Name: _____ NetID: _____

S&DS 365 / 665

Intermediate Machine Learning

Final Exam (sample solution)

Wednesday, December 20, 2023

Complete all of the problems. You have 2.5 hours (150 minutes) to complete the exam.

The exam is closed book, computer, phone, etc. You are allowed one double-sided $8\frac{1}{2} \times 11$ sheet of paper with hand-written notes.

Please use a black pen or dark pencil so that your exam scans clearly.

You can use the back of pages for scratch work. Do not write answers on the back of the pages; they will not be scanned.

1. ***Short derivation or answer*** (30 points)

(1.1) Consider a data set with $p = 2$ dimensions and $n = 2$ data points with design matrix and response vector

$$\mathbb{X} = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \qquad Y = \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

What is the lasso estimator

$$\arg\min_{\beta} \frac{1}{2}\|Y - \mathbb{X}\beta\|^2 + \lambda\|\beta\|_1$$

at regularization levels $\lambda = 1$ and $\lambda = \frac{1}{2}$? Show your work.

Since the design is orthogonal the lasso solution is $\text{Soft}_{\lambda/2}(\frac{1}{2}X^TY)$:

$$\widehat{\beta} = \text{Soft}_{\frac{1}{2}}((0,-1)^T) = (0,-\tfrac{1}{2})^T$$
$$\widehat{\beta} = \text{Soft}_{\frac{1}{4}}((0,-1)^T) = (0,-\tfrac{3}{4})^T$$

(1.2) Suppose that an estimation method has tuning parameter t for which the squared bias scales as $O(t^{1/3})$ and the variance scales as $O\left(\frac{1}{\sqrt{nt}}\right)$ as $t \to 0$ with $nt \to \infty$, for a sample of size n. What is the fastest rate at which the risk (expected squared error) will decrease with sample size for this method? Show your work.

Balancing bias and variance we have $t \asymp n^{-3/5}$ so the risk decays as $n^{-1/5}$.

(1.3) A convolutional neural network is constructed to classify color images, each with 3 color channels (red, blue, green). The training images are 28×28 pixels in size. The network has two convolutional layers followed by two dense layers. No pooling operations are used.

The first convolutional layer uses 100 filters, each 2×2, constructed with a command such as `model.add(layers.Conv2D(100,(2,2)))`. How many trainable parameters does this layer have? Show your work.

2×2 filter parameters for each of 3 channels, plus an intercept, for each of 100 filters: $(2 \times 2 \times 3 + 1) \times 100 = 1300$.

The second convolutional layer also uses 100 filters, each 3×3, constructed with a command such as `model.add(layers.Conv2D(100,(3,3)))`. How many trainable parameters does this layer have? Show your work.

3×3 filter parameters for each of 100 channels, plus an intercept, for each of 100 filters: $(3 \times 3 \times 100 + 1) \times 100 = 90100$.

(1.4) Consider the generative model

$$Z \sim N(0,1)$$
$$X \mid z \sim N(f(z), I_d)$$

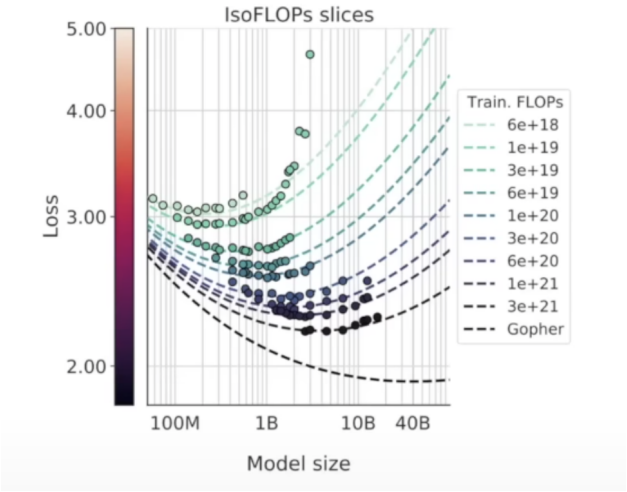
where $f : \mathbb{R} \to \mathbb{R}^d$. Suppose we approximate the posterior $p(Z \mid x)$ with the variational approximation $q(Z \mid x)$ which is Gaussian with mean $\mu(x)$ and variance $\sigma^2(x)$.

The evidence lower bound (ELBO) is a lower bound on $\log p(x)$ used for approximate inference, for example in variational autoencoders. Derive an expression for the ELBO in terms of the quantities $f(z)$, $\mu(x)$ $\sigma^2(x)$, and a sample $Z_1, \ldots, Z_S$ of standard Gaussian random variables.

You can give your answer up to a fixed additive constant. You may use the fact that the the entropy of a univariate Gaussian $N(\mu, \sigma^2)$ is $\frac{1}{2}\log(2\pi e\sigma^2)$.

$$\mathbb{E}_q \log p(x \mid Z) - D_{\text{KL}}(q,p) =$$
$$-\frac{1}{2S}\sum_{s=1}^{S} \|x - f\left(\mu(x) + \sigma^2(x)Z_s\right)\|^2 + \frac{1}{2}\log\left(2\pi e\sigma^2(x)\right) - \frac{1}{2}\left\{\sigma^2(x) + \mu^2(x)\right\}$$

(1.5) The following figure is from Karpathy's video "LLMs for Busy People" presented and discussed in the last two classes.



(1) Explain the meaning of the figure. What are the quantities on the x and y axes, and what are the lines and points?

The x-axis is model size in gigabytes. A 20B model would have 10 billion parameters, if the parameters are stored with 16-bit precision. The y-axis is the loss function on (presumably) the test data, which is a scaled version of the log-loss or perplexity. Each point is a large language model that was trained, and the iso-lines are models that have the same training time; this is a function of the model size and the number of passes through the data (epochs).

(2) What is the significance of this plot? Why is it important for the future development of LLMs and AI?

The consequence of this remarkable plot is that current models are nowhere near to "bottoming out." By building larger models and training them on more data with larger computers, significant gains are achievable, even with no advances in the underlying models and algorithms. Because loss (perplexity) is strongly correlated with performance of downstream applications, this will translate to improvements in performance, and more advanced "intelligence."

(1.6) Recall that a GRU (gated recurrent unit) is a type of recurrent neural network that uses two types of gates to move information in and out of memory.

Suppose that a GRU is constructed with an input/output vocabulary of V symbols using a single layer of H neurons. How many trainable parameters p does the GRU have? Assume the specific GRU architecture presented in class and on Assignment 5.

Express your answer as a multinomial $p(H, V)$ in H and V. For example, one answer might be $p(H, V) = 4H^3 + 3H^2V + 2HV + 1$. Explain your answer and describe the architecture you are assuming for full credit.

The "vanilla" RNN hidden layer and each gate have $H^2 + HV + H$ parameters. The linear transformation from the state to scores has $VH+V$ parameters. The total number of parameters is therefore:

$$3H^2 + 4HV + 3H + V$$

(1.7) Transformers are sequence-to-sequence models that transform an input sequence by a series of encoder blocks followed by a series of decoder blocks to generate the target sequence. The crucial ingredient is attention, which is used by the encoder and decoder blocks in different ways. The following is pseudocode for the transformer architecture.

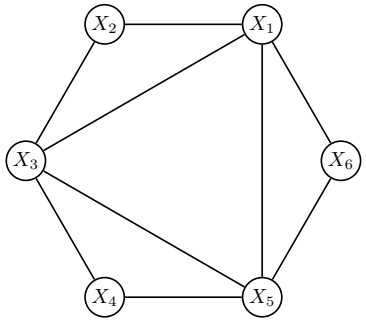
```
def EncoderBlock(X):
  Z = LayerNorm(MultiHeadAttn(Q=[a], K=[b], V=[c]) + X)
  E = LayerNorm(FeedForward(Z) + Z)
  return E

def Encoder(X, L):
  E = POS(Embed(X))
  for n in range(L):
     E = EncoderBlock(E)
  return E

def DecoderBlock(Y, E):
  Z = LayerNorm(MultiHeadAttn(Q=[d], K=[e], V=[f]) + Y)
  Z' = LayerNorm(MultiHeadAttn(Q=[g], K=[h], V=[i]) + Z)
  D = LayerNorm(FeedForward(Z') + Z')
  return D

def Decoder(Y, E, N):
  D = POS(Embed(Y))
  for n in range(N):
     D = DecoderBlock(D,E)
  return D
```

Circle the appropriate answer for each of the possible choices for the query `Q`, keys `K`, and values `V` in the above pseudocode.

[a]:	[X]	Y	Z	E	D
[b]:	[X]	Y	Z	E	D
[c]:	[X]	Y	Z	E	D
[d]:	X	[Y]	Z	E	D
[e]:	X	[Y]	Z	E	D
[f]:	X	[Y]	Z	E	D
[g]:	X	Y	[Z]	E	D
[h]:	X	Y	Z	[E]	D
[i]:	X	Y	Z	[E]	D

2. ***Graphical models*** (10 points)

Consider the following graph on six variables:

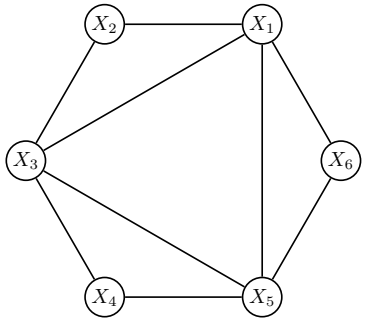


(a) Write down the general form for a strictly positive probability density that factors (is Markov) with respect to this graph.

$$p(x_1, x_2, x_3, x_4, x_5, x_6) = \psi_1(x_1, x_2, x_3)\, \psi_2(x_3, x_4, x_5)\, \psi_3(x_5, x_6, x_1)\, \psi_4(x_1, x_3, x_5)$$

(b) If a multivariate Gaussian has this graph, what is the form of the sparsity pattern of the precision matrix Ω? Enter 0 for a zero entry and $*$ for a non-zero entry.

$$\Omega = \begin{array}{c} \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \end{array} \begin{array}{c} \begin{array}{cccccc} X_1 & X_2 & X_3 & X_4 & X_5 & X_6 \end{array} \\ \left[\begin{array}{cccccc} * & * & * & 0 & * & * \\ * & * & * & 0 & 0 & 0 \\ * & * & * & * & * & 0 \\ 0 & 0 & * & * & * & 0 \\ * & 0 & * & * & * & * \\ * & 0 & 0 & 0 & * & * \end{array}\right] \end{array}$$



(c) Based on this graph (repeated above) which of the following independence relations hold? Circle all that apply.

(1) $X_1 \perp\!\!\!\perp \{X_2, X_5\} \mid \{X_4, X_6\}$

(2) $X_3 \perp\!\!\!\perp X_5 \mid X_4$

(3) $\boxed{X_2 \perp\!\!\!\perp X_4 \mid \{X_3, X_5\}}$

(4) $X_3 \perp\!\!\!\perp X_6 \mid X_2$

(5) $\boxed{X_2 \perp\!\!\!\perp X_5 \mid \{X_1, X_3\}}$

(6) $X_1 \perp\!\!\!\perp \{X_4, X_5\} \mid \{X_2, X_3, X_6\}$

3. ***Coding: Loss functions for language models*** (10 points)

In this problem you are asked to specify and implement the loss function for training language models.

We studied neural network models for sequences, including RNNs, GRUs, and Transformers. This problem asks you to code the loss function used to train these sequence models.

For concreteness, consider sequence models that output characters, with a 27 letter vocabulary for the letters `a-z` and a space `␣` where letters are numbered 0-25 and the space (␣) is numbered 26. So, the string `hello␣world` corresponds to the integer sequence

$$[7, 4, 11, 11, 14, 27, 22, 14, 17, 11, 3].$$

In all of the sequence models we discussed, given a sequence $x_1, x_2, \ldots, x_{t-1}$ of characters so far, the neural network outputs a score

$$s_t \equiv s(x_1, x_2, \ldots, x_{t-1}) \in \mathbb{R}^{27}.$$

used to assign a probability to the next character x_t. The training text is a long character sequence $x_1, x_2, x_3, \ldots, x_T$ where T is the length of the entire training data. Assume that s_1 is the all zeros vector.

(a) Write down a mathematical expression for the loss function used to train the language model, expressed in terms of the score s_t and character sequence x_t defined above.

$$Loss = \frac{1}{T} \sum_{t=1}^{T} \left\{ \log \left(\sum_{j=0}^{27} \exp(s_{tj}) \right) - s_{tx_t} \right\}.$$

(b) Assume that in stochastic gradient descent, a mini-batch of data are processed, leading to a numpy array s of shape $(B, 27)$ where B is the batch size, and an integer array y of length B; the values $s[b]$ are the scores (logits) for item b in the minibatch, and $y[b]$ is the integer value of the next character.

Implement your loss function in part (a) in Python as a function `loss(s, y)` below. (Note: You can use numpy operations rather than tensorflow operations that would allow automatic differentiation to compute the gradients.)

```
def loss(s, y):
  return np.mean(np.log(np.sum(np.exp(s), axis=1)) - s[range(B),y])
```

4. ***Reinforcement learning*** (15 points)

Suppose that an environment has six states, $s = 0, \ldots, 5$, and an agent can take two actions $a = \texttt{left}$ and $a = \texttt{right}$. States 0 and 5 are terminal states, meaning that when the environment transitions to either of these states, the episode ends. Otherwise, when the agent is in state s and takes action `left`, the next state is $s - 1$; when the agent takes action `right`, the next state is $s + 1$: That is, the state transitions and rewards are defined as follows:

$$\begin{aligned} \texttt{next}(s, \texttt{left}) &= s - 1 \quad \text{if } s > 0 \\ \texttt{next}(s, \texttt{right}) &= s + 1 \quad \text{if } s < 5. \end{aligned}$$

If the agent moves to state 0, it receives a reward of 100 (and the episode ends), and if it moves to state 5, it receives a reward of 40 (and the episode ends). Otherwise, it receives a reward of zero. Thus:

$$\texttt{reward}(s, \texttt{left}) = \begin{cases} 100 & \text{if } s = 1 \\ 0 & \text{otherwise;} \end{cases}$$

$$\texttt{reward}(s, \texttt{right}) = \begin{cases} 40 & \text{if } s = 4 \\ 0 & \text{otherwise.} \end{cases}$$

(a) Give an optimal (deterministic) policy $\pi^*(s)$ using discount $\gamma = \frac{1}{2}$. Write your answer by showing which action to take in each non-terminal state below:

$\pi^*(1) = \texttt{left}$

$\pi^*(2) = \texttt{left}$

$\pi^*(3) = \texttt{left}$

$\pi^*(4) = \texttt{right}$

(b) Give the optimal value function v^* corresponding to this policy, again assuming discount factor $\gamma = \frac{1}{2}$. Write your answer by showing the optimal value in each non-terminal state below:

$$v^*(1) = 100$$

$$v^*(2) = 50$$

$$v^*(3) = 25$$

$$v^*(4) = 40$$

(c) Show that the optimal value function satisfies the Bellman equation.

The Bellman equation is

$$v^*(s) = \max_a \{r(s,a) + \gamma \cdot v^*(s')\}$$

where s' is the next state after taking action a in state s. In this case:

$$\begin{aligned}
s = 1: &\quad 100 = \max\{100 + \gamma \cdot 0,\ 0 + \gamma \cdot 50\} \\
s = 2: &\quad 50 = \max\{0 + \gamma \cdot 100,\ 0 + \gamma \cdot 25\} \\
s = 3: &\quad 25 = \max\{0 + \gamma \cdot 50,\ 0 + \gamma \cdot 20\} \\
s = 4: &\quad 40 = \max\{0 + \gamma \cdot 25,\ 40 + \gamma \cdot 0\}
\end{aligned}$$

Boundary cases:

$$\begin{aligned}
s = 0: &\quad 0 = \max\{0 + \gamma \cdot 0,\ 0 + \gamma \cdot 0\} \\
s = 5: &\quad 0 = \max\{0 + \gamma \cdot 0,\ 0 + \gamma \cdot 0\}
\end{aligned}$$

(d) What is the smallest value of the discount factor γ for which the optimal policy π^* in (a) remains the same? Explain your answer.

If $\pi^*(3) = \texttt{left}$ then $v^*(3) = 0 + \gamma \cdot v^*(2) = \gamma^2 \cdot 100$.
If $\pi^*(3) = \texttt{right}$ then $v^*(3) = 0 + \gamma \cdot v^*(4) = \gamma \cdot 40$.
The two values are the same when $100\,\gamma^2 = 40\,\gamma$ or $\gamma = \frac{2}{5}$.

THANK YOU FOR BEING PART OF IML THIS SEMESTER! HAVE A WONDERFUL WINTER BREAK!