AVII - Projeto 1

Alunos	Rodrigo Nogueira (46416), Sofia Condesso(50308)
Unidade Curricular	Ambientes Virtuais Interativos e Inteligentes
Professor	Arnaldo Abrantes

1. Modelos baseados em bigramas

Os modelos baseados em bigramas, são modelos que utilizam a informação das ocorrências de dois tokens adjacentes (caracteres ou palavras) no conjunto de treino, para gerar tokens, dada uma sequência de tokens, ou apenas um token.

1.1. Parâmetros estimados através de contagens:

À semelhança do que foi feito na aula prática, quando estimamos os parâmetros com base nas contagens necessitamos, primeiro, de obter as contagens de todos os bigramas presentes no corpus. No caso em que os bigramas são conjuntos de palavras, se o corpus for muito grande e variado, podemos ter que limitar o número de bigramas a construir. No caso dos bigramas em que os tokens são caracteres, o conjunto de bigramas obtido é de menor dimensão que o anterior, pois temos caracteres limitados e, por isso, as combinações não escalam tanto.
Para obter um modelo com parâmetros estimados através de contagens, basta calcular as probabilidades condicionadas de cada bigrama, e utilizar essa matriz de probabilidades, para prever o caractere seguinte.

Nível 1

Neste primeiro nível, utilizamos um conjunto de nomes em língua inglesa para treinar o nosso modelo baseado em bigramas. Primeiramente, foi feito um ciclo que dividia os diversos nomes, do conjunto, em carateres e organizava estes carateres em bigramas.
De seguida, foram criados dois dicionários (*stoi* e *itos*) e, com a ajuda destes dicionários, criou-se uma matriz "N" com a frequência de cada bigrama de caracteres.
Com o auxílio da matrix N, foram calculadas as distribuições de probabilidade condicionada de cada caratere.
Antes de terminar esta etapa, verificou-se se a distribuição de probabilidade foi feita corretamente, através da soma das probabilidades dos bigramas que começam em cada caratere. Como em cada caratere a soma deu 1, confirmou-se que a distribuição de probabilidade foi feita corretamente.
No fim deste modelo, foi usada a função *torch.multinomial* para gerar n amostras (cujos valores tem por base as probabilidades calculadas anteriormente) para obter a previsão que o modelo tem em relação aos carateres dados, chamados de contexto.

Nível 2

O segundo nível tem, como principal diferença, a utilização de um conjunto de dados diferente. Sendo este conjunto uma lista com nomes próprios em Portugal, implica a existência de alguns caracteres especiais como acentos e cedilhas. Alguma análise aos dados demonstrou que constam alguns nomes que se escrevem com hífens, apóstrofes e espaços brancos. Optou-se por manter estas carcterísticas, para preservar a integridade dos nomes, sendo apenas eliminados caracteres estranhos, caso estes se encontrem nas pontas extremas dos nomes. Outro processamento feito foi também a minusculização das palavras. O número total de carateres que passou de 27 (no nível 1) para **54**.
Também foi necessário uma atenção especial à extração dos nomes, pois os ficheiros .csv não tinham todos a mesma estrutura.

1.2. Parâmetros estimados através da otimização do critério da máxima verosimilhança:

Para estimar os parâmetros através da otimização do critério da máxima verosimilhança (*maximum likelihood*), partimos também das frequências dos bigramas nos dados de treino.

Nível 1

Neste primeiro nível utiliza-se um conjunto de nomes de língua inglesa para treinar o nosso modelo de bigramas com parâmetros estimados através da otimização do critério da máxima verosimilhança.

Começamos por criar um conjunto de treino e um conjunto de teste, que consistem em conjuntos de bigramas e para cada bigrama, guardou-se o índice do primeiro caractere como *input* e o indice do segundo caractere como um *target*.

De seguida, foi criada uma matriz de pesos (W), com valores aleatórios e gradientes já calculados, e fez-se uma multiplicação entre essa matriz e o conjunto de vetores criados através da função que codifica os *targets* em *one-hot encoding*, para obter um conjunto de *logits*. Estes *logits*, através de um método idêntico ao da função *softmax*, são transformados em probabilidades e, após a sua transformação, são usados para calcular a *cross-entropy loss*, que irá servir para optimizar os parâmetros do modelo.

Após calculada a *loss*, reiniciaram-se os gradientes da matriz W, para serem atualizados com os gradientes da *loss*, que foram calculados com respeito aos parâmetros do modelo e usando *back-propagation*.

Por fim, é atualizada a matriz W, com os gradientes calculados através da loss, usa a descida de gradiente com uma taxa de aprendizagem fixa de 10.

Nível 2

Neste nível, utilizou-se o mesmo modelo que o nível anterior e tal como no modelo de bigramas anterior, o conjunto de dados utilizado é o mesmo por isso em termos de diferenças também vai ao encontro do que foi dito no modelo anterior.

2. Modelo MLP com camada de Embedding

Enquanto que, na secção 1, apenas foram utilizadas as frequências relativas e o critério de máxima verosimilhança para estimar os parâmetros dos modelos que geram caracteres, dada uma sequência, nesta secção, a abordagem será mais complexa, inspirada no artigo "*A Neural Probabilistic Language Model*", utilizando uma rede neuronal com uma *embedding layer* e com, pelo menos, uma *hidden layer*, isto é, um *Multi Layer Perceptron*.
Cada camada (*layer*) serve um propósito específico no processamento do input e na geração de output. Segue uma breve descrição das camadas que constituem a rede neuronal a testar:

Camada de input (*input layer*) - É a primeira camada da rede neuronal e é responsável por receber os dados de entrada. Neste caso, os dados de entrada são sequências de N caracteres, representadas matrizes NxD, onde D é a cardinalidade do conjunto de caracteres distintos, nos dados de treino, e N é o número de caracteres no contexto. No código, esta camada é denominada por "C".

Camada de embedding (*embedding layer*) - Camada que se encarrega da representação numérica do contexto para o resto da rede. Iniciando os valores dos embeddings aleatoriamente e utilizando o mecanismo de retro-propagação do erro, as operações feitas no treino do modelo fazem com que as representações de cada carácter, que se formam nesta camada, sejam cada vez mais apuradas e que isso contribua para melhores resultados.

Camada(s) escondida(s) (*hidden layer(s)*) - Qualquer camada de uma rede neuronal excluindo as que têm funções de input ou de output. Uma camada de embedding também pode ser uma camada escondida, desde que não seja a camada de entrada.

Camada de output (*input layer*) - Camada responsável por mostrar as previsões da rede. Para problemas onde apenas pode ser prevista uma classe de cada vez, como é no nosso caso, é comum utilizar-se a função de ativação *softmax*, já para problemas onde, em cada inferência, o modelo pode prever mais do que uma classe, a função mais utilizada costuma ser a *sigmoid*.

Para elaborar o estudo e perceber qual a melhor arquitetura de *Multi-layer Perceptron* com uma camada de *embedding*, foi realizada uma procura de hiper-parâmetros ótimos, dentro do seguinte espaço de opções:

Contexto (nº de caracteres usados para prever o seguinte) = [2, 3, 4, 5]

Dimensão dos embeddings = [2, 3, 4, 5]

Dimensão da camada intermédia = [50, 100, 150]

Taxa de aprendizagem ("tamanho do passo" na descida de gradiente) = [0.5, 1, 5]

Para que a procura não seja exaustiva e computacionalmente cara, serão testados os valores para o tamanho do contexto, para a dimensão dos embeddings e para o número de neurónios da camada intermédia, primeiro, isoladamente e, depois, juntar as alterações que isoladamente tiveram mais sucesso. Todas as experiências são feitas com um treino de **300** épocas e uma taxa de aprendizagem fixa de **0.5**.
No Jupyter Notebook do trabalho, podem ser visualizados o relatório de classificação (classification_report do Sklearn), a matriz de confusão (confusion_matrix), os gráficos das curvas da loss no conjunto de treino (a azul) e no conjunto de validação (a laranja) e, também, um gráfico de pontos a representar os embeddings. Para efeitos de relatório e escolha do melhor modelo, serão visualizadas a loss e accuracy, sendo a decisão feita com base na loss.

Nível 1

Experiência 1.0
Experiência inicial, de controlo, com os valores recomendados no enunciado.

- Tamanho do contexto: 2
- Dimensão do espaço de "embedding": 2
- Dimensão da camada intermédia: 100
- Loss Validação: 2.7534
- Loss Teste: 2.7596
- Accuracy Teste: 0.195

2.1. Contexto (número de caracteres usados para predizer o seguinte):

Experiência 2.0
Ao aumentar o tamanho do contexto, verificamos que a accuracy aumenta ligeiramente, mas a loss piora ligeiramente.

- Tamanho do contexto: 3
- Dimensão do espaço de "embedding": 2
- Dimensão da camada intermédia: 100
- Loss Validação: 2.8050
- Loss Teste: 2.8264
- Accuracy Teste: 0.1995

Experiência 2.1
Ao aumentar ainda mais o tamanho do contexto (para 4), verificamos que a accuracy também aumenta, e a loss também melhora (diminui em relação à experiência 1.0).

- Tamanho do contexto: 4
- Dimensão do espaço de "embedding": 2
- Dimensão da camada intermédia: 100
- Loss Validação: 2.7357
- Loss Teste: 2.7349
- Accuracy Teste: 0.2211

2.2. Dimensão do espaço de "embedding":

Experiência 3.0
Ao aumentar a dimensão do espaço de embeddings, mantendo os restantes valores iguais à experiência de controlo, a loss piora, apesar de a accuracy aumentar.

- Tamanho do contexto: 2
- Dimensão do espaço de "embedding": 3
- Dimensão da camada intermédia: 100
- Loss Validação: 2.8205
- Loss Teste: 2.8267
- Accuracy Teste: 0.2129

Experiência 3.1
Ao aumentar ainda mais a dimensão do espaço de embedding (para 4), verificamos que não há melhorias.

- Tamanho do contexto: 2
- Dimensão do espaço de "embedding": 4
- Dimensão da camada intermédia: 100
- Loss Validação: 2.9005
- Loss Teste: 2.9148
- Accuracy Teste: 0.1962

2.3. Dimensão da camada intermédia de processamento:

Experiência 4.0
Ao diminuir, para metade, a dimensão da camada intermédia de processamento, mantendo os restantes valores iguais à experiência de controlo, a loss melhora bastante, sendo a experiência com melhor loss até agora. Esta melhoria faz sentido, pois dado que a dimensão do espaço de embedding não é muito grande, não existem assim tantas características possíveis de reter, pelo que não necessitamos de tantos neurónios na camada intermédia.

- Tamanho do contexto: 2
- Dimensão do espaço de "embedding": 2
- Dimensão da camada intermédia: 50
- Loss Validação: 2.6958
- Loss Teste: 2.6985
- Accuracy Teste: 0.2119

2.4. Experiências com várias alterações:

Experiência 5.0
Ao juntar duas das melhorias, não se obteve um melhor resultado do que na experiência 2.1

- Tamanho do contexto: 4
- Dimensão do espaço de "embedding": 2
- Dimensão da camada intermédia: 50
- Loss Validação: 2.8461
- Loss Teste: 2.6843
- Accuracy Teste: 0.2015

Nível 2

Para os dados do nível 2, foram feitas experiências semelhantes, à exceção da última experiência. **Experiência 1.0** Experiência inicial, de controlo, com os valores recomendados no enunciado.

- Tamanho do contexto: 2
- Dimensão do espaço de "embedding": 2
- Dimensão da camada intermédia: 100
- Loss Validação: 2.8461
- Loss Teste: 2.9435
- Accuracy Teste: 0.1749

2.1. Contexto (número de caracteres usados para predizer o seguinte):

Experiência 2.0
Ao aumentar o tamanho do contexto, verificamos que os resultados pioram bastante.

- Tamanho do contexto: 3
- Dimensão do espaço de "embedding": 2
- Dimensão da camada intermédia: 100
- Loss Validação: 3.2063

- Loss Teste: 3.3990
- Accuracy Teste: 0.1297

Experiência 2.1
Ao aumentar ainda mais o tamanho do contexto (para 4), obtêm-se melhores resultados do que na experiência 2.0, mas piores do que na experiência de controlo.

- Tamanho do contexto: 4
- Dimensão do espaço de "embedding": 2
- Dimensão da camada intermédia: 100
- Loss Validação: 2.8036
- Loss Teste: 2.9556
- Accuracy Teste: 0.1472

2.2. Dimensão do espaço de "embedding":

Experiência 3.0
Ao aumentar a dimensão do espaço de embeddings, mantendo os restantes valores iguais à experiência de controlo, a loss piora, apesar de a accuracy aumentar.

- Tamanho do contexto: 2
- Dimensão do espaço de "embedding": 3
- Dimensão da camada intermédia: 100
- Loss Validação: 2.9251
- Loss Teste: 3.1344
- Accuracy Teste: 0.1546

Experiência 3.1
Ao aumentar ainda mais a dimensão do espaço de embedding (para 4), verificamos que se obtém o melhor resultado de loss no conjunto de validação, até agora.

- Tamanho do contexto: 2
- Dimensão do espaço de "embedding": 4
- Dimensão da camada intermédia: 100
- Loss Validação: 2.6813
- Loss Teste: 2.8402
- Accuracy Teste: 0.2463

2.3. Dimensão da camada intermédia de processamento:

Experiência 4.0
Ao diminuir, para metade, a dimensão da camada intermédia de processamento, mantendo os restantes valores iguais à experiência de controlo, os resultados são muito parecidos.

- Tamanho do contexto: 2
- Dimensão do espaço de "embedding": 2
- Dimensão da camada intermédia: 50
- Loss Validação: 2.8262
- Loss Teste: 2.8963
- Accuracy Teste: 0.1757

2.4. Experiências com várias alterações:

Experiência 5.0
Ao juntar as duas melhores alterações (experiências 3.1 e 4.0), obtém-se uma loss de validação melhor do que a que se obteve na experiência 3.1.

- Tamanho do contexto: 2
- Dimensão do espaço de "embedding": 4
- Dimensão da camada intermédia: 50
- Loss Validação: 2.6332
- Loss Teste: 2.8020
- Accuracy Teste: 0.2419

Bibliografia

Funções de ativação: definição, características, e quando usar cada uma
A Neural Probabilistic Language Model