CS 189/289A Introduction to Machine Learning
Fall 2023 Jennifer Listgarten, Jitendra Malik Final

- Please do not open the exam before you are instructed to do so.
- **Electronic devices are forbidden on your person**, including cell phones, tablets, headphones, and laptops. Leave your cell phone off and in a bag; it should not be visible during the exam.
- The exam is closed book and closed notes except for your two 8.5×11 inch cheat sheets.
- You have 2 hours and 50 minutes (unless you are in the DSP program and have a larger time allowance).
- Please write your initials at the top right of each page after this one (e.g., write "JD" if you are John Doe). Finish this by the end of your 2 hours and 50 minutes.
- Mark your answers on the exam itself in the space provided. Do **not** attach any extra sheets.
- For multiple choice questions, fill in the bubble for the single best choice.
- For short and long answer questions, write within the boxes provided. If you run out of space, you may use the last four pages to continue showing your work.

Your Name	
Your SID	
Name and SID of student to your left	
Name and SID of student to your right	
Any winter break plans?	
Have a funny ML joke?	

○ CS 189

○ CS 289A

This page intentionally left blank.

initial here

1 Multiple Choice

For the following questions, select the **single best response**. Each question is worth 1.5 points.

1. Which of the following is a benefit of t-SNE over PCA?

   ◯ t-SNE is robust to local optima during optimization.

   ◯ t-SNE creates a highly interpretable mapping of the original points.

   ◯ t-SNE can be computed in a closed-form solution.

   ◯ t-SNE preserves rich local structures.

   Solution: The correct answer is (d). Check the last slide of Lecture 13 for more explanations.

2. Which of the following is **not true** about Transformer models?

   ◯ They add/concatenate a positional encoding to each token in the input sequence before passing them into the first transformer layer.

   ◯ The key, query and value vectors generated from each token in a self-attention layer must have the same dimension.

   ◯ The attention computations for each head in a multi-head attention layer can be parallelized.

   ◯ Transformer models use masked attention for autoregressive tasks during training time to prevent lookup of future tokens within each self-attention layer.

   Solution: The correct answer is (b). We only need the key and query vectors to have the same dimension since we need to take their dot products to compute the attention scores but the value vectors can have a different dimension.

3. In an L2-regularized linear regression model, which of the following is the most likely effect of increasing the value of λ?

   ◯ Decrease the model's bias and increase its variance.

   ◯ Increase the model's bias and decrease its variance.

   ◯ Increase both the model's bias and its variance.

   ◯ Decrease both the model's bias and its variance.

   Solution: The correct answer is (b). In Homework 5, we've derived the expressions for the bias and variance of the ridge estimator. Observing their dependency on λ, we can see that increasing λ increases the bias while decreasing the variance of the estimator.

4. The general form of regularized linear regression is

   $$\arg\min_{w} \|y - Xw\|_2^2 - \lambda \cdot r(w)$$

   where $r : \mathbb{R}^d \to \mathbb{R}$ is the regularization function. Which of these regularization functions would tend to induce a sparse w? In other words, which function would most likely cause some coefficients of w to be set to zero?

initial here

- ◯ 1
- ◯ $\sum_{i=1}^{d} |w_i|$
- ◯ $\sum_{i=1}^{d} w_i^2$
- ◯ $\max(|w_1|, \ldots, |w_d|)$

Solution: Clarification during exam: it should be $+\lambda \cdot r(w)$, not $-\lambda \cdot r(w)$.

The correct answer is (b), which is L-1/Lasso regularization. Choice (a) adds a constant term to the arg max and, thus, does not affect the result. Choice (c) is L-2/Ridge regularization and does not induce sparsity. Choice (d) penalizes by the max coefficient, and, thus, has no effect on non-max coefficients.

5. A classic example of a kernel for edge detection is shown below. Consider $f : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n}$, defined as the convolution of this kernel onto a single-channel image. Assume that there is appropriate padding resulting in an output with the same shape. What type of operation does this function **not** exhibit equivariance with?

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

- ◯ Permutations
- ◯ Translations
- ◯ Rotations in multiples of 90 degrees
- ◯ Horizontal and vertical reflections

Solution: The correct answer is (a). Permuting the pixels of the image before applying f does not have the same effect as applying f and then permuting the output. On the other hand, all convolutional kernels exhibit translational equivariance, and since this kernel is horizontally and vertically symmetric, it also exhibits rotational and reflectional equivariance.

6. In the context of graph neural networks, which of the following is an **invalid** function for updating each nodes' value based on its neighbors? Let h_v be the embedding for neighbor $v \in \mathcal{N}(u)$, where u is the target node.

- ◯ $\sum_v h_v$
- ◯ $\max_v h_v$
- ◯ $\sum_v v \cdot h_v$
- ◯ $\prod_v h_v$

Solution: Clarification during exam: for choice (d), the product is taken to be component-wise over the embeddings.

The correct answer is (c). It does not have permutation invariance, since the output activation map will change if you permute the input. However, it does have permutation equivariance, since a permutation of the input will result in the same permutation of the output.

initial here

7. Consider a binary classification problem in which we use an asymmetrical loss function: $L(1,0) = 5$, $L(0,1) = 1$, and 0 otherwise. Remember that $L(\hat{y}, y)$ denotes the loss for a class prediction of $\hat{y}$ and a ground-truth class y. Which of the following is true of a point lying on the Bayes decision boundary?

   ◯ The posterior probability of class 0 is equal to the posterior probability of class 1.

   ◯ The posterior probability of class 0 is equal to $\frac{1}{5}$ the posterior probability of class 1.

   ◯ The posterior probability of class 1 is equal to $\frac{1}{5}$ the posterior probability of class 0.

   ◯ The class-conditional probability of class 0 is equal to the class-conditional probability of class 1.

   Solution: The correct answer is (b). At the decision boundary, we have $L(1,0)P(Y = 0|X) = L(0,1)P(Y = 1|X)$.

8. Which of the following is **true** about k-Nearest Neighbors?

   ◯ As $k \to \infty$, the model's error approaches the Bayes error.

   ◯ Each data point's Voronoi cell defining the decision boundaries is a convex set for any possible distance metric.

   ◯ The k-Nearest Neighbors algorithm is generally computationally intensive at test time.

   ◯ k-Nearest Neighbors is an unsupervised learning technique.

   Solution: The correct answer is (c). Option (a) is incorrect because the error rate of k-NN approaches the Bayes error in the limit only if k/n approaches 0. Option (b) is incorrect because Voronoi cell convexity depends on the distance metric used. Option (d) is incorrect because k-NN is a supervised learning algorithm.

9. Which of the following is **not true** about the convergence of value iteration?

   ◯ For a finite MDP and discount factor $\gamma \in (0,1)$, value iteration is guaranteed to converge.

   ◯ Policy iteration and value iteration do not converge to the same optimal values.

   ◯ Value iteration can take an infinite amount of iterations to converge to the exact optimal values.

   ◯ Value iteration will always converge to the same values, regardless of how the values are initialized.

   Solution: The correct answer is (b). Policy iteration and value iteration converge to the same optimal values. All the other options are true.

10. When training trees in a random forest, a valid reason for why we restrict ourselves to a random feature subset at each split is:

    ◯ To encourage implicit feature selection, similar to what LASSO does in regression.

    ◯ To decrease single tree variance, which improves random forest performance.

- ◯ To prevent the dominance of some features to cause each tree to become very similar.
- ◯ None of the above.

Solution: The correct answer is (c). We want lower correlation between trees so that we can significantly lower overall random forest variance.

11. Which of the following is **true** about neural networks?
    - ◯ Adding nonlinear activation functions like ReLU or Sigmoid after each layer of a neural network increases model expressiveness, as opposed to linear activation functions.
    - ◯ A neural network's activation functions must be monotonic for it to converge.
    - ◯ Adding layers to a neural network without any nonlinearities will increase performance.
    - ◯ The tanh activation function does not suffer from vanishing gradients.

    Solution: The correct answer is (a). Option (b) is false because activation functions do not necessarily need to be monotonic (one common non-monotonic non-linearity is GELU). Option (c) is false because additional layers without non-linearities often do not increase performance. Option (d) is false because tanh saturates for large positive and negative values, so derivatives in such regions are small in magnitude.

12. Which of the following is NOT a way in which control theory differs from traditional pattern recognition?
    - ◯ The controller must be able to adapt to varying conditions in an environment.
    - ◯ The controller must generalize to sensory inputs under a variety of lighting and occlusion conditions in an environment.
    - ◯ The controller must deal with a dynamic environment that changes over time, as opposed to a static, unchanging environment.
    - ◯ The controller must learn to recover from disturbances in an environment.

    Solution: The correct answer is (b), because generalization is a desirable property in both control theory and pattern recognition. All the other properties are unique to control theory, however.

13. Which of the following best explains why human-centric video models lack deeper understanding and fail to achieve the performance of image models?
    - ◯ There is a lack of human-centric video data online as opposed to human-centric image data.
    - ◯ Human-centric video data is currently too low-quality (in terms of resolution) for deep models to meaningfully discern objects in an image.
    - ◯ Video models have yet to understand the hierarchy of human behavior, from particular actions at a timestep to larger goals and intentions.

◯ Image models are trained on i.i.d. images, while the temporality of frames in a video breaks this i.i.d. assumption.

Solution: The correct answer is (c), as we currently do not have robust methods that operate under multiple spatiotemporal scales. Taking an action not only depends on the current state of the environment, but also conditioning on a human's intent and their goals.

14. Consider a Hidden Markov Model (HMM) with a sequence of observations $O = (o_1, o_2, \ldots, o_T)$ and states $S = (s_1, s_2, \ldots, s_T)$. Which of the following is true about the Viterbi algorithm?

◯ The Viterbi algorithm maximizes the probability at each state and timestep t, i.e. it gets the optimal solution at each step.

◯ The Viterbi algorithm performs Monte Carlo sampling over the state sequence.

◯ The Viterbi probability $v_t(j)$ is computed by taking the sum over all possible paths leading up to state j at timestep t, weighted by the transition probability from the previous state to the current state.

◯ The Viterbi algorithm chooses the state sequence that maximizes the likelihood of the observation sequence.

Solution: The correct answer is (d). Option (a) is false because the Viterbi algorithm does not assume optimality at each timestep. Option (b) is not true. Option (c) is false because it takes the maximum over the previous path probabilities, not the sum; the forward algorithm takes the sum.

initial here

2 Short Answer

1. (4 points) Suppose you create a 2D convolutional layer with n_{in} input channels, n_{out} output channels, kernel size k, stride s and padding p.

   (a) How many learnable parameters does this layer have?

   Solution: Each filter will have $k \times k \times n_{in}$ weights and a single bias term. Since the number of output channels is n_{out}, this layer will have n_{out} learnable filters. Thus, the total number of parameters will be $n_{out}(k^2 \cdot n_{in} + 1) = k^2 n_{in} n_{out} + n_{out}$.

   (b) We pass in a 3D tensor with height H, width W and depth n_{in} as input to this layer. What will be the height, width and depth of the output?

   Solution: The output will have a depth of n_{out}, height H' and width W' where

$$H' = \left\lfloor \frac{H - k + 2p}{s} \right\rfloor + 1$$
$$W' = \left\lfloor \frac{W - k + 2p}{s} \right\rfloor + 1$$

2. (2 points) In discussion, we derived the Bellman expectation equation for state value functions, which expressed $V(s)$ for some state s as a recursive function of $V(s')$ (where s' is the state at the following time step), the reward r, transition probabilities $p(s', r \mid s, a)$, policy $\pi(a \mid s)$ and discount factor $\gamma \in [0, 1]$. Write down a Bellman equation for the state-action value function, i.e., express $Q(s, a)$ as a recursive function in terms of $Q(s', a')$ (where s' is the state at the next time step and a' the action taken from it), the reward r, transition probabilities $p(\cdot, \cdot \mid \cdot, \cdot)$, policy $\pi(\cdot \mid \cdot)$ and discount factor $\gamma \in [0, 1]$.

   Solution: We can see that

$$\begin{aligned} Q(s,a) &= \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{k+t+1} \mid S_t = s, A_t = a\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma \sum_{k=0}^{\infty} R_{k+t+2} \mid S_t = s, A_t = a\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a\right] \end{aligned}$$

   The conditional expectation is taken over all possible next states S_{t+1} and next actions A_{t+1}. The probability of transitioning to some state s' (and collecting reward r) after taking action a in state s is given by $p(s', r \mid s, a)$. Moreover, the probability of taking some action a' from state s' is given by the policy $\pi(a' \mid s')$. This lets us turn the expectation above into the following sum:

$$Q(s,a) = \sum_{r,\, s',\, a'} (r + \gamma Q(s', a')) p(s', r \mid s, a) \pi(a', s')$$

initial here

$$= \sum_{r,\, s'} p(s', r \mid s, a)\left(r + \sum_{a'} \pi(a', s')Q(s', a')\right)$$

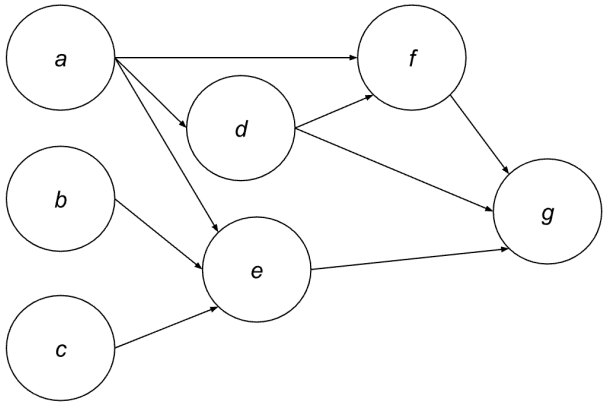
3. (2 points) In discussion we showed why training a random forest of stumps (trees with a single split) is generally a bad idea. However, another form of ensemble learning covered in class, boosting, is often done using stumps. Explain why stumps are good learners for boosting algorithms, in contrast to the deep trees used in random forests.

   Solution: Acceptable answers (need just one):

   (1) Boosting is trained sequentially, in contrast to random forests which are trained in parallel. Short trees significantly speed up the algorithm run-time.

   (2) Boosting reduces bias through weighting misclassified points more during sequential training; the high bias of stumps is not an issue in this case. Meanwhile, deep trees already have low bias, thus boosting with deep trees is more susceptible to overfitting. In contrast, random forests reduce variance across the trees, so deep trees are more appropriate.

4. (3 points) Consider the following computation graph:



   The computations of the intermediary/output nodes follow these equations:

$$d = s(a)$$
$$f = \frac{d}{a}$$
$$g = 3e + 4d + 2f$$
$$e = a^2 * c + b$$

   Here, $s(z) = \frac{1}{1+e^{-z}}$ is the sigmoid function. Find an expression for $\frac{\partial g}{\partial a}$. In this problem, all variables are scalars.

   Solution: By the chain rule,

$$\frac{\partial g}{\partial a} = \frac{\partial g}{\partial d} \cdot \frac{\partial d}{\partial a} + \frac{\partial g}{\partial f} \cdot \frac{\partial f}{\partial a} + \frac{\partial g}{\partial e} \cdot \frac{\partial e}{\partial a}$$

initial here

$$\frac{\partial g}{\partial d} = \frac{\partial g}{\partial d} + \frac{\partial g}{\partial f} \cdot \frac{\partial f}{\partial d} = 4 + 2 \cdot \frac{1}{a} = 4 + \frac{2}{a}$$
$$\frac{\partial g}{\partial e} = 3$$
$$\frac{\partial e}{\partial a} = 2ac$$
$$\frac{\partial d}{\partial a} = s(a)(1 - s(a))$$
$$\frac{\partial g}{\partial f} = 2$$
$$\frac{\partial f}{\partial a} = -\frac{d}{a^2}$$

Therefore,

$$\frac{\partial g}{\partial a} = \left(4 + \frac{2}{a}\right) \cdot s(a)(1 - s(a)) - \frac{2d}{a^2} + 6ac$$

5. (3 points) In lecture, we saw to how derive linear regression using maximum likelihood estimation assuming Gaussian errors. Mathematically, we saw that OLS was the MLE for β under the model

$$y \sim N(X\beta, \sigma^2 I)$$

In this problem, we relax the assumption that each y_i has the same (known) error variance σ^2 and instead has its own (known) variance σ_i^2. Our probabilistic model is therefore:

$$y \sim N(X\beta, \Sigma)$$

where $\Sigma = \text{diag}(\sigma_1^2, \sigma_2^2, ...\sigma_n^2)$.

Derive the maximum likelihood estimate of β under this model. Assume $\beta \in \mathbb{R}^d$, $X \in \mathbb{R}^{n \times d}$, $y \in \mathbb{R}^n$ and that X is full rank.

Solution: Writing out the PDF of our model, with $\Sigma = \text{diag}([\sigma_1^2, \sigma_2^2, ..., \sigma_n^2])$

$$f(y) \propto \exp\left(-\frac{1}{2}(X\beta - y)^T \Sigma^{-1}(X\beta - y)\right)$$
$$\implies l(y) = \left(-\frac{1}{2}(X\beta - y)^T \Sigma^{-1}(X\beta - y)\right)$$
$$= \|\Sigma^{-1/2}(X\beta - y)\|_2^2$$

Here, $\Sigma^{-1/2} = \text{diag}\left(1/\sigma_1, 1/\sigma_2, \cdots, 1/\sigma_n\right)$. We can then proceed to optimize our cost function:

$$\underset{\beta}{\text{argmin}}\ l(y) = \underset{\beta}{\text{argmin}}\ \|\Sigma^{-1/2}(X\beta - y)\|_2^2$$

initial here

Taking the gradient and setting it to 0,

$$\nabla_\beta \|\Sigma^{-1/2}(X\beta - y)\|_2^2 = (\Sigma^{-1/2}X)^T(\Sigma^{-1/2}X\beta - \Sigma^{-1/2}y) = 0$$
$$\implies X^T\Sigma^{-1}X\beta = X^T\Sigma^{-1}y$$
$$\implies \hat{\beta} = (X^T\Sigma^{-1}X)^{-1}X^T\Sigma^{-1}y$$

This is precisely the weighted least squares solution, where $W = \Sigma^{-1}$. This intuitively makes sense as we should trust samples with higher error variance less than those with lower variance.

6. (4 points) Consider the problem of text classification, where you want to learn a probability of whether a token is associated with one of d classes. An example use case of this model would be sentiment analysis, where you would want to classify a word as being happy, sad, or angry.

   More formally, let X be a random variable that captures the distribution of possible labels for some token. The possible labels themselves are one of d classes: $1, 2, \ldots, d$. We collect n samples of X and sum up the counts of each class label, which we will call $C_1, C_2, \ldots, C_d$ (note these are also random variables). Note that $C_1 + C_2 + \ldots + C_d = n$. With this data, we generate estimates of the class probabilities $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_d$. Let the ground truth class probabilities be $p_1, p_2, \ldots, p_d$.

   One way to determine $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_d$ is through **Laplace smoothing**, which pretends that we saw the token α additional times for each class. *Without* Laplace smoothing, the probabilities would be:

$$\hat{p}_i = \frac{C_i}{n}$$

   With Laplace smoothing, the probabilities would be:

$$\hat{p}_i = \frac{C_i + \alpha}{n + d\alpha}$$

   (a) (2 points) Provide a reason for why Laplace-smoothed probabilities may be preferable to basic counting in text classification. 1-2 sentences is sufficient. *Hint*: think about edge cases.

   Solution: Laplace smoothing leads to less erratic probabilities for uncommon tokens. For example, let's say we want to perform 3-class classification and observe the token "zeitgeist" 2 times (i.e. $n = 2$, $d = 3$), with $c_1 = 2, c_2 = 0, c_3 = 0$. Without Laplace smoothing, $\hat{p}_1 = 1, \hat{p}_2 = 0.0, \hat{p}_3 = 0.0$, which is undesirable since we only observed a few samples.

   If we performed Laplace smoothing with $\alpha = 10$, then $\hat{p}_1 = \frac{2+10}{2+30} = 0.375, \hat{p}_2 = \frac{0+10}{2+30} = 0.3125, \hat{p}_3 = \frac{0+10}{2+30} = 0.3125$. These are more desirable probabilities, because our model gives a slight preference for the observed class, but the model is clearly not very sure about its answer, as p_2 and p_3 are close to p_1.

initial here

(b) (2 points) For a specific class i, calculate the bias of the Laplace-smoothed probabilities, as compared to the ground truth probabilities. Simplify as much as possible. *Hint*: What is $\mathbb{E}[C_i]$, i.e. what's our expectation of the number of labels with class i?

Solution: Recall the equation of bias:

$$\begin{aligned}
\mathbb{E}[\hat{p}_i - p_i] &= \mathbb{E}[\frac{C_i + \alpha}{n + d\alpha} - p_i] \\
&= \frac{\mathbb{E}[C_i] + \alpha}{n + d\alpha} - p_i \\
&= \frac{np_i + \alpha}{n + d\alpha} - \frac{(n + d\alpha)p_i}{n + d\alpha} \\
&= \frac{np_i + \alpha - np_i - d\alpha p_i}{n + d\alpha} \\
&= \frac{\alpha - d\alpha p_i}{n + d\alpha} \\
&= \frac{\alpha(1 - dp_i)}{n + d\alpha}
\end{aligned}$$

We see that $\mathbb{E}[C_i] = np_i$, since our expected count should scale with the number of samples we have.

initial here

3 Everything is Gaussian

One activation function that has become popular in language models and modern deep learning models is GELU, or Gaussian Error Linear Units. GELU is defined as follows:

$$\text{GELU}(x) = xP(X \leq x) = x\Phi(x)$$

where $X \sim \mathcal{N}(0, 1)$. In other words, it is the product of x and the CDF $\Phi(x)$ of a standard normal distribution. In this problem, we will compare it to the ReLU activation you learned about in class.

(a) (3 points) Prove that $\text{GELU}(x) \leq \text{ReLU}(x)$ for all $x \in \mathbb{R}$.

Solution: We can break this down to two cases that cover $\mathbb{R}$:

- When $x > 0$, we see that $\text{ReLU}(x) = x$. Since the CDF of a distribution is always less than or equal to 1, $x\Phi(x) \leq x \implies \text{GELU} = (x) \leq \text{ReLU}(x)$.
- When $x \leq 0$, we see that $\text{ReLU}(x) = 0$. Since $\Phi(x)$ is always non-negative, $x\Phi(x) \leq 0 \implies \text{GELU}(x) \leq \text{ReLU}(x)$.

(b) (2 points) Consider the value of $\text{GELU}(x)$ at $x = 0$ and at the limits $x \to \pm\infty$. Explain why GELU can be thought of as a smoothed approximation of ReLU.

Solution: We see that

- $x \to 0 \implies \text{GELU}(x) \to \text{ReLU}(x) = 0$.
- $x \to \infty \implies \text{GELU}(x) \to \text{ReLU}(x) = x$.
- $x \to -\infty \implies \text{GELU}(x) \to \text{ReLU}(x) = 0$.

In addition, while ReLU is a piecewise function, GELU is smooth and differentiable across all real values. Thus, it approximates ReLU smoothly.

(c) (3 points) Prove or disprove that GELU is a convex function. What about ReLU?
Hint: the derivative of a distribution's CDF is its PDF.

Solution: GELU is non-convex. To prove convexity, we need to show that the second derivative is always nonnegative. We observe that

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}x}\text{GELU}(x) &= \frac{\mathrm{d}}{\mathrm{d}x}x\Phi(x) \\
&= \Phi(x) + xf_X(x) \\
\frac{\mathrm{d}^2}{\mathrm{d}x^2}\text{GELU}(x) &= \frac{\mathrm{d}}{\mathrm{d}x}(\Phi(x) + xf_X(x)) \\
&= f_X(x) + f_X(x) + x(-xf(x)) \\
&= f_X(x)(2 - x^2)
\end{aligned}$$

We note that $f_X(x)$ is always non-negative, so we only have to worry about $2 - x^2$, which will become negative when $x > \sqrt{2}$ or $x < -\sqrt{2}$. Thus, GELU is not convex.
On the other hand, ReLU is a convex function.

initial here

4 Books Are All You Need

Alice works at a local bookstore and is responsible for their weekly promotion strategy. Having learned about Markov Decision Processes (MDPs), she plans to apply this method to optimize book sales. The bookstore focuses mainly on two genres: fiction and non-fiction. Each week, Alice must decide which genre to promote based on the store's current main genre, aiming to maximize sales over the semester.

Alice formulates the problem as an MDP. The state space $\mathcal{S} = \{F, N\}$ represents the main genre at the beginning of week t (F for fiction, N for non-fiction). The action space $\mathcal{A} = \{A, B\}$ represents the promotion plan for week t, with plan A promoting fiction and plan B promoting non-fiction.

The transition dynamics $P(s'|s, a)$ describe the likelihood of the promotion strategy changing the main genre. For example, if the current main genre is fiction (F), promoting fiction (action A) has an 80% chance of keeping the genre as fiction and a 20% chance of switching to non-fiction (N), whereas promoting non-fiction has an equal chance of keeping the genre as fiction and switching to non-fiction. The same dynamics apply if the current main genre is non-fiction. The transition dynamics are:

$$P(s' = F|s = F, a = A) = 0.8;\ P(s' = N|s = F, a = A) = 0.2$$
$$P(s' = F|s = F, a = B) = 0.5;\ P(s' = N|s = F, a = B) = 0.5$$
$$P(s' = F|s = N, a = A) = 0.5;\ P(s' = N|s = N, a = A) = 0.5$$
$$P(s' = F|s = N, a = B) = 0.2;\ P(s' = N|s = N, a = B) = 0.8$$

The reward $R(s, a)$ represents the sales for week t. Sales are typically higher when the promotion strategy aligns with the current main genre of the store. The rewards are defined as:

$$R(s = F, a = A) = 10;\ R(s = F, a = B) = 5$$
$$R(s = N, a = A) = 5;\ R(s = N, a = B) = 10$$

Alice wishes to determine the optimal promotion strategy using the MDP model. Observing that non-fiction has been selling better, she initializes the value functions as $V_0(F) = 5, V_0(N) = 10$. However, she prefers promoting fiction, so she sets the initial policy as $\pi_0(A|s) = 0.6, \pi_0(B|s) = 0.4, \forall s \in \mathcal{S}$. She also assumes a discount factor $\gamma = 0.5$.

initial here

(a) (1 point) Both the policy iteration algorithm and the value iteration algorithm can be used to find the optimal policies given an MDP. In both algorithms, we need to compute the following expectation:

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a)V(s')$$

Compute $Q(s = F, a = A)$ using $V_0(s)$.

Solution: We substitute in the appropriate reward function $R(\cdot,\cdot)$, transition probabilities $P(\cdot|\cdot,\cdot)$ and value function $V_0(\cdot)$:

$$\begin{aligned} Q(s = F, a = A) &= R(F,A) + \gamma \left[P(F|F,A)V_0(F) + P(N|F,A)V_0(N)\right] \\ &= 10 + 0.5[0.8 * 5 + 0.2 * 10] \\ &= 13 \end{aligned}$$

(b) (2 points) In part (b)-(d), assume $Q(\cdot,\cdot)$ are the following:

$$\begin{aligned} Q(s = F, a = A) &= 10 \\ Q(s = F, a = B) &= 8 \\ Q(s = N, a = A) &= 6 \\ Q(s = N, a = B) &= 15 \end{aligned}$$

Perform one step of policy evaluation by computing the updated value functions $V(F)$ and $V(N)$ using the initial policy $\pi_0(\cdot|\cdot)$ and $Q(\cdot,\cdot)$.

Solution: The policy evaluation update for each state s is:

$$V(s) = \sum_a \pi(a|s)Q(s,a)$$

Hence,

$$\begin{aligned} V(F) &= \pi_0(A|F)Q(F,A) + \pi_0(B|F)Q(F,B) \\ &= 0.6 * 10 + 0.4 * 8 \\ &= 9.2 \\ V(N) &= \pi_0(A|N)\left(Q(N,A)\right) + \pi_0(B|N)Q(N,B) \\ &= 0.6 * 6 + 0.4 * 15 \\ &= 9.6 \end{aligned}$$

(c) (2 points) Perform one step of policy improvement by computing the best action $\pi(s)$ for states F and N using $Q(\cdot,\cdot)$.

Solution: The policy improvement update for each state s is:

$$\pi(s) = \operatorname*{argmax}_a Q(s,a)$$

Hence,

$$\pi(F) = \underset{A,B}{\text{argmax}} \ \{Q(F,A), Q(F,B)\} = A$$
$$\pi(N) = \underset{A,B}{\text{argmax}} \ \{Q(N,A), Q(N,B)\} = B$$

(d) (2 points) Perform one step of value iteration by computing the updated value functions $V(\cdot)$ for states F and N using $Q(\cdot,\cdot)$.

Solution: The value iteration update for each state s is:

$$V(s) = \max_a Q(s,a)$$

Hence,

$$V(F) = \max_{A,B} \{Q(F,A), Q(F,B)\} = 10$$
$$V(N) = \max_{A,B} \{Q(F,A), Q(F,B)\} = 15$$

(e) (2 points) Alice notices that the success of the current week's promotion depends on the sequence of promoted genres over the past two weeks. How might this new scenario be problematic in the original MDP model Alice formulated? How should the state space $\mathcal{S}$ be redefined to account for this new insight?

Solution: The main genre of week t, S_t, now depends on both S_{t-1} and S_{t-2}, as opposed to just S_{t-1}. Therefore, the states no longer have the Markov property.

For the states to still have the Markov property, we can change the state spaces to be a tuple representing the main genre of the store in the current and previous week:

$$\mathcal{S} = \{(F,F), (F,N), (N,F), (N,N)\}$$

The new states have the Markov Property, and we can define the transition probabilities $P(s'|s,a)$ accordingly. Note that the transition probabilities are zero between some states, e.g. from (F,F) to (N,N).

initial here

5 Mixture of Linear Regressors

In homework 7, we employed the Expectation-Maximization algorithm to learn the parameters of a Gaussian mixture model. In this problem, we will apply the EM algorithm to learn the parameters for a mixture of linear regression models (also called a "Mixture of Experts" model in ML literature).

Suppose we have a dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ of IID samples, where data point $\mathbf{x}_i$ belongs to one of two hidden classes $z_i = 0$ or $z_i = 1$. If $\mathbf{x}_i$ belongs to hidden class $z_i = 0$, then its corresponding label was generated according to $y_i = \boldsymbol{\theta}_0^T \mathbf{x}_i + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Similarly, if $\mathbf{x}_i$ belongs to hidden class $z_i = 1$, then its corresponding label was generated according to $y_i = \boldsymbol{\theta}_1^T \mathbf{x}_i + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. We will assume that σ^2 is known. On the other hand, the true parameters $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ as well as the hidden labels z_i (the hidden label for $\mathbf{x}_i$) are unknown.

Since it is a reasonable to assume that the binary-valued hidden variable z_i depends on the data point $\mathbf{x}_i$, we will model it using a logistic function with parameter $\boldsymbol{\phi}$, i.e.,

$$P(z_i = 1 \mid \mathbf{x}_i; \boldsymbol{\phi}) = s(\boldsymbol{\phi}^T \mathbf{x}_i)$$
$$P(z_i = 0 \mid \mathbf{x}_i; \boldsymbol{\phi}) = 1 - s(\boldsymbol{\phi}^T \mathbf{x}_i)$$

where $s(z) = \frac{1}{1+\exp(-z)}$ is the sigmoid function.

In this problem, we will walk through how the expectation-maximization algorithm can be applied to learn the parameters $\boldsymbol{\theta}_0$, $\boldsymbol{\theta}_1$ and $\boldsymbol{\phi}$. Why might we care about such a model? Intuitively, standard OLS linear regression might not model the following dataset as well as a combination of two linear regression models over different regions of the data domain:

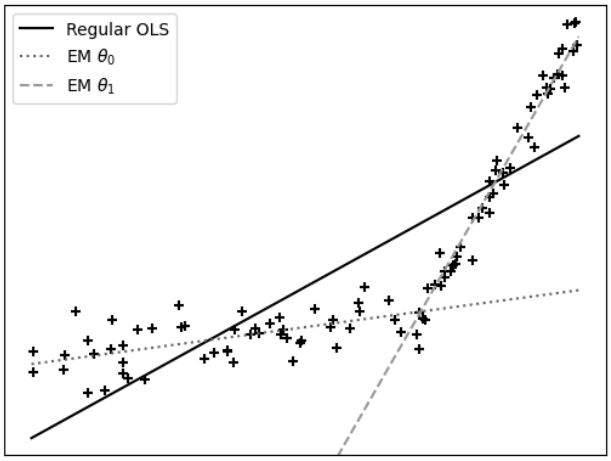


Figure 1: We can see that the mixture model parameters $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ derived using the EM algorithm fit the dataset much better than the regular OLS solution!

A note on notation: the expression $P(a_1, \ldots, a_i | b_1, \ldots, b_j; \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k)$ indicates that the distribution $P(a_1, \ldots, a_i | b_1, \ldots, b_j)$ is parameterized by $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$.

initial here

(a) (3 points) Find the log-likelihood of the parameters given the dataset D, i.e., write down an expression for $l(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\phi}) = \log P(y_1, \ldots, y_n \mid \mathbf{x}_1, \ldots, \mathbf{x}_n; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\phi})$. You may leave your expression as a sum of logarithms.

Hint: note that $P(y \mid x) = P(y, z = 0 \mid x) + P(y, z = 1 \mid x)$ by the law of total probability.

Solution: Since the data points are IID,

$$\log P(y_1, \ldots, y_n \mid \mathbf{x}_1, \ldots, \mathbf{x}_n; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\phi}) = \sum_{i=1}^{n} \log P(y_i \mid x_i; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\phi})$$

Following the law of total probability

$$\begin{aligned}
&P(y_i \mid \mathbf{x}_i; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\phi}) \\
&= P(y_i, z_i = 0 \mid \mathbf{x}_i; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\phi}) + P(y_i, z_i = 1 \mid \mathbf{x}_i; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\phi}) \\
&= P(y_i \mid \mathbf{x}_i, z_i = 0; \boldsymbol{\theta}_0) P(z_i = 0 \mid \mathbf{x}_i; \boldsymbol{\phi}) + P(y_i \mid \mathbf{x}_i, z_i = 1; \boldsymbol{\theta}_1) P(z_i = 1 \mid \mathbf{x}_i; \boldsymbol{\phi}) \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \boldsymbol{\theta}_0^T \mathbf{x}_i)^2}{2\sigma^2} \right) (1 - s(\boldsymbol{\phi}^T \mathbf{x}_i)) + \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \boldsymbol{\theta}_1^T \mathbf{x}_i)^2}{2\sigma^2} \right) s(\boldsymbol{\phi}^T \mathbf{x}_i)
\end{aligned}$$

Thus,

$$\begin{aligned}
l(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\phi}) = \sum_{i=1}^{n} \log\Bigg(&\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \boldsymbol{\theta}_0^T \mathbf{x}_i)^2}{2\sigma^2} \right) (1 - s(\boldsymbol{\phi}^T \mathbf{x}_i)) \\
&+ \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \boldsymbol{\theta}_1^T \mathbf{x}_i)^2}{2\sigma^2} \right) s(\boldsymbol{\phi}^T \mathbf{x}_i) \Bigg)
\end{aligned}$$

Our goal is to find the parameters that maximize the log-likelihood above. Unfortunately, there is no closed-form expression for the log-likelihood above so we cannot use our standard technique of setting its gradient equal to 0. Therefore, we will proceed with the iterative Expectation-Maximization algorithm.

Letting $q_i(k)$ (a shorthand for $q_i(z_i = k)$) be a possible distribution over z_i, we can use Jensen's inequality to lower bound our log-likelihood by

$$\begin{aligned}
l(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\phi}) \geq \mathcal{F}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\phi}, q) &= \sum_{i=1}^{n} \sum_{k \in \{0,1\}} q_i(k) \log \left(\frac{P(y_i, z_i = k \mid \mathbf{x}_i; \boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\phi})}{q_i(k)} \right) \\
&= \sum_{i=1}^{n} \sum_{k \in \{0,1\}} q_i(k) \log \left(\frac{P(y_i \mid \mathbf{x}_i, z_i = k; \boldsymbol{\theta}_k) P(z_i = k \mid \mathbf{x}_i; \boldsymbol{\phi})}{q_i(k)} \right)
\end{aligned}$$

We note that $q_i(0) + q_i(1) = 1$.

Now, as seen in the homework, we can view the EM algorithm as a coordinate ascent algorithm where we perform the following steps in an alternating order:

$$q^{t+1} = \operatorname*{argmax}_{q} \mathcal{F}(\boldsymbol{\theta}_0^t, \boldsymbol{\theta}_1^t, \boldsymbol{\phi}^t, q) \qquad \text{E-step}$$

initial here

$$\boldsymbol{\theta}_0^{t+1}, \boldsymbol{\theta}_1^{t+1}, \boldsymbol{\phi}^{t+1} = \underset{\boldsymbol{\theta}_0,\, \boldsymbol{\theta}_1,\, \boldsymbol{\phi}}{\operatorname{argmax}}\ \mathcal{F}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\phi}, q^{t+1}) \qquad \text{M-step}$$

where $t = 0$ represents the initial guesses for the variables above. Note that the $(\cdot)^t$ here does not represent exponentiation but it denotes the value of a variable after the tth iteration of the EM algorithm.

(b) (3 points) We claim that $q_i^{t+1}(k) = P(z_i = k \mid \mathbf{x}_i, y_i; \boldsymbol{\theta}_0^t, \boldsymbol{\theta}_1^t, \boldsymbol{\phi}^t)$ is a valid update for the E-step (note that this is just a generalization of the expectation step from the homework). Find an expression for $q_i^{t+1}(1)$.

Hint: you may find the Bayes rule helpful: $P(z \mid x, y) = P(y \mid x, z)P(z \mid x)/P(y \mid x)$.

Hint: you may the hint for part (a) helpful as well.

Solution: We appeal to Bayes rule and the law of total probability to write

$$\begin{aligned}
q_i^{t+1}(1) &= P(z_i = 1 \mid \mathbf{x}_i, y_i; \boldsymbol{\theta}_0^t, \boldsymbol{\theta}_1^t, \boldsymbol{\phi}^t) \\
&= \frac{P(y_i \mid \mathbf{x}_i, z_i = 1; \boldsymbol{\theta}_0^t, \boldsymbol{\theta}_1^t) P(z_i = 1 \mid \mathbf{x}_i; \boldsymbol{\phi}^t)}{P(y_i \mid x_i; \boldsymbol{\theta}_0^t, \boldsymbol{\theta}_1^t, \boldsymbol{\phi}^t)} \\
&= \frac{P(y_i \mid \mathbf{x}_i, z_i = 1; \boldsymbol{\theta}_0^t, \boldsymbol{\theta}_1^t) P(z_i = 1 \mid \mathbf{x}_i; \boldsymbol{\phi}^t)}{P(y_i \mid \mathbf{x}_i, z_i = 0; \boldsymbol{\theta}_0^t, \boldsymbol{\theta}_1^t) P(z_i = 0 \mid \mathbf{x}_i; \boldsymbol{\phi}^t) + P(y_i \mid \mathbf{x}_i, z_i = 1; \boldsymbol{\theta}_0^t, \boldsymbol{\theta}_1^t) P(z_i = 1 \mid \mathbf{x}_i; \boldsymbol{\phi}^t)} \\
&= \frac{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \mathbf{x}_i^T \boldsymbol{\theta}_1^t)^2}{2\sigma^2}\right) s(\mathbf{x}_i^T \boldsymbol{\phi}^t)}{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \mathbf{x}_i^T \boldsymbol{\theta}_0^t)^2}{2\sigma^2}\right)(1 - s(\mathbf{x}_i^T \boldsymbol{\phi}^t)) + \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \mathbf{x}_i^T \boldsymbol{\theta}_1^t)^2}{2\sigma^2}\right) s(\mathbf{x}_i^T \boldsymbol{\phi}^t)}
\end{aligned}$$

From here, you can also drop the common $\frac{1}{\sqrt{2\pi\sigma^2}}$ factor if you wish.

As a side note, just finding $q_i^{t+1}(1)$ suffices since $q_i^{t+1}(0) = 1 - q_i^{t+1}(1)$ can be also be computed from it.

(c) We will now derive our M-step updates:

$$\boldsymbol{\theta}_0^{t+1}, \boldsymbol{\theta}_1^{t+1}, \boldsymbol{\phi}^{t+1} = \underset{\boldsymbol{\theta}_0,\, \boldsymbol{\theta}_1,\, \boldsymbol{\phi}}{\operatorname{argmax}}\ \mathcal{F}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\phi}, q^{t+1})$$

Note that q^{t+1} is the output of the E-step above, and is fixed (in other words, the probabilities $q_i^{t+1}(k)$ for $i = 1, \ldots, n$ and $k = 0, 1$ are to be treated as fixed constants) throughout this part. Suppose the data points $\mathbf{x}_i \in \mathbb{R}^d$ are stacked row-wise into the $n \times d$ data matrix X and the labels are stacked into a column vector $\mathbf{y} \in \mathbb{R}^n$. For the sake of convenience, we will assume that X is full rank.

(i) (4 points) Find the closed-form solution for $\boldsymbol{\theta}_0^{t+1}$ by setting the gradient of $\mathcal{F}$ to $\mathbf{0}$.
Hint: it may be helpful to define the matrix $W_0 = \operatorname{diag}(q_1^{t+1}(0), \ldots, q_n^{t+1}(0))$. Note that $q_i^{t+1}(0) \geq 0$ since it is a probability.

Solution: We set the gradient of $\mathcal{F}$ with respect to $\boldsymbol{\theta}_0$ to zero:

$$\begin{aligned}
0 &= \nabla_{\boldsymbol{\theta}_0} \mathcal{F}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\phi}, q^{t+1}) \\
&= \nabla_{\boldsymbol{\theta}_0} \sum_{i=1}^{n} \sum_{k \in \{0,1\}} q_i^{t+1}(k) \log\left(\frac{P(y_i, \mid \mathbf{x}_i, z_i = k; \boldsymbol{\theta}_k) P(z_i = k \mid \mathbf{x}_i; \boldsymbol{\phi})}{q_i^{t+1}(k)}\right)
\end{aligned}$$

initial here

$$= \nabla_{\theta_0} \sum_{i=1}^{n} q_i^{t+1}(0) \log(P(y_i \mid \mathbf{x}_i, z_i = 0; \boldsymbol{\theta}_0))$$

$$= \nabla_{\theta_0} \sum_{i=1}^{n} q_i^{t+1}(0) \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \mathbf{x}_i^T \boldsymbol{\theta}_0)^2}{2\sigma^2} \right) \right)$$

$$= \nabla_{\theta_0} - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (q_i^{t+1}(0)^{1/2})^2 (y_i - \mathbf{x}_i^T \boldsymbol{\theta}_0)^2$$

$$= \nabla_{\theta_0} - \frac{1}{2\sigma^2} \left\| (W_0)^{1/2} (\mathbf{y} - X\boldsymbol{\theta}_0) \right\|_2^2$$

Moving from line 2 to line 3, we note that any terms **not** dependent on $\boldsymbol{\theta}_0$ get zeroed out by the gradient. Let $\tilde{\mathbf{y}} = W_0^{1/2}\mathbf{y}$ and $\tilde{X} = W_0^{1/2}X$. Then, we are solving

$$\nabla_{\theta_0} \left\| \tilde{\mathbf{y}} - \tilde{X}\boldsymbol{\theta}_0 \right\|_2^2 = \mathbf{0}$$

Solving this yields $\boldsymbol{\theta}_0^{t+1} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T\tilde{\mathbf{y}} = (X^T W_0 X)^{-1} X^T W_0 \mathbf{y}$ — this is just the solution to weighted least squares with weight matrix W_0!

This intuitively makes sense since any $\mathbf{x}_i$ belonging to hidden class 0 will have a high $q_i^{t+1}(0)$ and any belonging to hidden class 1 will have a low $q_i^{t+1}(0)$. The weighted least squares solution then appropriately re-weights the contribution of each data point to the update for $\boldsymbol{\theta}_0^{t+1}$ according to their posterior probabilities.

(ii) (4 points) Find the closed-form solution for $\boldsymbol{\theta}_1^{t+1}$ by setting the gradient of $\mathcal{F}$ to $\mathbf{0}$. *Hint:* it may be helpful to define the matrix $W_1 = \text{diag}(q_1^{t+1}(1), \ldots, q_n^{t+1}(1))$. Note that $q_i^{t+1}(1) \geq 0$ since it is a probability.

Solution: We just repeat the same solution above to get $\boldsymbol{\theta}_1^{t+1} = (X^T W_1 X)^{-1} X^T W_1 \mathbf{y}$.

(iii) (4 points) Unlike the updates $\boldsymbol{\theta}_0^{t+1}$ and $\boldsymbol{\theta}_1^{t+1}$ above, the update for $\boldsymbol{\phi}^{t+1}$ does not have a closed form solution. So, we can choose to approximate it using some iterations of gradient ascent $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \nabla_{\boldsymbol{\phi}} \mathcal{F}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\phi}, q^{t+1})$ instead. Write out the GA update. *Hint:* you may use without proof that $s'(z) = s(z)(1 - s(z))$.

Solution: Note that

$$\nabla_{\boldsymbol{\phi}} \mathcal{F}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\phi}, q^{t+1}) = \nabla_{\boldsymbol{\phi}} \sum_{i=1}^{n} \sum_{k \in \{0,1\}} q_i^{t+1}(k) \log\left(\frac{P(y_i, \mid \mathbf{x}_i, z_i = k; \boldsymbol{\theta}_k) P(z_i = k \mid \mathbf{x}_i; \boldsymbol{\phi})}{q_i(k)} \right)$$

$$= \nabla_{\boldsymbol{\phi}} \sum_{i=1}^{n} \sum_{k \in \{0,1\}} q_i^{t+1}(k) \log(P(z_i = k \mid \mathbf{x}_i; \boldsymbol{\phi}))$$

Moving from line 1 to line 2, the gradient zeros out any terms that **do not** depend on $\boldsymbol{\phi}$. We now examine the inner summation:

$$\sum_{k \in \{0,1\}} q_i^{t+1}(k) \log(P(z_i = k \mid \mathbf{x}_i; \boldsymbol{\phi})) = q_i^{t+1}(1) \log\Big(s(\boldsymbol{\phi}^T \mathbf{x}_i)\Big) + q_i^{t+1}(0) \log\Big(1 - s(\boldsymbol{\phi}^T \mathbf{x}_i)\Big)$$

Then,

$$\nabla_{\phi} \Big(q_i^{t+1}(1) \log\Big(s(\boldsymbol{\phi}^T \mathbf{x}_i)\Big) + q_i^{t+1}(0) \log\Big(1 - s(\boldsymbol{\phi}^T \mathbf{x}_i)\Big) \Big)$$

initial here

$$= q_i^{t+1}(1)\frac{1}{s(\boldsymbol{\phi}^T\mathbf{x}_i)}s(\boldsymbol{\phi}^T\mathbf{x}_i)(1-s(\boldsymbol{\phi}^T\mathbf{x}_i))\mathbf{x}_i - q_i^{t+1}(0)\frac{1}{1-s(\boldsymbol{\phi}^T\mathbf{x}_i)}s(\boldsymbol{\phi}^T\mathbf{x}_i)(1-s(\boldsymbol{\phi}^T\mathbf{x}_i))\mathbf{x}_i$$
$$= (q_i^{t+1}(1)(1-s(\boldsymbol{\phi}^T\mathbf{x}_i)) - (1-q_i^{t+1}(1))s(\boldsymbol{\phi}^T\mathbf{x}_i))\mathbf{x}_i$$
$$= (q_i^{t+1}(1) - s(\boldsymbol{\phi}^T\mathbf{x}_i))\mathbf{x}_i$$

Thus,

$$\nabla_\phi \mathcal{F}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\phi}, q^{t+1}) = \sum_{i=1}^{n}(q_i^{t+1}(1) - s(\boldsymbol{\phi}^T\mathbf{x}_i))\mathbf{x}_i$$

and the GA update is given by

$$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \sum_{i=1}^{n}(q_i^{t+1}(1) - s(\boldsymbol{\phi}^T\mathbf{x}_i))\mathbf{x}_i$$

This derivation follows the exact same steps as the logistic regression gradient ascent update derivation from class!

initial here

6 Machine Unlearning: Decision Trees

On the midterm, we explored how to remove data from OLS linear regression. Now, we investigate the same problem for decision trees. This problem is largely inspired by Brophy and Lowd's 2021 ICML paper "Machine Unlearning for Random Forests."

We have a model that has been trained on features $X \in \mathbb{R}^{n \times d}$ and their targets $y \in \mathbb{R}$. The problem machine unlearning tackles is "removing" the ith datapoint, (X_i, y_i); that is, modifying the model such that it was as if the datapoint was not included during training.

For this problem, we assume that we are fitting a decision tree on a binary classification problem with a single continuous feature.

(a) (3 points) To start, let's consider a naive strategy for unlearning that is computationally expensive: just retrain the model without the datapoint.

Consider a dataset, where $X = [1, 2, 3, 4, 5, 6, 7]$ and $y = [0, 0, 0, 1, 0, 1, 1]$. First fit a depth-1 decision tree (just one split) to the entire dataset. Then remove the fourth datapoint (4, 1) and refit the tree. **Draw both trees along with their splits including the leaf datapoints.**

Solution: Split between 3 and 4 for the original dataset, and between 5 and 6 for the second. The first dataset only has three reasonable splits, 3-4, 4-5, 5-6, of which 3-4 minimizes average entropy.

(b) (4 points) Typically, when removing a point from a decision tree, we will need to recompute many of the splits above that point as that point influenced those splits, which can introduce a lot of extra overhead. If we use a less naive unlearning approach than above, we will want to efficiently edit splits. Randomness can help us achieve this goal.

We consider two types of random splits:

(1) A split where we select the split uniformly at random on the range $[v_{\text{min}}, v_{\text{max}}]$, where v_{min} and v_{max} are the minimum and maximum feature values at a given split.

(2) Let's call a valid split the mean of two adjacent (if we sorted the data by the feature) datapoints. We then randomly sample (without replacement), k of these split and select the best one.

If we randomly sample a point to remove, what is the probability that we have to refit the first kind of random split? What about the second? Your answer should be in terms of n' and k.

Assume we have n' points at the split (therefore $n' - 1$ valid splits) and that all splits are equally likely to be the best split of a given subset. For this part only, assume that we don't care if we remove a point that affects a split which was not the best of the k-sample.

Solution: For the first kind of split, we'd only have to retrain if we selected either the min or max valued sampled. There are only two cases of this happening out of n' total cases, so we get $2/n'$ as our probability.

The second kind of split is a bit more involved. There are two cases: the first case is that we select a min or max valued sample (extreme sample) or we select an interior sample.

Extreme samples only affect a single valid split, but interior samples will each affect two valid splits. The probability of selecting an extreme sample is $2/n'$. We then compute the probability of retraining a split (which we will call event R) given we have selected an extreme sample (event E):

$$P(R|E) = \frac{\binom{n'-2}{k-1}}{\binom{n'-1}{k}} * \frac{1}{k} = \frac{1}{n'-1}$$

The first term is the probability of the extreme split being amongst the k and the second is the probability that given our split is sampled, that it is the best.

$$P(R|\neg E) = 1/k * \left(2 * \frac{\binom{n'-3}{k-1}}{\binom{n'-1}{k}}\right) + 2/k * \left(\frac{\binom{n'-3}{k-2}}{\binom{n'-1}{k}}\right) = \frac{2}{n'-1}$$

Putting both of these together, we get that our total probability of a retrain is:

$$P(R) = \frac{2}{n'}P(R|E) + \frac{n'-2}{n'}P(R|\neg E) = \frac{2}{n'}$$

Interestingly enough, this is the same probability.

(c) (2 points) Let's now consider how we could check if a split needed to be refit after removing a data point in the second kind of split. To do this, we would remove any splits affected by the removal (ie we remove one of the points used to compute the split boundary) and then resample new splits.

What statistics could we store on each of the type 2 splits that would allow us to determine if our current split became suboptimal after removing a datapoint? You should only use $O(1)$ space per split.

Hint: what do we need to compute the information gain of a split in binary classification?

Solution: Information gain is defined as

$$H(L+R) - \left(\frac{|L|}{|L|+|R|}H(L) + \frac{|R|}{|L|+|R|}H(R)\right)$$

With entropy, defined as

$$H(S) = -p_0 \log p_0 - (1-p_0)\log(1-p_0)$$
$$p_0 = \frac{\sum_{i\ in S} \mathbf{1}(i=0)}{|S|}$$

Therefore, all we need to compute information gain of a split is the number of samples at the split and the number of samples in positive class to the left and right of the split.

(d) (2 points) Which split would you prefer at shallower depths in the tree? Be sure to explain your reasoning.

Hint: see the 'for only this part' note in (b).

Solution: We prefer split (1), because we only have to do work if we select an extreme sample. However, in split (2), we'd have to do work whenever a removed datapoint's split was sampled as part of the k. This is because even if we didn't have to retrain a split, we'd still need to resample a split that was affected by our removed node. This increased our probability of having to do non-trivial work (and made the retrain check harder).

(e) (3 points) Devise an algorithm (in words) to remove a datapoint from a tree of max depth 1 where the split is of type (2). Assume each split contains a statistic that fulfills the criterion of part (c).

Pretend that this is the function stub you are writing the logic for:

```
def remove_point(root_node: TreeNode, split_type: Union[1, 2], X, y)
```

Hint: don't forget about leaf nodes.

Solution:

If the point removed affected any point in sample of k, then resample until we have k sampled splits. This is valid because removing splits can be thought of as rejection sampling. If the case where we removed the best split, we now need to re-iterate over all k splits to find the best. If we don't we only need to compare the new splits with the old best split (it's okay if students omitted this minor optimization). If the split changes, refit the leaf nodes. (2 points)

Finally, if the split hasn't changed, we simply need to update the leaf node that the datapoint was in. (1 point)

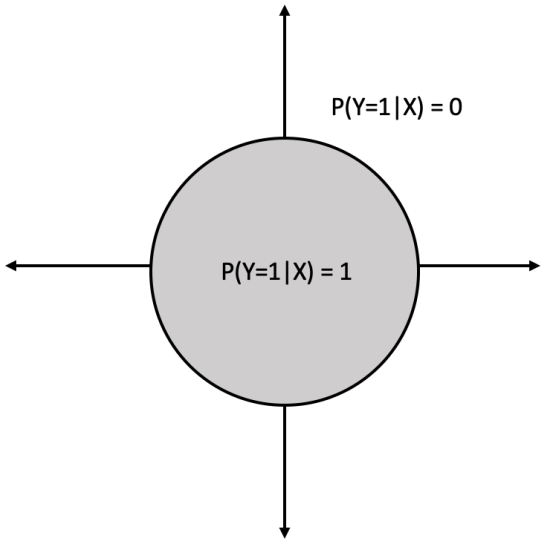
(f) (2 points) Explain (in words), how you might extend this algorithm to decision trees of arbitrary depth.

Solution: We could make the algorithm recursive, by at each split, if we don't change the split call the function of each descendant node.

initial here

7 Nearest Neighbors (CS 289A Only)

Consider performing a case of binary classification with feature vectors $\mathbf{x} = [x_1, x_2]^T$, where $x_1, x_2 \in [-1, 1]$. Suppose that we have an estimate for the conditional probability, shown in the figure below:



where the circular region is centered at (0,0) and has a radius of $r < 1$. We will investigate the performance of a nearest neighbor classifier in this problem.

(a) (2 points) To start, we are interested in training a 1NN classifier. We first uniformly sample n points from the region $[-1, 1] \times [-1, 1]$. Assume that the sampled points' labels exactly follow the estimated conditional distribution in the figure above. Now suppose we use our 1NN classifier to classify a point at (0, 0). Assuming this point comes from the same conditional distribution, what is the probability of misclassification?

Solution: The probability of misclassification of this point is simply the probability that none of the n sampled points falls in the circular region (as (0, 0) will always be closer to a point in the circle than a point outside of it). The probability of one sampled point falling in this region is

$$P(\mathbf{x}_i \in \text{circle}) = \frac{\pi r^2}{4}$$

Thus, the probability that none of the points fall in this region is given by

$$P(\mathbf{x}_1, \ldots, \mathbf{x}_n \notin \text{circle}) = \left(1 - \frac{\pi r^2}{4}\right)^n$$

(b) (2 points) What number of points n should we select so that the probability of misclassification is δ for $\delta \in (0, 1)$?

initial here

Solution: We plug in δ into our previous expression and solve for n:

$$\delta = \left(1 - \frac{\pi r^2}{4}\right)^n \implies n = \frac{\log(\delta)}{\log\left(1 - \frac{\pi r^2}{4}\right)}$$

(c) (2 points) Now instead of a 1NN classifier, we will consider the generalized case of a kNN classifier, for some arbitrary odd value k. Suppose that we again sample n points uniformly from the region $[-1, 1] \times [-1, 1]$ and they follow their labels follow the conditional above. Using this kNN classifier, what will the probability of misclassification of point (0, 0) be?

Solution: The probability of misclassification now will be the probability that fewer than $\frac{k+1}{2}$ points fall inside of the circular region (i.e. a majority of the closest k points are of class 0). Let C be a random variable for the number of points falling inside of the circle. Then the probability of misclassification will be

$$P(C < (k+1)/2) = P(C \leq (k-1)/2) = \sum_{i=0}^{\frac{k-1}{2}} \binom{n}{i} \left(\frac{\pi r^2}{4}\right)^i \left(1 - \frac{\pi r^2}{4}\right)^{n-i}$$

(d) (4 points) Suppose that instead of the estimated conditional distribution shown in the figure, we instead estimate the conditional $P(Y = 1|X) = 0.8$ inside the circular region, and $P(Y = 1|X) = 0.3$ everywhere else. Once again, we sample n points and then look to classify a point at (0, 0) with a 1NN classifier. Assuming the sampled and test point labels both follow this new conditional distribution, what is the probability of misclassification of this point? Hint: consider separately all possible scenarios for misclassification.

Solution: Formally, let $(\mathbf{x}, y)$ denote the point at (0, 0) and let $(\mathbf{x}', y')$ be its nearest neighbor. There are four possible scenarios for misclassification:

1. The point at (0, 0) is class 1, the nearest neighbor is in the circle and is class 0:

$$\begin{aligned} &P(\text{misclassification}) \\ &= P(y = 1 \mid \mathbf{x} \in \text{circle}) P(y' = 0 \mid \mathbf{x}' \in \text{circle})(1 - P(\mathbf{x}_1, \ldots, \mathbf{x}_n \notin \text{circle})) \\ &= 0.16\left(1 - \left(1 - \frac{\pi r^2}{4}\right)^n\right) \end{aligned}$$

2. The point at (0, 0) is class 1, the nearest neighbor is outside the circle and is class 0.

$$\begin{aligned} &P(\text{misclassification}) \\ &= P(y = 1 \mid \mathbf{x} \in \text{circle}) P(y' = 0 \mid \mathbf{x}' \notin \text{circle})(P(\mathbf{x}_1, \ldots, \mathbf{x}_n \notin \text{circle})) \\ &= 0.56\left(1 - \frac{\pi r^2}{4}\right)^n \end{aligned}$$

3. The point at (0, 0) is class 0, the nearest neighbor is in the circle and is class 1.

$$P(\text{misclassification})$$

initial here

$$= P(y = 0 \mid \mathbf{x} \in \text{circle})P(y' = 1 \mid \mathbf{x} \in \text{circle})(1 - P(\mathbf{x}_1, \ldots, \mathbf{x}_n \notin \text{circle}))$$
$$= 0.16\left(1 - \left(1 - \frac{\pi r^2}{4}\right)^n\right)$$

4. The point at (0, 0) is class 0, the nearest neighbor is outside the circle and is class 1.

$$P(\text{misclassification})$$
$$= P(y = 0 \mid \mathbf{x} \in \text{circle})P(y' = 1 \mid \mathbf{x}' \notin \text{circle})(P(\mathbf{x}_1, \ldots, \mathbf{x}_n \notin \text{circle}))$$
$$= 0.06\left(1 - \frac{\pi r^2}{4}\right)^n$$

Adding the four probabilities together:

$$P(\text{misclassification}) = 0.32 - 0.3\left(1 - \frac{\pi r^2}{4}\right)^n$$

initial here

You may use this page to show extra work. Clearly mark your work with the problem number here, and also mention in the problem-specific box that your work is continued here.

initial here

You may use this page to show extra work. Clearly mark your work with the problem number here, and also mention in the problem-specific box that your work is continued here.

initial here

You may use this page to show extra work. Clearly mark your work with the problem number here, and also mention in the problem-specific box that your work is continued here.

initial here

You may use this page to show extra work. Clearly mark your work with the problem number here, and also mention in the problem-specific box that your work is continued here.