- O overfitting acontece quando a RN tem pesos em excesso relativamente aos que seriam necessários para aprender o problema em questão e o treino é efectuado durante muitas épocas.
- Nestas condições, a RN começa a memorizar a informação do conjunto de treino (inclusive eventual ruído que este possa conter), perdendo assim a capacidade de generalizar.

Como evitar o overfitting?

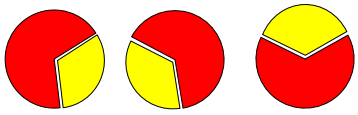
- O overfitting pode ser evitado de várias formas, recorrendo à **regularização**. Entre elas:
  - usando uma RN de dimensão apropriada ao problema (sem excesso de pesos);
  - parando o treino antes de o overfitting ocorrer (early-stopping);
  - fazendo alterações à função de custo (discutidas na aula 3).
- A primeira abordagem tem que ver com qual a arquitectura da rede mais adaptada ao problema em questão. Isto será discutido mais à frente.
- A segunda abordagem consiste em evitar que o treino se prolongue demasiadamente.
- Isto consegue-se usando um conjunto de validação. O conjunto de dados original passa a ser dividido em 3 subconjuntos disjuntos: treino, validação e teste.
- Muitas vezes o conjunto de validação é obtido efectuando uma divisão do conjunto de treino original.

Conjunto de validação



- A ideia é que durante o treino se vá estimando o erro de generalização no conjunto de validação: quando o erro neste conjunto aumentar, o treino deve parar.
- É usado o conjunto de validação pois durante o treino não se pode usar informação do conjunto de teste.

Validação cruzada



- Uma forma alternativa de obter estimativas do erro de generalização é através da validação cruzada (em inglês, cross-validation (CV)).
- Na figura acima, a roda completa representa todos os dados recolhidos do problema.
- A ideia é dividir o total dos dados em k dobras (folds) disjuntas (a amarelo) obtidas de forma aleatória: diz-se que se faz "k-fold CV".
- No exemplo temos $k = 3$. As regiões a vermelho passam a ser conjuntos de treino usados para construir o classificador.
- Para cada classificador construido, obtém-se uma estimativa do erro de generalização testando na região a amarelo.
- No final, faz-se a média das k estimativas para obter a estimativa final do erro de generalização.

Validação cruzada

- Uma vantagem da validação cruzada é podermos lidar com problemas com poucos dados que tornariam difícil construir conjuntos de treino e teste independentes.
- A validação cruzada pode ser apresentada indicando quantos pontos se deixa nos conjuntos de teste (amarelos).
- Nesses casos diz-se que se está a fazer "leave-p-out CV" onde p indica quantos pontos têm os conjuntos de teste.
- Idealmente faríamos $p = 1$, mas isso implicaria fazer m testes, onde m é o número total de pontos do conjunto, o que é muito pesado do ponto de vista computacional.
- Valores normais para o número de dobras são 5 ou 10.
- Existe ainda a possibilidade de repetir o processo de validação cruzada várias vezes e apresentar as médias e desvio padrão relativas às estimativas obtidas para o erro, sendo que em cada repetição, as dobras serão diferentes.

Amostras estratificadas

- A forma como se escolhe aleatóriamente os padrões em cada conjunto na validação cruzada pode ser puramente aleatória ou então estratificada.
- Consideremos que estamos a lidar com um problema de classificação com duas classes: a relação entre o número de padrões de uma das classes e a outra tem um determinado valor para o conjunto de dados. Por ex., podemos ter igual número de padrões das duas classes ou ter 10% de uma classe e 90% da outra.
- Quando a forma de divisão dos dados nos conjuntos é puramente aleatória, podemos ter casos em que a proporção de padrões das duas classes varia entre cada experiência.
- Já uma amostragem estratificada garante que essa proporção é mantida em todas as experiências e é igual à proporção original presente nos dados.
- Uma amostragem estratificada irá dar uma estimativa mais realista do desempenho do classificador em termos do seu erro de generalização.

Qual a arquitectura?

- Como decidir qual a arquitectura de rede a usar?
- Como referimos na aula sobre classificação supervisionada com redes multicamada, redes com duas camadas escondidas são suficientes para qualquer problema. Normalmente usam-se redes com uma camada escondida e se estas forem incapazes de aprender o problema em causa então usam-se redes com duas camadas escondidas.
- De seguida temos a questão da camada de saída. Já vimos também que o número de neurónios nesta camada depende do problema a resolver: normalmente usamos um neurónio por cada classe do problema; se for um problema de regressão tipicamente só temos um neurónio na camada de saída.
- Fica a faltar decidir quantos neurónios colocar na camada escondida.

Número de neurónios na camada escondida

- Não existe qualquer forma inequívoca de determinar quantos neurónios devem ser usados na(s) camada(s) escondida(s).
- Vamos no entanto referir duas abordagens que permitem chegar a um número aceitável.
- A primeira é a abordagem construtiva: começa-se com um número pequeno (pode ser inclusive 1) de neurónios na camada escondida e esse número vai sendo progressivamente aumentado até se verificar que o erro num conjunto de validação não decresce.
- A segunda é uma abordagem inversa à anterior (pruning em inglês): começa-se com um número elevado de neurónios e vão-se removendo até que o erro de validação pare de decrescer.

Abordagem construtiva versus pruning

- A abordagem construtiva tem algumas vantagens relativamente ao pruning.
  - É muito fácil especificar um valor inicial para o número de neurónios enquanto no caso do pruning é difícil saber à partida qual o número de neurónios que será usado.
  - A abordagem construtiva é computacionalmente mais atraente visto que começa com redes menores, ao passo que no pruning passamos a maior parte do tempo a treinar redes maiores que o necessário.

Neuroevolução

- Recentemente existe a capacidade de efetuar uma pesquisa pela melhor arquitetura, não só considerando os MLPs mas qualquer tipo de rede, incluindo as abordagens deep learning.
- Embora esta ideia seja antiga (mais de 20 anos) só recentemente é que o poder computacional permite usá-la de forma prática.
- A ideia é a de cruzar a computação evolucionária com as redes neuronais para que se faça evoluir potenciais redes para resolver um dado problema, até que se encontre uma arquitetura satisfatória.
- O principal problema continua a ser o custo computacional desta pesquisa, que nalguns métodos recentes exige milhares de horas de tempo de processamento para encontrar uma solução.

Como inicializar os pesos?

- Os métodos de optimização baseados na descida do gradiente são muito sensíveis aos valores iniciais dos pesos.
- Se a posição inicial for próxima de um mínimo a convergência é rápida.
- Se a posição inicial for numa região plana ou com pouco declive a convergência vai ser muito lenta.
- Uma boa estratégia será usar pesos num intervalo pequeno centrado em zero.
- É sugerido na literatura, que se usem pesos no intervalo $[-1/a, 1/a]$ onde a é o número de características do problema.

Codificação das saídas da rede

- Foi referido anteriormente que se tivermos um problema com L classes devemos usar L neurónios na camada de saída.
- Isto implica que quando um padrão é submetido à rede ela vai produzir na saída um vector y com L componentes.
- Como mapear este vector para as L classes?
- A solução normal é usar uma codificação chamada 1-em-L: das L possíveis saída apenas uma deve estar com valor 1 e as restantes com valor 0.
- Assim, se L for 3, teríamos as seguintes codificações paras as 3 classes do problema:
  classe 0: [1 0 0]; classe 1: [0 1 0] e classe 2: [0 0 1].
- De notar que nos conjuntos de dados as classes aparecem como valores inteiros: $\{0, 1, 2\}$ que terão que ser convertidos nos vectores acima para se poder efectuar o treino e o teste da rede.

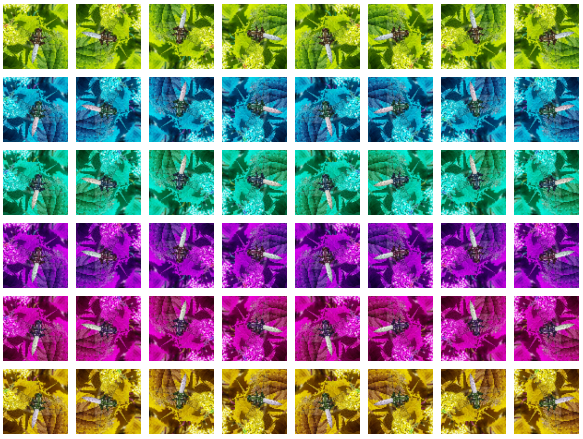
Aumento do número de dados

- As redes aprendem a partir dos dados do conjunto de treino.
- Assim, uma forma de lhes proporcionar mais informação para poderem aprender passa pelo uso de conjuntos de treino grandes.
- Na área do deep learning, em que os modelos têm milhões de pesos, são necessários muitos exemplos para os conseguirmos treinar, isto porque os problemas que estes modelos aprendem são complexos.
- Infelizmente é muito difícil obter dados em grandes quantidades para a aprendizagem supervisionada, por vários motivos:
  - é normalmente necessário que um humano coloque a etiqueta da classe nos dados: fazê-lo em milhares ou milhões de exemplos é uma tarefa árdua;
  - para determinado tipo de problema pode não ser possível obter muitos milhares ou mais de exemplos.

Aumento do número de dados

- A solução passa então muitas vezes por criar novos dados a partir dos existentes, para aumentar o tamanho do conjunto de treino (fazer data augmentation).
- Este processo envolve a aplicação de um conjunto de transformações que, a partir de um exemplo do conjunto de treino, consegue gerar uma dezena ou até mais, exemplos.
- Como é possível executar este processo? Para o caso em que temos dados sob a forma de imagens algumas das técnicas são:
  - aplicar rotações à imagem;
  - aplicar alterações aleatórias às cores;
  - fazer flip horizontal e vertical à imagem;
  - fazer cortes de dimensão aleatória;
  - aplicar ruído aleatório sobre a imagem.

Aumento do número de dados: exemplo



Normalização

- As características do nosso problema podem provir de sensores diferentes e portanto dizerem respeito a diferentes grandezas.
- Essas grandezas é natural que sejam representadas por gamas de valores em diferentes escalas.
- Um exemplo dos pontos anteriores é o conjunto de dados usado para exemplo em aulas anteriores:
  - as características são o peso e a altura, logo uma provém de uma balança e a outra de uma régua;
  - o peso é representado por valores da ordem das dezenas (de quilos) enquanto que a altura tem valores da ordem das unidades (de metro)
- O facto de diferentes características possuirem valores em escalas diferentes leva a que a sua influência sobre as entradas da rede seja diferente: quanto maiores os valores maior será a sua influência sobre a rede.

Normalização

- Assim, no exemplo anterior, os pesos iriam ter uma influência maior que as alturas, tipicamente cerca de 30 vezes mais influência!
- Ora de acordo com a representação dos padrões no espaço de características podemos observar que a importância das mesma na determinação da verdadeira classe é equilibrada.
- Para evitar uma potencial resposta desequilibrada, o que se faz é normalizar as características para que todas estejam compreendidas na mesma gama de valores.
- A normalização é feita tipicamente de acordo com uma das seguintes possibilidades:
  - obrigar as características a terem valor médio zero e variância unitária
  - fazer uma simples mudança de escala linear para que os valores de cada característica se encontrem no intervalo $[-1, 1]$

Normalização

- Porque é que se escolhe a gama próxima de $[-1, 1]$ para a normalização e não outra qualquer?
- Porque assim podemos usar pesos de valores relativamente pequenos e mantermos o valor de s na região em que as funções de activação normalmente usadas não se encontram saturadas.
- Isto permite que se efectue a aprendizagem.

Valores em falta

- Muitas vezes nos dados que nos são fornecidos podem faltar alguns valores de características.
- Isto pode acontecer por vários motivos: o sensor que as captava avariou; a pessoas que anotava as características não as anotou; erros na transmissão de dados; etc.
- Possíveis soluções para o problema:
  - remover o padrão em que existem valores em falta
  - substituir o valor em falta por: a média dos valores no caso destes serem contínuos ou pelo valor mais frequente quando são discretos

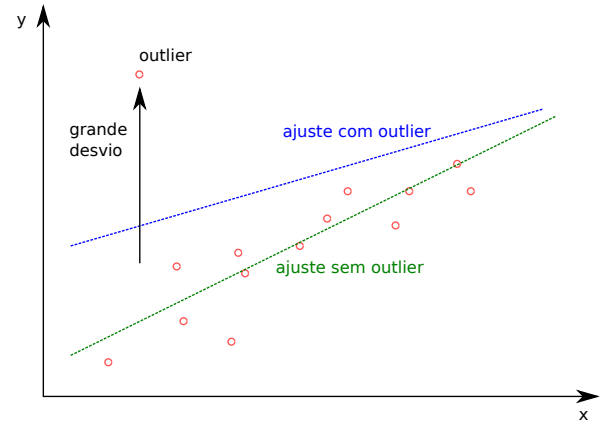
Codificação dos valores

- Uma RN só trabalha com valores numéricos.
- Quando temos características não numéricas, p.ex., uma cor que pode pertencer ao conjunto {amarelo, verde, azul, laranja}, o que fazer?
- A primeira ideia que surge é a de representar cada possível valor (neste caso, cada cor) por um número inteiro.
- Teríamos neste caso uma característica que corresponderia à cor e poderia tomar os seguintes valores: $\{0, 1, 2, 3\}$.
- Mas esta não é uma boa solução. A rede irá interpretar esta característica como podendo ser um contínuo e não um conjunto de valores discretos distintos.
- Assim, uma melhor solução será usar N características binárias (o N no exemplo acima vale 4), de forma que apenas uma delas valha 1 e as restante valham 0.

Outliers

- Um outlier é um padrão cujas características se desviam grandemente das apresentadas pelos restantes padrões.
- Os outliers podem existir nos dados por motivos variados: erros nas medições das características; erros na comunicação dos dados; má classificação; etc.
- O problema que eles introduzem na aprendizagem resulta de levarem a rede a efectuar ajustes nos pesos de grande amplitude, podendo alterar significativamente o resultado da aprendizagem.

Outliers



Outliers

- Podemos resolver o problema dos outliers de diversas formas, entre as quais:
  - remover os outliers antes do início do treino da rede: para os identificar usar técnicas estatísticas
  - modificar a função de erro para limitar o potencial efeito de erros muitos elevados
- A primeira abordagem tem a desvantagem de estar a remover informação que pode ser importante: o facto de alguns pontos serem aparentemente outliers pode ser uma característica dos próprios dados.
- A segunda abordagem consiste em alterar a função do erro, p. ex., o erro para um ponto p poderá ser obtido com

$$e_p = \begin{cases} (d_p - y_p)^2 & \text{se} \quad (d_p - y_p)^2 < \alpha \\ \alpha & \text{caso contrário} \end{cases}$$

onde α é um valor máximo a aceitar para o erro de um ponto.

Leitura recomendada

- Engelbrecht, sec. 7.1.1, 7.3.1, 7.3.2, 7.3.5.