IA048 - Aprendizado de Máquina: Prova 28/05/2024

Tiago Corrêa de Araújo de Amorim (RA: 100675)

1. Questão 1

As redes neurais generativas adversárias (GANs, generative adversarial networks) trouxeram uma abordagem inovadora para a área de aprendizado de máquina.

(a) (0,8) Explique por que as redes geradora e discriminadora são consideradas adversárias.
(b) (0,8) Tendo em vista os papeis desempenhados por cada rede, discorra sobre a função custo proposta por Goodfellow et al. (2014)[1] para o treinamento da GAN. Complementando essa análise, comente também sobre como a rede geradora consegue aprender as características típicas dos dados reais.

*1.1. Resposta **(a)***

As redes geradora e discriminadora são consideradas adversárias porque o treinamento dos seus pesos tem objetivos conflitantes. A função objetivo da rede discriminadora é maximizar o número de vezes em que classifica os dados reais como verdadeiros e os dados gerados pela rede geradora como falsos. Já a rede geradora tem como função objetivo maximizar o número de vezes em que a rede discriminadora classifica as saídas da rede geradora como verdadeiras. É este jogo de gato e rato que faz com que a rede geradora aprenda a gerar amostras cada vez mais parecidas com as amostras reais, mesmo sem explicitamente definir o que é uma amostra parecida com uma amostra real.

*1.2. Resposta **(b)***

A função custo proposta por Goodfellow segue o que foi discutido no item anterior. Os pesos da rede discriminadora são ajustados para minimizar a entropia cruzada desta rede em função dos dados reais (classe *verdadeira*) e das saídas da rede geradora (classe *falsa*). Os pesos da rede geradora são ajustados para maximizar a entropia cruzada na saída da rede discriminadora, em função das saídas da rede discriminadora. Nesta etapa o objetivo é a rede geradora *enganar* a rede discriminadora, para que classifique as saídas da geradora na classe *verdadeira*. Como os objetidos são conflitantes, os autores propõe ajustar os pesos de cada rede de forma intercalada, ou seja, a redes não são ajustadas simultaneamente. Ademais, também comentam que existe um balanço delicado e é preciso evitar treinar *demais* a rede discriminadora, a ponto de deixar a tarefa da rede geradora *muito difícil* e não conseguir convergência.

Na proposta original a rede geradora recebe como entrada um vetor aleatório. O que se busca é que a rede geradora consiga mapear este vetor aleatório no espaço de distribuição dos dados reais. Uma rede geradora competente consegue aproximar o espaço de distribuição dos dados reais de forma a se tornar quase indistinguível do espaço real. Ao conseguir se tornar uma rede competente, a rede geradora estará efetivamente utilizando o vetor de entrada como uma *codificação* das principais características dos dados reais.

[1] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., Generative Adversarial Networks, Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS), pp.2672-2680, 2014.

2. Questão 2

(1,2) Explique os conceitos de mapa de características (feature map), neurônio, campo receptivo e compartilhamento de pesos no contexto de uma camada convolucional de uma CNN (convolutional neural network).

2.1. Resposta

- **Mapa de características**: Uma rede com camadas convolucionais funciona como uma série de filtros não-lineares em cascata (no caso de mais de uma camada convolucional na rede). O que se entende como mapa de características são os resultados gerados por cada uma das camadas convolucionais. O ajuste dos pesos dos filtros objetivam encontrar características nos dados de entrada que sejam úteis para o problema posto.
- **Neurônio**: O neurônio é a unidade básica de compução de uma rede neural. Em uma camada densa cada neurônio se conecta a todos os neurônios da camada anterior. Em uma camada convolucional o neurônio se conecta a um número limitado de neurônios da camada anterior, e depois *passeia* pelos neurônios (operação de convolução), efetivamente se conectanto a todos os neurônios da camada anterior. Existem situações em que nem todos os neurônios são *vistos* pela camada convolucional, dependendo dos valores de tamanho do filtro (*kernel*) e do passo (*stride*) utilizados. Cada *canal* na saída de uma camada convolucional é o resultado da aplicação de um mesmo neurônio seguidamente a porções dos neurônios da camada anterior.
- **Campo receptivo**: Uma camada densa trabalha *observando* toda a informação de uma única vez, enquanto que uma camada convolucional *observa* os neurônios da camada anterior em pequenas porções. Cada saída da camada convolucional é o resultado da aplicação do filtro em uma porção limitada dos dados. Ao ter camadas convolucionais seguidas o efeito é que cada saída das camadas mais profundas são o resultado da composição destas regiões em cascata. O campo receptivo de uma camada é o resultado desta composição de *lentes* de volta para o dado de entrada, onde as saídas de camadas mais profundas são o resultado de regiões cada vez maiores dos dados de entrada.
- **Compartilhamento de pesos**: Uma camada convolucional pode ser entendida como uma camada densa em que os pesos são compartilhados entre os diferentes neurônios. As operações envolvidas são as mesmas de uma camada densa, mas é como se cada neurônio tivesse pesos ajustáveis (e não-nulos) apenas para as conexões a um número limitado de dados de entrada. Além disso os pesos utilizados são os mesmos entre estes diferentes neurônios. Este mecanismo ajuda a entender a grande diferença entre o número de pesos ajustáveis de camadas densas e camadas convolucionais.

3. Questão 3

Tendo em mente a teoria subjacente às máquinas de vetores-suporte (SVMs, do inglês support-vector machines):

(a) (0,5) Defina margem de classificação no contexto de um problema linearmente separável e explique por que a maximização da margem é, intuitivamente, uma abordagem segura de projeto.
(b) (0,5) Por que a formulação matemática relacionada à obtenção do classificador de máxima margem possui similaridades com as ideias de regularização (e.g. Tikhonov / ridge regression) vistas no curso?

*3.1. Resposta **(a)***

A marge de classificação é a mínima *distância* entre qualquer classe e o hiperplano que separa estas classes. Ao maximixar a margem de classificação de um problema linearmente separável estamos efetivamente maximizando a mínima *distância* do hiperplano classificador para qualquer uma das classes. Como não temos garantias dos dados de treino serem suficientes para definir exatamente a região correspondente a cada classe, novos dados (de teste) podem se posicionar na região *entre* as classes (mais próximas do hiperplano classificador). Maximizar a margem leva a uma maior chance deste dados serem classificados corretamente (se posicionando no *lado correto* do hiperplano que separa as classes).

3.2. Resposta ***(b)***

A ideia da regularização é de reduzir a flexibilidade do modelo utilizado, fazendo uma troca entre viés e variância. O objetivo é que um modelo mais *bem comportado* (menos flexível) consiga generalizar melhor que um modelo mais flexível, e que possa sofrer menos com *overfitting*.

Entre duas classes linearmente separáveis é possível definir infinitos hiperplanos classificadores. A ideia do SVM é de reduzir a liberdade deste hisperplano, usando como critério a máxima margem, com o intuito de ter ter um classificador que generalize melhor. Um classificador de máxima margem se apoia nos dados mais *difíceis* de serem classificados (mais próximos da outra classe), de forma que aumenta a chance de que novos dados que venham a se aproximar ainda mais da outra classe (mais *difícieis* de classificar que os dados de treinados) sejam corretamente classificados, ou seja, gerando um classificador mais generalista.

4. Questão 4

(1,2) Numa importante conferência da área de inteligência computacional, foi lançada uma competição no âmbito de um problema de classificação de imagens. Duas jovens pesquisadoras decidiram, então, empregar uma CNN já consolidada na literatura (e.g. uma ResNet) para resolver a tarefa. Partindo de uma inicialização aleatória para os pesos da rede escolhida, e utilizando apenas os dados disponíveis na competição, elas, ao final do treinamento, observaram um nível de acurácia bastante adequado. No entanto, quando exposta a novos padrões de entrada (conjunto de teste), esta rede não atingiu um bom desempenho, tendo ficado muito abaixo das expectativas iniciais de projeto. Recomende (com argumentos bem fundamentados) duas estratégias para amenizar este problema.

4.1. Resposta

Uma possibilidade é que a base de dados utilizada pelas pesquisadoras não tenha tamanho adequado para o treinamento de uma rede com muitos pesos ajustáveis. Redes como a ResNet são muito profundas, com um número significativo de pesos a serem ajustados. Estas redes funcionam bem porque foram ajustadas com um número **muito** grande de dados. Uma forma de buscar contornar o problema do limitado número de dados é utilizar *data augmentation*, gerandos amostras adicionais a partir dos próprio dados (rotação, translação, ...). Ainda assim a quantidade de dados pode ser insuficiente para uma rede muito grande. Pode ser mais viável associar o *data augmentation* à utilização de apenas parte da rede profunda conhecida, de forma a ter uma melhor relação entre a quantidade de amostras (para treino, validação e teste) e o número de pesos ajustáveis.

Uma estratégia mais interessante é fazer *transfer learning*, ou seja, utilizar os pesos da rede já treinados e adaptar a estrutura da rede para o problema específico. Em diferentes publicações foi explorada a estrutura destas redes *famosas*, verificando que as camadas vão se especializando em elementos cada vez mais complexos à medida que a informação segue pela rede. As primeiras camadas se especializam em formas simples (semi-círculo, linha vertical, ...), e camadas profundas em elementos mais complexos (olho, boca, ...). Boa parte destas atividades podem ser úteis para outros problemas de classificação. Assim é possível utilizar boa parte das primeiras camadas da rede existente, incluir algumas novas camadas e fazer o treinamento dos pesos apenas das novas camadas que foram adicionadas ao final. Os pesos das últimas camadas da rede original também podem ser ajustados, mas partindo dos valores originalmente ajustados.

5. Questão 5

(1,4) Considere o problema de identificação de tumores cerebrais a partir de um conjunto de 1500 imagens de ressonância magnética (MRI) do crânio, de dimensão 200 x 200, em tons de cinza (veja um exemplo na Figura 1). As possíveis classes do problema são: (0) glioma; (1) meningioma; (2) pituitário; (3) saudável (normal). Descreva detalhadamente como uma rede MLP (multilayer perceptron), com uma única camada intermediária, poderia ser aplicada a este problema, indicando aspectos referentes à arquitetura da rede, bem como à metodologia como um todo (e.g. tratamento dos dados, treinamento do modelo etc.).

5.1. Resposta

Tratamento dos Dados

Os dados de entrada são imagens 200 x 200 em escala de cinza, logo podem ser representadas por uma matriz de 200 por 200 de números reais (ou inteiros). Uma operação opcional, mas que usualmente tem um efeito positivo é fazer a normalização dos dados de entrada. Uma opção é normalizar os valores entre 0 e 1.

Uma etapa opcional é incrementar a base de dados através do processo de *data augmentation*. Novas amostras podem ser geradas a partir dos dados existentes. Variações das imagens originais podem ser geradas aplicando operações de rotação, translação, zoom, inversão (*flip*), entre outras. As modificações apropriadas dependem das imagens da base de dados (eg.: girar em 180 graus uma imagem transversal de ressonância magnética pode não fazer sentido, pois todas tem a cabeça virada para a parte superior da imagem).

Os dados das classes precisam ser codificados em um vetor *one-hot* de tamanho 4 (número de classes), em que cada posição está associada a uma das classes. Por exemplo, o vetor associado a uma amostra classificada na classe glioma tem a forma $[1, 0, 0, 0]^T$.

Para utilizar estes dados de imagens em uma rede MLP temos primeiro que redimensionar cada matriz para um vetor (*flatten*). Neste exemplo teremos vetores de 40 000 números reais.

Arquitetura da Rede

A camada de entrada da rede recebe os 40 000 valores de entrada de cada amostra. Cada neurônio da camada de entrada é associado a uma das posições do vetor de entrada, e recebe o respectivo valor da amostra. Logo, temos 40 000 neurônios. Não há pesos a serem ajustados nesta camada.

A camada seguinte é a camada intermediária, onde teremos n neurônios. Nesta camada densa cada neurônio estará conectado a todos os neurônios da camada anterior. Desta forma esta camada tem tem $40\,001n$ pesos a serem ajustados (para cada neurônio são 40 000 pesos que multiplicam os valores dos neurônios da camada de entrada e um peso associado ao *bias*). Aos resultados destas somas ponderadas em cada neurônio é usualmente aplicada uma função não-linear (*relu*, *sigmóide*, *tanh* etc.). Se não for aplicada uma função não-linear, a rede terá uma menor capacidade de resolver o problema proposto, pois só conseguirá encontrar relações lineares.

A camada de saída é composta por 4 neurônios, um associado a cada classe. Cada neurônio se conecta a todos os n neurônios da camada intermediária. Desta forma esta camada tem $4(n + 1)$ pesos ajustáveis. Ao resultado da soma ponderada é aplicada a função *softmax*, que irá dar uma ideia de probabilidade associada a cada classe.

Hiperparâmetros como o número de neurônios da camada intermediária e a função de ativação podem ser otimizados com uma busca em grade (*grid search*). Uma busca em grade nada mais é que uma busca exaustiva de todas as combinações destes hiperparâmetros, verificando que combinação gera os melhores resultados ao final do treinamento.

Treinamento do Modelo

Para o treinamento do modelo é uma boa prática separar os dados disponíveis entre treino, validação e teste. Os dados de treino serão efetivamente utilizados para controlar o processo de ajuste dos pesos da rede. Os dados de validação servem para acompanhar o treinamento da rede e evitar *overfitting*, mas não são utilizados para guiar o algoritmo de ajuste. Os dados de teste são apresentados ao modelo apenas no final do processo de ajuste, servindo de indicador da qualidade da rede.

Para o ajuste dos pesos são usualmente utilizados algoritmos baseados no gradiente da função de perda. Este algoritmos utilizam o gradiente para indicar a direção em que devem ser modificados os valores dos pesos de forma a minimizar a função de perda. A direção de minimização da função de perda é a direção contrária ao gradiente. O passo a ser dado pelo algoritmo de otimização é usualmente definido pelo usuário. Algoritmos de segunda ordem (com cálculo, ou aproximação, da Hessiana) conseguem uma estimativa da direção e do passo a ser dado, mas são menos comuns.

O gradiente é calculado por retro-propagação (*backpropagation*) das derivadas. Uma amostra passa pela rede, fazendos os cálculos até o final. A retro-propagação começa do final do cálculo e vai retornando até o início da rede, calculando as derivadas associadas a cada peso da rede.

Para problemas de classificação a função de perda utilizada é normalmente a entropia cruzada. A minimização da entropia cruzada equivale a maximizar a verossimilhança (maximiza as probabilidades associadas à classe correta de cada amostra dos dados de treino).

A estrutura geral dos algoritmos de otimização utilizados é basicamente a mesma. É apresentado um conjunto de dados de treino (o número de amostras por *batch* pode variar de uma a todas as amostras) e a média dos gradientes associados a cada amostra define a direção de alteração dos pesos. O tamanho do passo é informado pelo usuário (diretamente ou indiretamente, a depender do algoritmo utilizado). Após todos os dados de treino terem sido apresentados à rede, considera-se que acabou esta **época**. Ao final da época é feita uma avaliação da rede com os dados de validação. O processo de ajuste dos pesos é repetido até que algum critério de parada é atingido (número máximo de épocas, valor mínimo da função de perda, valor mínimo da redução da função de perda entre épocas, não melhora da função de perda com os dados de validação após um determinado número de épocas etc.).

Ao final do treino são aplicados os dados de teste à rede ajustada, e as métricas de interesse (*f1-score*, acurácia balanceada etc.) são calculadas e avaliadas pelo usuário.

6. Questão 6

Ao realizar um processo de PCA sobre uma base de dados, um pesquisador obteve os seguintes autovalores para a matriz de autocorrelação:

$$\lambda = \begin{bmatrix} 0,2 \\ 2,3 \\ 1,5 \\ 3 \\ 0,05 \\ 0,15 \\ 0,5 \\ 0,3 \end{bmatrix} \tag{6.1}$$

(a) (0,3) Qual é a dimensão do espaço original dos dados?

(b) (0,7) Caso se busque uma preservação de ao menos 90% do conteúdo energético dos dados, qual é o menor número possível de componentes principais empregadas? Justifique.

*6.1. Resposta **(a)***

Assumindo uma matriz de autocorrelação de posto completo, e dado que a matriz de autocorrelação é aproximada pela média dos produtos externos dos dados de entrada, o número de autovalores coincide com a dimensão do espaço original: 8. O caso mais geral (sem assumir uma matriz de autocorrelação de posto completo) é que a dimensão do espaço original é ao menos 8.

*6.2. Resposta **(b)***

A parcela de variância associada a cada componente principal é dada pela razão entre o valor do autovalor associado e a soma de todos os autovalores. Para preservar ao menos 90% da variância observada nos dados (conteúdo energético) precisamos usar ao menos os 4 componentes principais associados aos 4 maiores autovalores.

i	λ_i	$\sum_{j=1}^{i} \lambda_j$	$\frac{\sum_{j=1}^{i} \lambda_j}{\sum_{j=1}^{8} \lambda_j}$
1	3.00	3.00	0.375
2	2.30	5.30	0.663
3	1.50	6.80	0.850
4	0.50	7.30	0.913
5	0.30	7.60	0.950
6	0.20	7.80	0.975
7	0.15	7.95	0.994
8	0.05	8.00	1.000

7. Questão 7

Considere que uma CNN tenha sido aplicada à classificação de imagens de retina em três classes: normal, glaucoma e catarata. Na etapa de teste, a seguinte matriz de confusão foi obtida:

	Classe estimada		
Classe real	**Normal**	**Glaucoma**	**Catarata**
Normal	410	15	25
Glaucoma	10	100	40
Catarata	55	25	100

(a) (0,4) Considerando a classe **Glaucoma** como positiva, obtenha as quantidades de verdadeiros positivos (TP), falsos positivos (FP), verdadeiros negativos (TN) e falsos negativos (FN).
(b) (0,6) Determine o valor da acurácia balanceada atingida pela CNN, mostrando explicitamente os valores das métricas intermediárias necessárias para o cálculo.

*7.1. Resposta **(a)***

Para encontrar os valores considerando Glaucoma como classe positiva temos que agregar os valores das demais classes:

1. O número de verdadeiros positivos permanece o mesmo.
2. A coluna de classe positiva estimada tem o mesmo total que a coluna de Glaucoma estimado da tabela anterior.
3. A linha de classe positiva real tem o mesmo total que a linha de Glaucoma real da tabela anterior.
4. O número de negativos verdadeiros é a soma dos termos fora da linha ou coluna Glaucoma da tabela anterior.

	Classe estimada	
Classe real	**Positiva**	**Negativa**
Positiva	TP=100	FN=50
Negativa	FP=40	TN=590

*7.2. Resposta **(b)***

A acurácia balanceada é a média do *recall* para cada classe. E *recall* é a taxa de positivos verdadeiros: $TPR = \frac{TP}{TP+FN}$. O cálculo dos valores de TP e FN de cada classe seguem a lógica apresentada no item anterior. A acurácia balanceada da CNN é **0.711**.

Classe	**TP**	**FN**	***Recall***
Normal	410	40	0.911
Glaucoma	100	50	0.667
Catarata	100	80	0.556
		Média	0.711

8. Questão 8

(a) (1,0) Explique como funciona o mecanismo de auto-atenção de um transformer.
(b) (0,6) De que maneira um transformer consegue aproveitar possíveis interdependências de curto e longo prazo em uma sequência de entrada apesar de não ter recorrência?

8.1. Resposta ***(a)***

Dado um vetor de elementos, o mecanismo de auto-atenção de um transformer ajusta a posição de cada elemento no espaço latente em função de todos os elementos do vetor (que é uma representação de uma sequência). Os dados de entrada são os elementos em algum espaço latente (eg.: a localização de cada palavra, ou *token*, em um espaço semântico). É montada uma matriz de auto-correlação calculando o produto interno de todos os possíveis pares de elementos (inclusive de cada elemento com ele mesmo). Estes produtos internos dão uma ideia do grau de correlação entre os elementos. Cada elemento é *reescrito* como uma combinação linear convexa de todos os elementos ponderada pelos produtos internos calculados (os produtos internos passam por uma *softmax* antes de multiplicar os elementos). Esta transformação acaba por mudar a posição de cada elemento em função do seu grau de correlação com os demais elementos do vetor (eg.: muda o sentido semântico de cada palavra em função das palavras que compõem a frase), criando uma ideia de contexto para os elementos do vetor.

A este mecanismo podem ser adicionadas matrizes que *ajustam* as posições de cada um dos elementos do vetor antes em cada operação. Uma matriz pode multiplicar os elementos que entram no primeiro termo de cada produto interno (M_{query}), outra multiplica o elemento que entra no segundo termo do produto interno (M_{key}) e uma terceira multiplica cada elemento que será utilizado nas multiplicações da combinação convexa (M_{value}). Desta forma se dá mais flexibilidade à camada transformer para que busque representações alternativas da sequência (eg.: considerar uma frase como uma ironia ou como uma afirmação).

8.2. Resposta ***(b)***

O *transformer* é um mecanismo do tipo vec2vec, de forma que recebe como entrada um vetor de elementos. Ao utilizar vetores é perdida a informação de posição que uma sequência carrega. É preciso adicionar aos dados de entrada (já em algum espaço latente) um termo de codificação posicional, para que a rede consiga *entender* a posição relativa de cada elemento do vetor de entrada. Cada elemento recebe um código único que representa a sua posição dentro da sequência original, que foi transformada em vetor. Na proposta original dos *transformers* são utilizados os valores de uma família de senoides para codificar estas posições.