ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

Departamento de Engenharia de Computação e Sistemas Digitais

PCS 5022 - Redes Neurais e Aprendizado Profundo

Prof. Artur Jordão

1 Fundamentos Básicos de Machine Learning

1. O código abaixo apresenta um problema teórico grave. Qual? Como resolver? Crie um modelo preditivo (i.e., OLS, SVM, XGBoost) e analise o comportamento da performance preditiva do modelo antes e após resolver o problema.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

n_samples = 1000
n_features = 2
n_classes = 5
X = np.random.rand(n_samples, n_features)
y = np.random.randint(0, n_classes, size=n_samples)
y = np.eye(n_classes)[y]

mean = np.mean(X, axis=0)
std = np.std(X, axis=0)
X = (X - mean) / std

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33,
random_state=42)

model = ...
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

accuracy_score(y_test, y_pred)
```

Código 1: Exercício 1

2. Gere dados aleatórios com dimensão $n \times m$ para k categorias (classes) e $m = 2$. A partir dos dados gerados realize as seguintes tarefas. (i) Calcule a média das amostras; (ii) Calcule a média das amostras de cada classe; (iii) Para cada classe, identifique qual a amostra mais distante da amostra média da classe. Expresse formalmente a solução dos itens (i)-(iv).

3. Gere dados aleatórios com dimensão $n \times m$, com $m = 2$. A partir desses dados mostre o espaço de características, destacando a amostra média geral e a amostra média por classe.

4. Faça o mesmo que o exercício anterior, porém agora utilize $m > 4$. Como mostrar o espaço de características quando m é arbitrariamente grande?

5. Um problema envolvendo a solução analítica da regressão linear, $(X^T X)^{-1} X^T Y$, é que os dados de treinamento não podem apresentar mais features (m) do que amostras (n) – *the sample size problem*. Como resolver esse problema sem utilizar a otimização de *gradient descent*?

6. Suponha um cenário hipotético em que só podemos aprender um modelo usando um determinado número de amostras k, onde $k \ll n$. Como lidar com essa restrição evitando uma solução trivial de amostrar exemplos aleatoriamente?
   Leitura recomendada para o exercício: [12]

2 FeedForward Networks

1. Assuma dois valores inteiros n e k. A partir desses valores, crie uma rede MLP com n camadas, onde a i-ésima camada ($i \in \{1, 2, 3, \ldots, n\}$) possui k^i neurônios.

2. Uma similaridade entre projeções lineares (ex., predição de uma regressão linear ou transformação via *Principal Components Analysis* – PCA) e uma rede neural com somente uma camada é que podemos formular a predição como $xW^T + b$. Tecnicamente, podemos utilizar essa similaridade para modelar predições de modelos lineares como redes neurais. Como? Tente realizar esse processo para uma projeção linear simples (ex. PCA). Observe como a matriz de projeção é armazenada: matriz coluna – $k \times 1$ – ou matriz linha – $1 \times k$. Note também as possíveis transformações que podem ser realizadas nos dados antes de projetá-los. O código abaixo ilustra com utilizar o PCA disponível no sklearn.

```
from sklearn.decomposition import PCA
from sklearn.datasets import make_classification
import numpy as np

X, y = make_classification(n_samples=10000, n_features=3000...)

pca = PCA(n_components=2)
pca.fit(X)
X_latent = pca.transform(X)
w = pca.components_
...
```

Código 2: Exercício 2

3. Podemos dizer que modelos superparametrizados podem classificar dados com rótulos aleatórios. Desenvolva um setup experimental para demonstrar (mesmo que parcialmente) essa afirmação. Tal característica corresponde a uma propriedade positiva ou negativa desses modelos?
   Leitura recomendada para o exercício: [19, 13]

3 Basics Hyperparameters

1. Formule o grafo computacional das seguintes expressões: $y = 2 * (a - b) + c$; $y = x^T w + b$ e $y = (\frac{x-\mu}{\sigma}) * \gamma + \delta$.

2. A partir do grafo computacional abaixo, indique qual a derivada parcial de $\frac{\partial g}{\partial b}$ (isto é, como alterações em b afetam g).

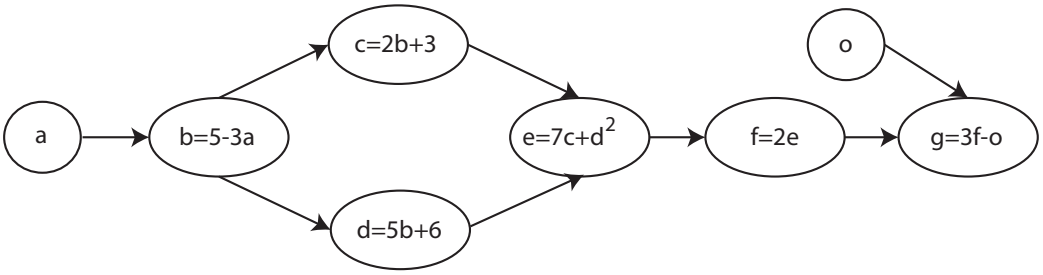

Figure 1: Grafo computacional para o Exercício 2

3. Proponha uma técnica *learning rate scheduler*.

4. Proponha uma função de ativação.
Leitura recomendada para o exercício: [17].

5. Proponha uma função de erro customizada. A função de erro pode ser uma variação de alguma função existente (ex. adicionando um termo de penalização) ou combinação de operadores matemáticos primitivos.
Leitura recomendada para o exercício: [10, 11].

6. Elabore uma técnica para explorar um espaço n dimensional de hiperparâmetros (ex. *grid search*). Seu espaço deve conter, pelo menos, os seguintes hiperparâmetros: (i) otimizador, (ii) inicialização e (iii) *learning rate* inicial. Em cada eixo do espaço (isto é, cada hiperparâmetro) deve existir pelo menos duas opções. Visualizando essa questão da perspectiva de *feature space*, seria possível predizer a performance preditiva do modelo para valores de hiperparâmetros não observáveis (dica: pense como seria a composição das matrizes X e Y)?

4 Deep Learning

1. Frequentemente em *Deep Learning*, a quantidade de parâmetros é um indicativo da capacidade de expressividade do modelo. A partir dessa observação, uma forma de controlar a capacidade de um modelo é aumentar o número de neurônios ou a quantidade de camadas (paradigma *we need to go deeper*). Assuma um valor inteiro n indicando a quantidade de

neurônios de duas camadas ocultas compondo uma rede neural. Construa outra arquitetura distribuindo esses n neurônios em k camadas. Por exemplo, a partir de uma rede com $n = 2048$ (largura $\{2048, 2048\}$) podemos criar outra com o mesmo número de neurônios usando $k = 4$ ($\{1024, 1024, 1024, 1024\}$) ou $k = 8$ ($\{512, 512, 512, 512, 512, 512, 512, 512\}$). Qual dos dois modelos é mais eficiente em termos de latência[1], armazenamento e número total de parâmetros?
Leitura recomendada para o exercício: [18, 6, 2]

2. Podemos entender a transformação aplicada por uma camada i como a representação interna dessa camada para os dados. A partir dessa descrição, como podemos mensurar a similaridade entre representações internas de duas camadas distintas da mesma rede? E de redes neurais diferentes? O código abaixo sumariza como obter a representação interna de uma camada i usando um modelo pré-definido.
Leitura recomendada para o exercício: [9, 15, 16, 3]

```
model_ = keras.models.Model(model.input,
model.get_layer(index=i).output)
internal_representation = model_.predict(X)
```

3. Combine a representação interna de diferentes camadas e aplique essa combinação ($X = \bigcup_{i=1}^{L} X_i$) a um classificador simples.

4. Considerando uma rede neural qualquer, elabore um experimento para ilustrar a separabilidade fornecida pela representação interna das diferentes camadas compondo o modelo.

5. De acordo com o trabalho de Zhang et al. [20], as camadas em modelos profundos podem ser categorizadas em robustas e críticas. Camadas robustas são aquelas que, após o treinamento, podem ser reinicializadas a sua inicialização original sem degradar (ou degradando marginalmente) a habilidade preditiva do modelo. Camadas críticas, por outro lado, são sensível a reinicialização. Elabore um setup experimental simples para demonstrar essa observação. Ao final, mostre quais foram as camadas críticas e as robustas. Observação importante: as conclusões de Zhang et al. [20] são aplicáveis às arquiteturas residuais que serão abordadas posteriormente no curso.
Leitura recomendada para o exercício: [20, 14]

5 Regularization

1. Construa uma arquitetura de rede neural simples e a regularize-a utilizando o mecanismo de ensemble. Defina formalmente a estratégia de ensemble adotada.

[1] Latência refere-se ao tempo que um modelo leva para predizer a resposta a partir de uma amostra ou conjunto de amostras (tempo de *foward*).

2. Períodos críticos correspondem a um fenômeno relacionado à efetividade da regularização na dinâmica do treinamento. De acordo com trabalhos anteriores, tal fenômeno ocorre nas épocas iniciais de treinamento. A partir de uma rede neural qualquer, elabore um setup experimental para inspecionar se períodos críticos emergem durante o treinamento dessa arquitetura.
   Leitura recomendada para o exercício: [5, 7, 8]

3. Conforme visto em aula, estudos mostraram a possibilidade de aprender somente os parâmetros das camadas de Batch Normalization. Elabore um treinamento para demonstrar a eficácia dessa estratégia (isto é, se um modelo treinado seguindo essa ideia obtém habilidade preditiva não trivial).
   Leitura recomendada para o exercício: [4, 1]

References

[1] Rebekka Burkholz. Batch normalization is sufficient for universal function approximation in CNNs. In *International Conference on Learning Representations (ICLR)*, 2024.

[2] Mostafa Dehghani, Yi Tay, Anurag Arnab, Lucas Beyer, and Ashish Vaswani. The efficiency misnomer. In *International Conference on Learning Representations (ICLR)*, 2022.

[3] Lyndon R. Duong, Jingyang Zhou, Josue Nassar, Jules Berman, Jeroen Olieslagers, and Alex H. Williams. Representational dissimilarity metric spaces for stochastic neural networks. In *International Conference on Learning Representations (ICLR)*, 2023.

[4] Jonathan Frankle, David J. Schwab, and Ari S. Morcos. Training batchnorm and only batchnorm: On the expressive power of random features in cnns. In *International Conference on Learning Representations (ICLR)*, 2021.

[5] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Time matters in regularizing deep networks: Weight decay and data augmentation affect early learning dynamics, matter little near convergence. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[6] Kai Han, Yunhe Wang, Qiulin Zhang, Wei Zhang, Chunjing Xu, and Tong Zhang. Model rubik's cube: Twisting resolution, depth and width for tinynets. In *Neural Information Processing Systems (NeurIPS)*, 2020.

[7] Michael Kleinman, Alessandro Achille, and Stefano Soatto. Critical learning periods for multisensory integration in deep networks. In *Conference on Computer Vision and Pattern Recognition (CVPR*, 2023.

[8] Michael Kleinman, Alessandro Achille, and Stefano Soatto. Critical learning periods emerge even in deep linear networks. In *International Conference on Learning Representations (ICLR)*, 2024.

[9] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning (ICML)*, 2019.

[10] Chuming Li, Xin Yuan, Chen Lin, Minghao Guo, Wei Wu, Junjie Yan, and Wanli Ouyang. AM-LFS: automl for loss function search. In *International Conference on Computer Vision (ICCV)*, 2019.

[11] Hao Li, Tianwen Fu, Jifeng Dai, Hongsheng Li, Gao Huang, and Xizhou Zhu. Autoloss-zero: Searching loss functions from scratch for generic tasks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[12] Sepideh Mahabadi and Stojan Trajanovski. Core-sets for fair and diverse data summarization. In *Neural Information Processing Systems (NeurIPS)*, 2023.

[13] Pratyush Maini, Michael Curtis Mozer, Hanie Sedghi, Zachary Chase Lipton, J. Zico Kolter, and Chiyuan Zhang. In *International Conference on Machine Learning (ICML)*, 2023.

[14] Wojciech Masarczyk, Mateusz Ostaszewski, Ehsan Imani, Razvan Pascanu, Piotr Miłoś, and Tomasz Trzcinski. The tunnel effect: Building data representations in deep neural networks. In *Neural Information Processing Systems (NeurIPS)*, 2023.

[15] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations (ICLR)*, 2021.

[16] Thao Nguyen, Maithra Raghu, and Simon Kornblith. On the origins of the block structure phenomenon in neural network representations. *Transactions on Machine Learning Research*, 2022.

[17] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. In *International Conference on Learning Representations (ICLR)*, 2018.

[18] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, 2019.

[19] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017.

[20] Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal? *Journal of Machine Learning Research*, 2022.