Name: ____________________

Supervised learning

1. (6 points) For each question in this section, please select the correct answer(s) and provide a *short* explanation in the box.

Overparameterization

(a) (1 point) Every model with more parameters than number of training data points will have poor generalization.

○ TRUE ○ FALSE

Memorization and nearest neighbours

Suppose we have a data space $\mathcal{X} = \mathbb{R}^d$ and binary labels $\mathcal{Y} = \{-1, 1\}$. Given a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m \subseteq \mathcal{X} \times \mathcal{Y}$, let $f_k(\boldsymbol{x}) = f_k(\boldsymbol{x}; \mathcal{D})$ denote the kNN classifier that makes predictions using the uniform voting rule (i.e., predict whatever is the most common label of the nearest neighbours). In the event of a tie, assume we break ties by always predicting $f_k(\boldsymbol{x}) = 1$.

Recall that we say that the kNN classifier memorizes $\mathcal{D}$ if $f_k(\boldsymbol{x}_i) = y_i$ for all $i = 1, \ldots, m$. We define the memorization capacity of f_k to be the largest integer m_k such that f_k memorizes all datasets of size m_k with distinct $\boldsymbol{x}$ values (i.e., $\boldsymbol{x}_i \neq \boldsymbol{x}_j$ whenever $i \neq j$).

(b) (1 points) We have $m_1 \geqslant m_2 \geqslant m_3$.

○ TRUE ○ FALSE

Stochastic gradient descent

(c) (2 points) Consider ordinary least squares (OLS) on a dataset $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, with n observations that are d-dimensional, and the target vector $\boldsymbol{y} \in \mathbb{R}^n$:

$$L(\boldsymbol{w}) = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2.$$

Suppose we wish to minimize $L(\boldsymbol{w})$ using SGD. Which of the following is true:

[redacted]

◯ SGD randomly selects **rows** of $\boldsymbol{X}$ and the corresponding coordinates of $\boldsymbol{y}$.
◯ SGD randomly selects **columns** of $\boldsymbol{X}$ and the corresponding coordinates of $\boldsymbol{w}$.

[redacted]

Name: ______________________

Approximation of indicator functions

(d) (2 points) Consider the function

$$g(x) = \begin{cases} 1 & \text{if } a \leqslant x \leqslant b \\ 0 & \text{otherwise} \end{cases}$$

defined for all $x \in \mathbb{R}$.

Our goal is to express this step function as a feed forward neural network of the form:

$$f(x) = \operatorname{sign}\left(\sum_{i=1}^{d} \boldsymbol{w}_i^2 \cdot \boldsymbol{z}_i \right)$$

where $\boldsymbol{z} = \text{ReLU}(\boldsymbol{w}^1 \cdot x + \boldsymbol{b})$ is the hidden later output, and we recall that the sign function is defined as: $\operatorname{sign}(a) = 1$ if $a \geqslant 0$ and $\operatorname{sign}(a) = -1$ if $a < 0$.

So $\boldsymbol{w}^1, \boldsymbol{w}^2, \boldsymbol{b} \in \mathbb{R}^d$ are the parameters of the model f. Specifically, letting $\theta = (\boldsymbol{w}^1, \boldsymbol{w}^2, \boldsymbol{b})$ denote all the parameters of f, our goal is to pick θ such that $g(x) = f(x; \theta)$ for all $x \in \mathbb{R}$ (for simplicity you can ignore the edge cases $x = a, b$, though it is possible to handle these cases too).

Note that d is the number of neurons in the hidden layer. What is the minimum number of neurons d needed in order to express g using f?

○ $d = 1$ ○ $d = 2$ ○ $d = 3$ ○ $d = 4$

[redacted]

Name: ____________________

Convergence of Deep Networks

2. (5 points) Consider two neural networks represented by functions, $\hat{y}_i^t = f_{\theta^t}(x_i)$, $\tilde{y}_i^t = g_{\phi^t}(x_i)$, where x_i is the input, $\hat{y}_i^t, \tilde{y}_i^t$ denote the outputs of the neural network at the t^{th} training iteration and θ^t, ϕ^t are the parameters of the network at the t^{th} training iteration. Let the training dataset be $\{x_i, y_i\}_{i=1}^N$.

(a) (3 points) Lets assume that we initialize the weights of the network such that $\hat{y} == \tilde{y}_i^0 \forall i \in [1_i^0\ N]$. Next, we train both the networks to convergence using SGD. Would these two networks have the same performance on the test set sampled from the same distribution as the training set? Please explain.

(b) (2 points) If the answer to the above question is yes, then would the performance be the same on a test set sampled from a different distribution than the training dataset? If the answer was no, then if the initialization condition was stronger, $\hat{y}_i^0 == \tilde{y}_i^0$ for all possible x_i (even beyond the training dataset), then would the two networks converge to parameters that would yield the same accuracy on the test sampled from the same distribution as the training set?

[redacted]

Training of Neural Networks

3. (10 points) Answer the following questions:

   (a) (1 point) Suppose we train a K layer neural network to a supervised learning problem and we find that the network over-fits. Could increasing the number of layers, and consequently the number of parameters of the network reduce over-fitting?

   (b) (3 points) Suppose we find that for a deep neural network, the magnitude of features is decreasing with training iterations and has become very small. The training loss is high. Which would one of the modifications will be helpful and why?: (a) Dropout; (b) BatchNorm; (c) Residual Connections; (d) Increasing the Magnitude of Weights at Initialization; (e) Decreasing the Magnitude of Weights at Initialization; (f) Increasing the learning rate.

   (c) (2 points) Are transformers a generalization of graph neural networks? Please explain.

   (d) (4 points) Choice of Loss Functions: Lets assume we have loss functions – (A) Binary Cross Entropy Loss (pyTorch implementation); (B) Negative Log-Likelihood Loss (pyTorch Implementation); (C) L2 loss; (D); L1 loss. The problems are: (i) Image classification, where an image can only belong to one-class. (ii) Image classification where an image can belong to multiple-classes (e.g., door and cat can be present in the same image); (iii) a linear regression problem where the model is known to be sparse; (iv) a linear regression problem where the model is known to be Gaussian.

[redacted]

Name: ____________________

Disease Diagnosis with Hierarchical Factor Analysis

4. (21 points) Dr. I. N. Ference has recently noticed an unusually high influx of patients exhibiting symptom x and starts to suspect the presence of a hidden variable, disease z_2. Furthermore, Dr. Ference also hypothesizes that the existence of gene z_1 makes a person much more predisposed to con-tract disease z_2. Dr. Ference's hypothesis can be represented using the simplified hierarchical model below, with a single gene of interest and disease:

$$\begin{tikzcd} z_1 \arrow[r] \arrow[rr, bend left] & z_2 \arrow[r] & x \end{tikzcd}$$

Dr. Ference thinks that a reasonable first step is to evaluate the efficacy of this simple hierarchical setup in the case where z_1 is a Gaussian variable (some measurement of how much gene z_1 is expressed) that can cause disease z_2 to be expressed with varying intensities. Then, the effects of z_1 and z_2 both contribute to the intensity of the observed symptom x. In this generative model, the observations x are drawn by the following procedure:

- Sample $z_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$.
- Draw $z_2 \sim \mathcal{N}(wz_1 + \mu_2, \sigma_2^2)$ (i.e. $p(z_2|z_1, \theta) = \mathcal{N}(wz_1 + \mu_2, \sigma_2^2)$, where $\theta = \{w, \mu_2, \sigma_2\}$
- For each observation x, draw $x \sim \mathcal{N}(G\tilde{z} + \mu_3, \sigma_3^2)$, where $\tilde{z} = [z_1, z_2]^T$ and G is a 1×2 weight matrix.

(a) (4 points) Suppose Dr. Ference's colleague supplies their best guess for the posterior $p(z_1, z_2|x)$, which we denote as $\tilde{p}(z_1, z_2)$. Without directly computing the posterior $p(z_1, z_2|x)$, how could we use $\tilde{p}(z_1, z_2)$ to obtain a lower bound on the marginal likelihood $p(x)$?

Solution: Recall that for generic distributions q,

$$\begin{aligned} \mathrm{KL}(q(z_1, z_2)\|p(z_1, z_2|x)) &= -(\mathrm{ELBO}) + p(x) \geqslant 0 \\ -(\mathrm{E}_q[\log p(z_1, z_2, x)] - \mathrm{E}_q[\log q(z_1, z_2)]) + p(x) &\geqslant 0 \\ p(x) &\geqslant \mathrm{E}_q[\log p(z_1, z_2, x)] - \mathrm{E}_q[\log q(z_1, z_2)] \end{aligned}$$

We therefore can compute the ELBO, replacing $q(z_1, z_2)$ in the right hand side of the inequality with $\tilde{p}(z_1, z_2)$ to get $\log p(x) \geqslant \mathrm{E}_{\tilde{p}}[\log p(z_1, z_2, x)] - \mathrm{E}_{\tilde{p}}[\log \tilde{p}(z_1, z_2)]$, which we can use to a lower bound for $p(x)$ by exponentiating:

$$\boxed{p(x) \geqslant e^{\mathrm{E}_{\tilde{p}}[\log p(z_1, z_2, x)] - \mathrm{E}_{\tilde{p}}[\log \tilde{p}(z_1, z_2)]}}$$

Note: We did not take points off for not recognizing the ELBO is a lower bound on $\log p(x)$, not $p(x)$ itself

(b) (6 points) Suppose we are provided with N observations $x_1, \ldots, x_N$ and a general distribution $q(z_1, z_2; \phi_i)$ with variational parameters $\phi_1, \ldots, \phi_N$ that we will use to approximate the posterior $p(z_1, z_2|x_i; \theta)$ with appropriate parameters θ, for $1 \leqslant i \leqslant N$. We can

[redacted]

use the following block coordinate ascent algorithm to find a setting of the variational parameters $\phi_1, \ldots, \phi_N$ that closely approximates the posterior in question:

1. Initialize variational parameters $\phi_1, \ldots, \phi_N$ and parameters θ of p.
2. For each observation x_i, compute the ELBO $\mathcal{L}(x_i; \theta, \phi_i) = \mathbb{E}_q[\log p(z_1, z_2, x_i; \theta)] - \mathbb{E}_q[\log q(z_1, z_2; \phi_i)]$
3. Holding θ fixed, update the variational parameter ϕ_i to maximize the ELBO as shown below:

$$\phi_i = \underset{\phi}{\operatorname{argmax}}\, \mathcal{L}(x_i; \theta, \phi)$$

4. Holding ϕ_i's constant ($1 \leqslant i \leqslant N$), update the variational parameter θ to θ^* that maximizes the sum of the ELBOs as shown below:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} \mathcal{L}(x_i; \theta, \phi_i)$$

5. Repeat steps $2 - 4$ until the ELBO converges

For Step 4 in this procedure to be equivalent to the M-step update in the EM algorithm, what must be true about the distribution $q(z_1, z_2; \phi_i)$? Explain your answer.
Hint: Recall that the goal of the M-step of the EM algorithm is to find new parameters θ that maximize the log-likelihood over all the data - observed and latent - given the previous parameters θ' i.e:

$$\underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} \int\int p(z_1, z_2 | x_i; \theta') \log p(z_1, z_2, x_i; \theta)\, dz_1\, dz_2.$$

Solution:
As shown in class, the EM algorithm is a special case of variational inference corresponding to when $q(z_1, z_2; \phi_i) = p(z_1, z_2 | x; \theta')$. The proof is given below:
Per the hint, recall that the goal of the M-step of the EM algorithm is to update the parameters θ to maximize the log-likelihood of the entire data given the previous parameters θ', i.e:

$$\underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} \int\int p(z_1, z_2 | x_i; \theta') \log p(z_1, z_2, x_i; \theta)\, dz_1\, dz_2$$

and further note that we can rewrite the term inside the maximization as an expectation over $p(z_1, z_2 | x; \theta')$:

$$\mathbb{E}_{p(z_1, z_2 | x; \theta')}[\log p(z_1, z_2, x; \theta)]$$

[redacted]

Name: ______________________________

Now, we can note that this expression is precisely equal to the first term of the ELBO when $q(z_1, z_2; \phi_i) = p(z_1, z_2|x; \theta')$. However, there is an additional term of the ELBO

$$-E_q[\log q(z_1, z_2; \phi_i)] = -E_{p(z_1,z_2|x;\theta')}[\log p(z_1, z_2|x; \theta')]$$

which is dependent on θ', rather than θ. Thus, this is a constant which does not affect the optimal value of θ and the conclusion follows.

(c) (6 points) Suppose Dr. Ference wants to apply the algorithm mentioned in Part (b) using the family of 2D Gaussians $q(z_1, z_2)$ with full covariance matrices Σ. Are there always parameters that make the bound given by the ELBO tight? Justify your answer by either describing a distribution $q(z_1, z_2)$ where the ELBO is tight, or by arguing that the lower bound cannot actually be attained.

You may assume without proof the following identities:

Product of Gaussians (Special): Suppose for random vectors a and b, $p(a) = \mathcal{N}(\mu_1, \Sigma_1)$ and $p(b|a) = \mathcal{N}(Ma + \mu_2, \Sigma_2)$, where M is a weight matrix. Then the product of the two PDFs, $p(a)p(b|a)$, is also Gaussian. The covariance matrix is non-diagonal if M and Σ_2 are non-zero.

Gaussian Conditionals: Suppose we have random vectors a and b, where $[a, b]^T$ follows a multivariate Gaussian distribution. Then the conditional probability $p(a|b)$ is also Gaussian.

◯ Yes ◯ No

Solution: Yes, the ELBO can always be made tight. Since we take $q(z_1, z_2)$ to be the family of 2D Gaussians, it suffices to show that the posterior $p(z_1, z_2|x)$ is also Gaussian.

Let us demonstrate that the PDF of the posterior $p(z_1, z_2|x)$ is Gaussian. First, we note that the joint distribution $p(z_1, z_2, x)$ can be factored as

$$p(z_1, z_2, x) = p(x|z_1, z_2)p(z_2|z_1)p(z_1)$$

Since $p(z_1)$, $p(z_2|z_1)$, and $p(x|z_1, z_2)$ are all Gaussian, we can apply the special product of Gaussians identity twice to get that $p(z_1, z_2, x)$ is also Gaussian with some mean μ' and covariance matrix Σ'.

Now, we can apply the Gaussian conditional identity with $a = [z_1, z_2]^T$ and $b = x$. Since the identity states that the conditional distribution $p(a|b)$ is also Gaussian, it follows that there exists some $q(z_1, z_2) = p(z_1, z_2|x)$ (since we allow q to be an unrestricted multivariate Gaussian) where reverse KL-divergence is 0 and the ELBO is thus a tight bound.

(d) (5 points) Now, suppose Dr. Ference wants to instead use a q that satisfies the mean-field

[redacted]

assumption - that is, a function that can be factored as $q(z_1, z_2) = q(z_1)q(z_2)$. Under this assumption, are there always parameters for $q(z_1, z_2)$ that make the bound given by the ELBO tight? Justify your answer by either describing a distribution $q(z_1, z_2) = q(z_1)q(z_2)$ where the ELBO is tight, or by arguing that the lower bound cannot actually be attained.
◯ Yes ◯ No

Solution: No, there are not always parameters which make the ELBO tight. In order for this to happen, we must have that $\text{KL}(q(z_1, z_2) \| p(z_1, z_2 | x)) = 0$, which only occurs when $q(z_1, z_2) = p(z_1, z_2 | x)$.

For most values for the parameters in p, the posterior $p(z_1, z_2|x)$ will not be able to factorize. For example, consider any case in which $G = 0$ and $w \neq 0$. Since $G = 0$, x is independent of z_1 and z_2 and thus $p(z_1, z_2|x) = p(z_1, z_2)$. Then, since $w \neq 0$, z_1 and z_2 are not independent in p and as a result we cannot factorize p into two independent distributions $q(z_1)$ and $q(z_2)$.

To show this fact more rigorously, let us examine the pdf of the joint distribution $p(z_1, z_2, x)$ and show that the corresponding covariance matrix is non-diagonal. Since we are considering a scenario where $G = 0$, the distribution $p(z_1, z_2, x)$ can be factored as $p(z_1, z_2)p(x)$. Thus, it suffices to show under these settings that the covariance matrix of $p(z_1, z_2)$ is non-diagonal. From the hint in the previous question, we have that

$$p(z_1, z_2) = p(z_1)p(z_2|z_1) = \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ w\mu_1 + \mu_2 \end{bmatrix}, \begin{bmatrix} \sigma_1^{-2} + \sigma_2^{-2}w^2 & -\sigma_2^{-2}w \\ -\sigma_2^{-2}w & \sigma_2^{-2} \end{bmatrix}^{-1}\right)$$

Since $-\sigma_2^{-2}w \neq 0$ and the inverse of a non-diagonal matrix is also non-diagonal, z_1 and z_2 are not independent. It follows that there exists no q such that $p(z_1, z_2|x) = q(z_1)q(z_2)$, which means that for this set of example parameters the ELBO cannot be tight.

[redacted]

Name: ______________________________

PCA as an autoencoder

5. (6 points) An autoencoder learns to compress data from the input features into a low-dimensional representation, and then decompress that representation into an approximation that closely matches the original features. Consider a regularized linear autoencoder with the following objective:

$$\mathcal{L}(U, V) = \sum_{i=1}^{N} \frac{1}{2}\|x_i - UVx_i\|^2 + \lambda_1\|U\|_F^2 + \lambda_2\|V\|_F^2,$$

where x_i is the feature vector of the ith example and U, V are the weights to be learned from the data. What are the values of λ_1, λ_2 that recover PCA? Write down your answers and explanations in the box.

Solution: $\lambda_1 = \lambda_2 = 0$ recovers PCA. If we view PCA as minimizing projection error, it has the same objective as the regularized autoencoder.

[redacted]

Name: ______________________________

T/F Questions

6. (4 points) Each sub-part in this question is a True-False requiring explanation, and is worth 2 points, 1 for a correct T/F, and 1 for a correct explanation.

(a) (2 points) It is possible to find the maximum a posteriori (MAP) sequence under a given recurrent neural network language model with a linear-time algorithm in terms of sequence length n. If true, describe an algorithm. If false, give intuition for why (i.e., a formal proof is not needed).

Solution: False. This is a combinatorial search space—the set of all possible sequences is of size $|V|^n$ where V is your vocabulary and n is the sequence length. Approximate algorithms, such as Beam Search, exist, but solving the overall problem is computationally prohibitive.

(b) (2 points) Alice is training a model with parameters θ to minimize a loss $L(x, y; \theta)$ according to distribution $(x, y) \sim q$. However, she discovers her training data's distribution is corrupted, with sample points x being drawn from a new marginal distribution, $p(x)$ (though the conditional $p(y|x)$ remains unchanged (e.g., $p(y|x) = q(y|x)$). Alice claims she can still learn a true-distribution optimal learner by minimizing (over the corrupted train set) a re-weighted loss:

$$L'(x, y; \theta) = \frac{q(x)}{p(x)} L(x, y; \theta).$$

Is Alice correct? Why or why not?

Solution: Yes, Alice is correct. One can correct for dataset shift by computing a reweighting factor $\beta(x, y) = \frac{q(x,y)}{p(x,y)}$. Here, as we are exclusively subsampling based on x, we have that $p(y|x) = q(y|x)$, so $\frac{q(x,y)}{p(x,y)} = \frac{q(x)}{p(x)}$. See Lecture 11 for more details.

[redacted]

Name: ______________________________

Early Stopping, Gradient Descent, and Implicit Regularization

7. (15 points) In this problem, we will show that early stopping, a commonly used procedure in which optimization over the *train* set is truncated early, induces an implicit regularization effect by mitigating the effect that small singular values of the dataset have on the solution. We will develop this notion through several smaller sub-parts. First, we will work through a new way of looking at the solution to OLS regression, viewed as the result of recursive updates via gradient descent. Next, we will use this formulation to examine how an early-stopped solution exhibits regularization behavior.

(a) (3 points) Suppose, we want to perform OLS on a centered, full-rank dataset $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, with n d-dimensional input samples, $n > d$, and $\boldsymbol{y} \in \mathbb{R}^n$, n 1-D outputs:

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \underbrace{\frac{1}{n}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2}_{L(\boldsymbol{w})}.$$

Suppose we solve this with gradient descent; initializing our weight vector to some value $\boldsymbol{w}_0 = \boldsymbol{0}$, then updating in order to minimize the loss $L(\boldsymbol{w})$ according to a learning rate γ. Show that the update rule describing $\boldsymbol{w}_{t+1}$ (the value of our weight vector in the $t+1$ iteration) in terms of $\boldsymbol{w}_t$ is given by:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \frac{2\gamma}{n}\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{w}_t - \boldsymbol{y}).$$

Solution: First, note that our solution here is of the form $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma\nabla_{\boldsymbol{w}} L|_{\boldsymbol{w}_t}$. So, our real issue here is to determine $\nabla_{\boldsymbol{w}} L$. But this is simple; by matrix-vector calculus, we have

$$\begin{aligned}
\nabla_{\boldsymbol{w}} L &= \nabla_{\boldsymbol{w}}\left(\frac{1}{n}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2\right) & \\
&= \frac{2}{n}\nabla_{\boldsymbol{w}}\left(\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|\right) & \text{by chain rule} \\
&= -\frac{2}{n}\boldsymbol{X}^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) & \text{by derivative of norm} \\
&= \frac{2}{n}\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) & \text{manipulation.}
\end{aligned}$$

Therefore, our update rule is:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \frac{2\gamma}{n}\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{w}_t - \boldsymbol{y}).$$

[redacted]

(b) (4 points) Recall we've initialized $\boldsymbol{w}_0 = \mathbf{0}$. Show by induction that we can write the t-th iteration weight vector $\boldsymbol{w}_t$ as

$$\boldsymbol{w}_t = \frac{2\gamma}{n} \sum_{j=0}^{t-1} \left(\boldsymbol{I} - \frac{2\gamma}{n} \boldsymbol{X}^\top \boldsymbol{X} \right)^j \boldsymbol{X}^\top \boldsymbol{y}.$$

Solution: We will prove this via induction. To establish the base case, note that by part (b), we know that $\boldsymbol{w}_1 = \frac{2\gamma}{n} \boldsymbol{X}^\top \boldsymbol{y}$, which is also equal to $\gamma \frac{2}{n} \sum_{j=0}^{0} \left(\boldsymbol{I} - \gamma \frac{2}{n} \boldsymbol{X}^\top \boldsymbol{X} \right)^j \boldsymbol{X}^\top \boldsymbol{y}$, as any matrix to the zeroth power is the identity. Therefore, the base case holds.

To see the inductive case, note that

$$\begin{aligned}
\boldsymbol{w}_t &= \boldsymbol{w}_{t-1} - \frac{2\gamma}{n} \boldsymbol{X}^\top \left(\boldsymbol{X} \boldsymbol{w}_{t-1} - \boldsymbol{y} \right) \\
&= \left(\boldsymbol{I} - \frac{2\gamma}{n} \boldsymbol{X}^\top \boldsymbol{X} \right) \boldsymbol{w}_{t-1} + \frac{2\gamma}{n} \boldsymbol{X}^\top \boldsymbol{y}.
\end{aligned}$$

Now, assume for the sake of induction that $\boldsymbol{w}_t = \gamma \frac{2}{n} \sum_{j=0}^{t-1} \left(\boldsymbol{I} - \gamma \frac{2}{n} \boldsymbol{X}^\top \boldsymbol{X} \right)^j \boldsymbol{X}^\top \boldsymbol{y}$. Then, we have that

$$\begin{aligned}
\boldsymbol{w}_t &= \left(\boldsymbol{I} - \frac{2\gamma}{n} \boldsymbol{X}^\top \boldsymbol{X} \right) \left(\gamma \frac{2}{n} \sum_{j=0}^{t-2} \left(\boldsymbol{I} - \gamma \frac{2}{n} \boldsymbol{X}^\top \boldsymbol{X} \right)^j \boldsymbol{X}^\top \boldsymbol{y} \right) + \frac{2\gamma}{n} \boldsymbol{X}^\top \boldsymbol{y} \\
&= \gamma \frac{2}{n} \sum_{j=1}^{t-1} \left(\boldsymbol{I} - \gamma \frac{2}{n} \boldsymbol{X}^\top \boldsymbol{X} \right)^j \boldsymbol{X}^\top \boldsymbol{y} + \frac{2\gamma}{n} \boldsymbol{X}^\top \boldsymbol{y} \\
&= \gamma \frac{2}{n} \sum_{j=0}^{t-1} \left(\boldsymbol{I} - \gamma \frac{2}{n} \boldsymbol{X}^\top \boldsymbol{X} \right)^j \boldsymbol{X}^\top \boldsymbol{y},
\end{aligned}$$

as desired.

Thus, as the inductive case holds, and the base case holds, the overall statement holds by induction.

[redacted]

(c) (2 points) For $\gamma > 0$ such that $\|\gamma \frac{2}{n} \boldsymbol{X}^\top \boldsymbol{X}\| < 1$, show that

$$\boldsymbol{w}^* = \lim_{t\to\infty} \boldsymbol{w}_t = \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y}.$$

Recall that for $\boldsymbol{A}$ such that $\|\boldsymbol{A}\| < 1$, $\sum_{i=0}^{\infty} \boldsymbol{A}^i = (\boldsymbol{I} - \boldsymbol{A})^{-1}$.

Solution: At the risk of a slight abuse of notation, let $\boldsymbol{w}_\infty = \lim_{t\to\infty} \boldsymbol{w}_t$. Then, for γ such that $\|\gamma \frac{2}{n} \boldsymbol{X}^\top \boldsymbol{X}\| < 1$ (note that this is feasible—e.g., there exists at least one γ such that this is true—as $\boldsymbol{X}^\top \boldsymbol{X}$ has strictly non-negative eigenvalues), we have that

$$\begin{aligned}
\boldsymbol{w}_\infty &= \lim_{t\to\infty} \gamma \frac{2}{n} \sum_{j=0}^{t-1} \left(\boldsymbol{I} - \gamma \frac{2}{n} \boldsymbol{X}^\top \boldsymbol{X}\right)^j \boldsymbol{X}^\top \boldsymbol{y} \\
&= \gamma \frac{2}{n} \left(\sum_{j=0}^{\infty} \left(I - \gamma \frac{2}{n} \boldsymbol{X}^\top \boldsymbol{X}\right)^j \right) \boldsymbol{X}^\top \boldsymbol{y} \\
&= \gamma \frac{2}{n} \left(I - \left(I - \gamma \frac{2}{n} \boldsymbol{X}^\top \boldsymbol{X}\right)\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y} \\
&= \gamma \frac{2}{n} \left(\gamma \frac{2}{n} \boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y} \\
&= \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y}
\end{aligned}$$

Note that this is also simply the OLS solution to regression, which makes sense as gradient descent converges to the optimal solution for OLS regression.

(d) (4 points) Let us decompose our data matrix $\boldsymbol{X}$ according to singular value decomposition: $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, such that $\boldsymbol{U} \in \mathbb{R}^{n\times n}, \boldsymbol{V} \in \mathbb{R}^{d\times d}$ are orthogonal (e.g., $\boldsymbol{U}\boldsymbol{U}^\top = \boldsymbol{I}$) matrices, and $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times d}$ is a rectangular diagonal matrix with strictly positive entries (generally, you can only assume its entries are non-negative, but recall we've assumed $\boldsymbol{X}$ is full-rank here, so we can assume they are strictly positive). Show that we can express the OLS solution $\boldsymbol{w}^* = \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y}$ as

$$\boldsymbol{w}^* = \sum_{j=1}^{d} \left(\frac{1}{\sigma_j}\right) \left(\boldsymbol{u}_j^\top \boldsymbol{y}\right) \boldsymbol{v}_j,$$

where $\boldsymbol{u}_j, \boldsymbol{v}_k$ indicates the j-th, k-th column of $\boldsymbol{U}, \boldsymbol{V}$, respectively.

[redacted]

Solution: First, note that

$$\begin{aligned}
\boldsymbol{w}^*(\lambda) &= \left(\boldsymbol{V}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\right)^{-1}\boldsymbol{V}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top\boldsymbol{y} \\
&= \left(\boldsymbol{V}\left(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}\right)\boldsymbol{V}^\top\right)^{-1}\boldsymbol{V}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top\boldsymbol{y} \\
&= \boldsymbol{V}\left(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{V}^\top\boldsymbol{V}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top\boldsymbol{y} \\
&= \boldsymbol{V}\left(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top\boldsymbol{y}
\end{aligned}$$

Note in this problem that these matrices are *not* generally commutable, so we cannot, for example, cycle factors of $(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma})^{-1}$ around in this product. Similarly, as $\boldsymbol{\Sigma}$ is not square, we cannot invert it nor is it symmetric.

However, upon examining the structure of $\left(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\Sigma}^\top$, we see

$$\boldsymbol{w}^* = \sum_{j=1}^{r}\frac{1}{\sigma_j}\left(\boldsymbol{u}_j^\top\boldsymbol{y}\right)\boldsymbol{v}_j,$$

as desired.

(e) (2 points) Applying a similar derivation to $\boldsymbol{w}_t$ in (b), we can also represent $\boldsymbol{w}_t$ using SVD of the data matrix. We skip the derivation and directly give the result, which is compared with $\boldsymbol{w}$ of original OLS in (d)

A. Original OLS: $$\boldsymbol{w} = \sum_{j=1}^{d}\frac{1}{\sigma_j}\left(\boldsymbol{u}_j^\top\boldsymbol{y}\right)\boldsymbol{v}_j,$$

B. Early-stopped OLS: $$\boldsymbol{w}_t = \sum_{j=1}^{d}\frac{1-(1-\frac{2\gamma}{n}\sigma_j^2)^t}{\sigma_j}(\boldsymbol{u}_j^\top\boldsymbol{y})\boldsymbol{v}_j.$$

Recall that we typically associate small singular values (and their associated vectors) with noise in the data, rather than signal.

Which of the two weight vectors, A or B, is more likely to provide a solution less dependent on data noise? Explain your answer.

Solution: Option B, as it will favor small singular values less than option A.

[redacted]

Covariate Shift

Name: ____________________

8. (6 points) Consider a covariate shift domain adaptation problem. We have a large number of labeled source domain examples, effectively having access to $P_S(x, y) = P_S(x)P_S(y|x)$. We also have a large number of unlabeled examples from the target domain, thus effectively $P_T(x)$. You can assume that the support of the target distribution is contained within the support of the source distribution. A friend of ours was kind enough to train a Bayes optimal domain classifier for us. But, unfortunately, in training the classifier, they used twice as many target examples as source examples.

(a) **(2 points)** Write down an expression for the friend's domain classifier $Q(d|x)$, $d = S, T$.

(b) **(4 points)** You are given a loss function $\text{Loss}(y, h(x))$, where $h(x)$ is a classifier we wish to estimate. Provide an expression for the target risk of h only as a function of the source distribution and the friend's domain classifier.

[redacted]

Name: ____________________

9. (10 points) [Basics] Let's consider a simple mixture of two spherical Gaussians model, i.e.,

$$P(x;\theta) = \pi_1 N(x;\mu_1,\sigma_1^2 I) + \pi_2 N(x;\mu_2,\sigma_2^2 I)$$

where, initially, $\mu_1 = [1,1]^T$, $\mu_2 = [1,-1]^T$, $\sigma_1 = 1/4$, $\sigma_2 = 1$, and $\pi_1 = \pi_2 = 0.5$. This initialization is also shown graphically in Figure 1. Define

$$p(j|x^{(i)}) = \frac{\pi_j N(x^{(i)};\mu_j,\sigma_j^2 I)}{\sum_{k=1,2}\pi_k N(x^{(i)};\mu_k,\sigma_k^2 I)}$$

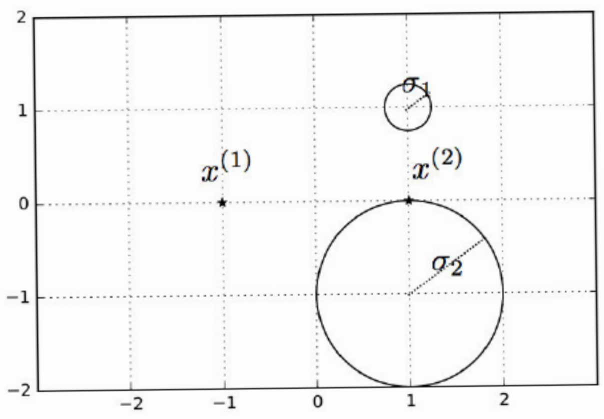


Figure 1: Initialization of the two component mixture model along with two data points

(a) **(4 points)** Consider the first E-step of the EM algorithm. Please assign $<$, $\approx$, or $>$ to each empty relation between the probabilities.

$p(1\|x^{(1)})$	( )	$p(2\|x^{(1)})$
$p(1\|x^{(2)})$	( )	$p(2\|x^{(2)})$
$p(2\|x^{(1)})$	( )	$p(2\|x^{(2)})$
$\sum_{i=1,2} p(1\|x^{(i)})$	( )	1

(b) **(6 points)** Consider the first M-step of the EM algorithm. Which of the following statements are true for $\mu_2 = [\mu_{21},\mu_{22}]^T$ after the first M-step? Check all that apply.

( ) $\mu_{21} < 0$
( ) $\mu_{21} = 0$
( ) $\mu_{22} < 0$ 2nd coordinate
( ) $\mu_{22} = 0$ 2nd coordinate

Briefly justify your answer to whether $\mu_{21} < 0$

[redacted]

Name: ____________________

10. (12 points) Let's look at a basic VAE model, abstractly, and figure out some properties of the associated ELBO criterion. To this end, let the generative model be $P_0(z)P(x|z)$, where $P_0(z)$ remains fixed. Our posterior approximation is $Q(z|x)$ (typically parametric) which we have to set conditioned on each observed x. We also assume the data distribution $P_d(x)$ from which the training data is sampled from. You can think of this as assuming that we have a very large training set, effectively the same as using the distribution $P_d(x)$ directly. The ELBO criterion in this case is

$$\text{ELBO}(Q; P) = E_{x \in P_d(\cdot)} \left[E_{z \sim Q(\cdot|x)} \log P(x|z) + \text{KL}(Q(\cdot|x) \| P_0(\cdot)) \right]$$

(a) **(2 points)** Suppose we place no constraints on the form that the posterior distribution $Q(z|x)$ takes and find $Q^* = \arg\max_Q \text{ELBO}(Q; P)$. Provide an expression for $Q^*(z|x)$.

(b) **(3 points)** Briefly explain why the solution $Q^*(z|x)$ does not simply concentrate around a single point $z^* = \arg\max_z \log P(x|z)$ for each x?

(c) **(3 points)** Suppose we have set $Q(z|x)$, not necessarily optimally, and find $P^* = \arg\max_{P_{x|z}} \text{ELBO}(Q; P)$. Provide an expression for the resulting $P^*(x|z)$.

(d) **(4 points)** Assume now that $P(x|z) = N(x; \mu_z, 1)$, $z = 0, 1$, and $x \in \mathbb{R}$ (scalar). $P_0(z) = 0.5$ for $z = 0, 1$. In other words, we are estimating a simple equally weighted mixture of two Gaussians with unit variances. Describe what family of restricted posterior approximations should we use so that the ELBO estimation criterion would reduce to finding means μ_z, $z = 0, 1$, as in $k = 2$ means clustering?

[redacted]

Let us Do Regression

11. (34 points) For the purpose of this question, we will consider data being generated as per a *mixture model*. Specifically, let $X \in \mathbb{R}$ denote the random variable representing the features and $Y \in \mathbb{R}$ denote the target or label that we would like to *predict* using the features.
We assume that random variable X is generated as per mixture of m-Gaussian distributions with ith mixture component being Gaussian with mean $\mu_i \in \mathbb{R}$ and variance $\sigma \geqslant 0$, and $\overset{2}{\underset{i}{t}}$he probability of ith mixture being p_i with $1 \leqslant i \leqslant m$. To put it another way, to generate a sample of random variable X:

- we sample random variable π which has multinomial distribution on $\{1, \ldots, m\}$ such that $\mathbb{P}(\pi = i) = p_i$, for $1 \leqslant i \leqslant m$
- if the outcome of the multinomial is i, that is $\pi = i$, then we generate a sample from a Gaussian distribution with mean μ_i and variance $\sigma\,{}^2_i$

Given X, we generate Y as follows: if $\pi = i$, then $Y = w_i X + \epsilon$, where $w_i \in \mathbb{R}$ is a fixed parameter associated with mixture component i and ϵ is independent Gaussian with mean 0 and variance 1.

(a) Let us make sure that we understand the setup by answering few simple questions.

1. What is the probability density function of X? *Note*: the probability density function of a Gaussian distribution with mean μ and variance σ^2 is $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$.

Solution:

$$\mathbb{P}(X = x) = \sum_{i=1}^{m} \mathbb{P}(X = x|\pi = i)\mathbb{P}(\pi = i) = \sum_{i=1}^{m} \frac{p_i}{\sqrt{2\pi}\sigma_i} \exp(\frac{-1}{2\sigma_i^2}(x - \mu_i)^2)$$

2. Suppose you observe that $X = x$, what is the likelihood that it came from mixture component i? That is, compute $\mathbb{P}(\pi = i|X = x)$.

Solution:

$$\mathbb{P}(\pi = i|X = x) = \frac{\mathbb{P}(\pi = i, X = x)}{\mathbb{P}(X = x)} = \frac{\frac{p_i}{\sigma_i} \exp(\frac{-1}{2\sigma_i^2}(X - \mu_i)^2)}{\sum_{j=1}^{m} \frac{p_j}{\sigma_j} \exp(\frac{-1}{2\sigma_j^2}(X - \mu_j)^2)}$$

(b) Next, let us suppose that you know that data is generated as per the above described distribution. Let $f^* : \mathbb{R} \to \mathbb{R}$ be such that it minimizes $\mathbb{E}[(Y - f(X))^2]$ over all choices of function $f : \mathbb{R} \to \mathbb{R}$. Please provide an explicit form of f^* using your knowledge of the joint distribution of X, Y.
Note: if you didn't work out $\mathbb{P}(\pi = i|X = x)$ in (a).2, you can use the notation $\mathbb{P}(\pi = i|X = x)$ in your solution for (b) instead of its explicit form.

[redacted]

Name: ______________________________

Solution:

$$f^* = \mathbb{E}(Y|X) = \mathbb{E}_{\pi|X}[\mathbb{E}(Y|\pi, X)] = \sum_{i=1}^{m} \mathbb{P}(\pi = i|X)\mathbb{E}(Y|\pi = i, X) = \sum_{i=1}^{m} \frac{\mathbb{P}(\pi = i, X)}{\mathbb{P}(X)} w_i X$$

$$= \frac{\sum_{i=1}^{m} \frac{p_i}{\sigma_i} \exp(\frac{-1}{2\sigma_i^2}(X - \mu_i)^2) w_i X}{\sum_{j=1}^{m} \frac{p_j}{\sigma_j} \exp(\frac{-1}{2\sigma_j^2}(X - \mu_j)^2)}$$

(c) Now, suppose we do not have the knowledge of the joint distribution of X, Y. But we observe N data points, (x_n, y_n) for $1 \leqslant n \leqslant N$. Using these observations, we would like to identify f^*. As a start, we would like to understand whether simple linear regression would be a good idea or not. That is, we want to find a function $f(x) = a_N^* x + b_N^*$ so that (a_N^*, b_N^*) are solutions to

$$\text{minimize} \sum_{n=1}^{N} (y_n - a x_n - b)^2 \quad \text{over} \quad a, b \in \mathbb{R}.$$

Identify limiting quantities, $a_\infty^* = \lim_{N\to\infty} a_N^*$ as well as $b_\infty^* = \lim_{N\to\infty} b_N^*$. You may write your results using statistical quantities of X and Y (e.g. expectation, variance, etc).

Solution: This is a simple linear regression problem, the solutions for a_N^* and b_N^* are given by

$$a_N^* = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}, \; b_N^* = \bar{y} - a_N^* \bar{x}.$$

When $N \to \infty$,

$$\bar{x} \to \mathbb{E}(X),\; \bar{y} \to \mathbb{E}(Y),\; \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y}) \to \text{Cov}(X, Y) \text{ and } \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 \to \text{Var}(X).$$

So we have

$$a_\infty^* = \frac{\text{Cov}(X, Y)}{\text{Var}(X)} \text{ and } b_\infty^* = \mathbb{E}(Y) - a_\infty^* \mathbb{E}(X).$$

(d) Given that we understand a_∞^* and b_∞^*, let us examine the performance of linear regression (assuming $N = \infty$). To that end, let us consider a concrete scenario where $m = 2$, $p_1 = p_2 = 1/2$, $w_1 = 1$, $w_2 = -1$, $\mu_1 = \theta$, $\mu_2 = -\theta$ for some $\theta > 0$ and $\sigma_1 = \sigma_2 = 1$.
What are the values of a_∞^* and b_∞^* for this specific setting? Will the linear estimator trained with infinitely many observations yield good predictions for this setting (yes/no)?

Solution: We have

$$\mathbb{E}(XY) = \mathbb{E}_\pi \mathbb{E}(XY|\pi) = \frac{1}{2}\mathbb{E}(XY|\pi = 1) + \frac{1}{2}\mathbb{E}(XY|\pi = 2) = \frac{1}{2}\mathbb{E}(X^2|\pi = 1) + \frac{1}{2}\mathbb{E}(-X^2|\pi = 2) = 0.$$

[redacted]

Since $\mathbb{E}(X) = 0$, we have $\mathrm{Cov}(X, Y) = \mathbb{E}(XY) - \mathbb{E}(X)\mathbb{E}(Y) = 0$. Hence

$$a_\infty^* = 0 \text{ and } b_\infty^* = \mathbb{E}(Y) = \theta.$$

This means that linear estimator is not a good choice even with infinitely many observations.

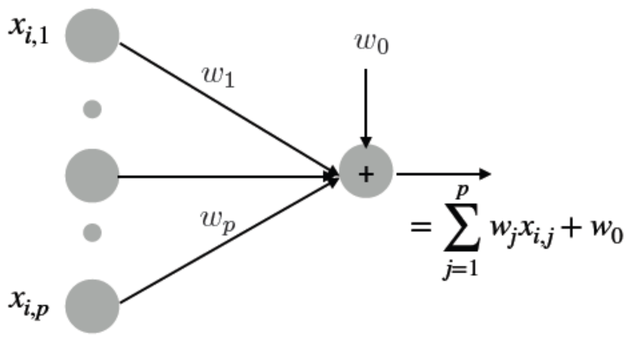
[redacted]

Name: ____________________

"Dropout" Linear Regression

12. (22 points) Dropout is a technique developed to regularize neural networks by randomly setting the output of some nodes to 0 during training. In this part, we aim to explore its effect in the case of ordinary linear regression. Consider a dataset of n training examples, $\{(x_i, y_i)\}_{i=1}^n$, where $y_i \in \mathbb{R}$ is the target value and $x_i \in \mathbb{R}^p$ is the p-dimensional input feature vector. We use the notation $x_{i,k}$ to represent the kth entry of x_i.

In general, one could view linear regression as a 1-layer (i.e. no hidden layer) *linear* network.



Recall that the objective of ordinary linear regression is to minimize the squared loss, i.e. solve $\text{argmin}_w L(w)$, where the loss function $L(w)$ is

$$L(w) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - w_0 - \sum_{j=1}^{p} w_j x_{i,j}\right)^2 \tag{1}$$

With this 1-layer linear network setup, it is now easy to apply the general "dropout" principle: for each input node corresponding to the input feature, $x_{i,k}$, we set the value to 0 with probability q (independent of the other nodes). With probability $1-q$, we scale it by s. Formally, instead of using $x_{i,k}$ at the corresponding node in the input layer, we use $x_{i,k}I_{i,k}$, where each random variable $I_{i,k}$ is IID, and

$$I_{i,k} = \begin{cases} 0 & \text{with probability } q \\ s & \text{with probability } 1-q \end{cases}$$

(a) (4 points) On average, we want the "dropout" input $x_{i,k}I_{i,k}$ to be the same as $x_{i,k}$. What should the scale constant s be in order to make sure $\mathbb{E}[x_{i,k}I_{i,k}] = x_{i,k}$?

Solution:

$$\mathbb{E}[x_{i,k}I_{i,k}] = q \times 0 + (1-q) \times s \times x_{i,k} = (1-q)sx_{i,k}.$$

Therefore, $s = \frac{1}{1-q}$

(b) (4 points) Based on the problem description and the loss defined in Equation (1), write down the new dropout squared loss $L^D(w)$ in terms of the dataset $\{(x_i, y_i)\}_{i=1}^n$, weights w, and the random variables $\{I_{i,k}\}_{1\leqslant i\leqslant n, 1\leqslant k\leqslant p}$.

[redacted]

Solution:

$$L^D(w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - w_0 - \sum_{j=1}^{p} w_j x_{i,j} I_{i,k})^2$$

(c) (4 points) Recall that for ridge regression, one minimizes the regularized squared loss:

$$L^R(w) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - w_0 - \sum_{j=1}^{p} w_j x_{i,j}\right)^2 + \lambda \sum_{j=1}^{p} w_j^2$$

Calculate the gradient for the regularized squared loss, $L^R(w)$ with respect to a particular parameter w_k ($k \neq 0$). That is, compute $\frac{\partial L^R(w)}{\partial w_k}$.

Solution:

$$\frac{\partial L^R(w)}{\partial w_k} = \frac{1}{n}\sum_{i=1}^{n} -2[(y_i - w_0 - \sum_{j=1}^{p} w_j x_{i,j})x_{i,k}] + 2\lambda w_k$$

(d) (10 points) We now investigate the relationship between this dropout formulation and ridge regression. To this end, we provide the following background information.

Background (stochastic optimization): suppose we want to minimize $f(w)$, where f is a strictly convex function. Let w^* be the optimal solution.

- The usual gradient descent algorithms suggest that at each time step t,

$$w_{t+1} = w_t - \alpha_t \frac{\partial f(w_t)}{\partial w}.$$

  With a proper step size α_t, we have $w_t \to w^*$ as $t \to \infty$.
- Consider the following generalized gradient descent algorithm.

$$w_{t+1} = w_t - \alpha_t g(w_t).$$

  Suppose that $g(w)$ is a (possibly randomized) function such that $\mathbb{E}[g(w)] = \frac{\partial f(w)}{\partial w}$ for any w (i.e., the expected value of $g(w)$ equals the gradient of $f(w)$). Then, with proper step size α_t, w_t also converges to w^* as $t \to \infty$.

For the purposes of this problem, assume that for each $1 \leqslant j \leqslant p$, $\frac{1}{n}\sum_{i=1}^{n} x_{i,j}^2 = 1$ (i.e. our features are standardized).

Using the information given above, as well as your answers in (a)-(c), argue that running gradient descent on the dropout squared loss $L^D(w)$ converges to w^*, where w^* is the solution to ridge regression with a particular λ. In addition, identify the value λ in terms of q and the dataset $\{(x_i, y_i)\}_{i=1}^{n}$.

[redacted]

Solution: We first compute the gradient of $L^D(w)$ w.r.t w_k.

$$\begin{aligned}\frac{\partial L^D(w)}{\partial w_k} &= \frac{1}{n}\sum_{i=1}^{n} -2[(y_i - w_0 - \sum_{j=1}^{p} w_j x_{i,j} I_{i,j})x_{i,k} I_{i,k}] \\ &= \frac{1}{n}\sum_{i=1}^{n} -2[(y_i - w_0)x_{i,k} I_{i,k} - (\sum_{j=1}^{p} w_j x_{i,j} I_{i,j})x_{i,k} I_{i,k}].\end{aligned}$$

Then, we have

$$\begin{aligned}E[\frac{\partial L^D(w)}{\partial w_k}] &= \frac{1}{n}\sum_{i=1}^{n} -2[(y_i - w_0)x_{i,k} - x_{i,k}(\sum_{j=1}^{p} w_j x_{i,j} E[I_{i,j} I_{i,k}])] \\ &= \frac{1}{n}\sum_{i=1}^{n} -2[(y_i - w_0)x_{i,k} - x_{i,k}(w_k x_{i,k}\frac{1}{1-q} + \sum_{j=1, j\neq k}^{p} w_j x_{i,j})] \\ &= \frac{1}{n}\sum_{i=1}^{n} -2[(y_i - w_0 - \sum_{j=1}^{p} w_j x_{i,j})x_{i,k}] \\ &\quad + (\frac{2q}{(1-q)n}\sum_{i=1}^{n} x_{i,k}^2)w_k.\end{aligned}$$

Comparing this with the gradient of ridge regression from (c), and using the background information provided, we see that the dropout linear regression effectively solves the ridge regression with λ being $\frac{q}{1-q}$ for each w_k^2 term. That is, gradient descent on $L^D(w)$ converges to w^*, where w^* solves the following ridge regression

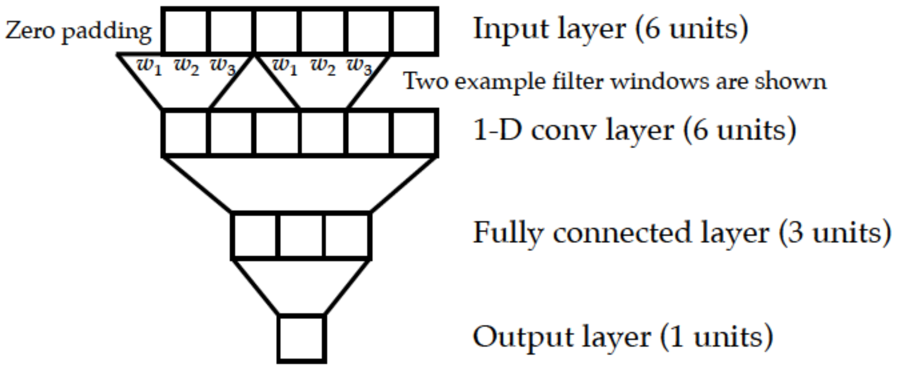
$$L^R(w) = \frac{1}{n}\sum_{i=1}^{n}\Big(y_i - w_0 - \sum_{j=1}^{p} w_j x_{i,j}\Big)^2 + \sum_{j=1}^{p}\Big(\frac{q}{1-q}\Big)w_j^2$$

[redacted]

Name: ______________________

Convolutional Neural Networks

13. (13 points)In this question, we'll first consider the convolutional neural network (CNN) architecture below. The input layer has six neurons. It is followed by a one-dimensional convolutional layer that has one filter with a window size of 3, a stride length of 1, and is padded by a single zero on each side. This is followed by a fully connected layer, and a single output unit.



(a) (3 points) How many parameters are there in the convolutional layer?

Solution: Three or four, depending on if there is a bias.

We will now try and express a convolutional layer as a special case of a fully-connected layer. We express the weights in a fully connected layer as a weight matrix W where $z = Wx$. Here, x is the input and z is the output (before activation), both expressed as column vectors.

(b) (5 points) Suppose the convolutional filter weights are $[w_1, w_2, w_3]$. Write the corresponding fully-connected weight matrix in terms of w_1, w_2, and w_3. Hint: many of the elements in the matrix will be repeated.

Solution:

$$\begin{bmatrix} w_2 & w_3 & 0 & 0 & 0 & 0 \\ w_1 & w_2 & w_3 & 0 & 0 & 0 \\ 0 & w_1 & w_2 & w_3 & 0 & 0 \\ 0 & 0 & w_1 & w_2 & w_3 & 0 \\ 0 & 0 & 0 & w_1 & w_2 & w_3 \\ 0 & 0 & 0 & 0 & w_1 & w_2 \end{bmatrix}$$

(c) (5 points) If the convolutional layer used a stride length of 2 (with the starting window centered on the first unit of the input), what would the weight matrix be instead? Hint: the convolutional layer now has only three units.

Solution:

$$\begin{bmatrix} w_2 & w_3 & 0 & 0 & 0 & 0 \\ 0 & w_1 & w_2 & w_3 & 0 & 0 \\ 0 & 0 & 0 & w_1 & w_2 & w_3 \end{bmatrix}$$

[redacted]

Neural Network Approximation and Representation

14. (a) (6 points) A piece-wise constant function $h(x) : \mathbb{R} \longrightarrow \mathbb{R}$ is plotted in Figure 1, formally written as

$$h(x) = \begin{cases} -2, & \text{if } x \leqslant -2 \\ 1, & \text{if } -2 < x \leqslant -1 \\ -2, & \text{if } -1 < x \leqslant 1 \\ 3, & \text{if } 1 < x \end{cases}$$

We know $h(x)$ can be approximated by a one-hidden-layer neural network arbitrarily well. Now you will approximate $h(x)$ by hand with a neural network in Figure 2 below, with one input neuron, one hidden layer and one output neuron. The hidden neurons use sigmoid activation. The activation of the output is simply linear. Find a possible solution to the biases b1 through b4, and weights W1 through W3.

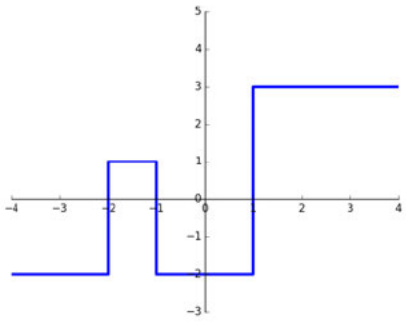


Figure 1: plot of $y = h(x)$

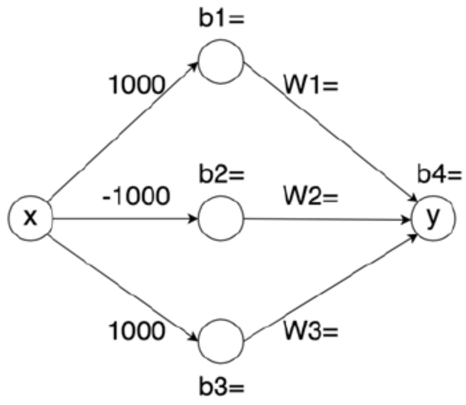


Figure 2: Neural Network to approximate $y = h(x)$. The numbers 1000, −1000, 1000 on the edges are the corresponding weights. They are made large intentionally.

Write your solution (values for b1 through b4, and W1 through W3) here:

Solution:
possible solutions:
$[b_1, b_2, b_3, b_4], [W1, W2, W3]$ = [1000, -2000, -1000, 1], [-3, -3, 5]
$[b_1, b_2, b_3, b_4], [W1, W2, W3]$ = [2000, 1000, 1000, 3], [3, -5, -3]
$[b_1, b_2, b_3, b_4], [W1, W2, W3]$ = [2000, -1000, -1000, -5], [3,3,5]

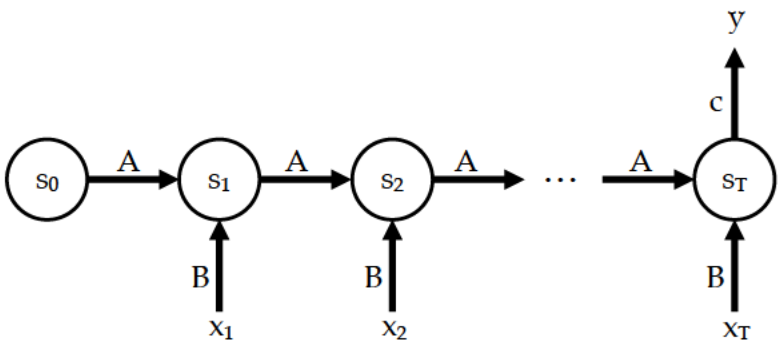
$(b_1, W1)$ and $(b_3, W3)$ are interchangeable.

[redacted]

Now we'll investigate how a simple recurrent neural network (RNN) can represent two different linear models. We use the RNN architecture below to classify an input text according to its sentiment (positive or negative). Suppose all possible words are listed in a vocabulary called V. Let the number of words in the input text (including repeats) be T.

We break the input text apart into a sequence of words, and we represent the kth word as a column vector x_k of length $|V|$. The ith element of x_k is a 1 if the word is the ith word in the vocabulary. All other elements are zero. The RNN maintains a hidden state s, which is also a column vector.

After setting the initial state s_0 to be some value, we pass in x_1, then x_2, and so on, updating the hidden state. The binary output y is a function of the hidden state after all T words have been passed as input. These equations and a simplified diagram of the RNN is given below.



$$s_t = \mathrm{ReLU}(A\, s_{t-1} + B\, x_t)$$
$$y = \mathrm{sign}(c^T s_T + d)$$

where $A \in \mathbb{R}^{|s|\times|s|}$, $B \in \mathbb{R}^{|s|\times|V|}$, $c \in \mathbb{R}^{|s|}$, $d \in \mathbb{R}$, and $\mathrm{sign}(a) = \begin{cases} 1, & \text{if } a \geqslant 0 \\ 0, & \text{otherwise} \end{cases}$

(b) (6 points) It is possible to represent the entire sentence at once, using a single bag of words column vector z. Here, z has length $|V|$ and the ith element of z is a count of how many times the ith word of the vocabulary appears in the input text. A simple model to classify text sentiment is a linear model on this z, i.e.

$$y = \mathrm{sign}(\alpha^T z)$$

where α is some column vector in $\mathbb{R}^{|V|}$.

Suppose we want the output of the RNN we described earlier to match the output of this linear model. Describe a way of setting the RNN weights (A, B, and c), bias d, and initial state s_0 in terms of α to do this.

Solution: We make the hidden state s have size $|V|$. One setting that would work is $s_0 = 0^{|V|}$, $A = I_{|V|}$, $B = I_{|V|}$, $c = \alpha$, $d = 0$.

[redacted]

(c) (6 points) Suppose we instead define the bag of words vector z using indicators. That is, the ith element of z is a 1 if the ith word appears anywhere in the input text and is a 0 otherwise.

Suppose we want the output of the RNN we described earlier to match the output of the linear model on this newly defined z. Describe a way of setting the RNN weights (A, B, and c), bias d, and initial state s_0 in terms of α to do this.

Solution: We again make the hidden state s have size $|V|$. One setting that would work here is $s_0 = 1^{|V|}$, $A = I_{|V|}$, $B = -I_{|V|}$, $c = -\alpha$, $d = \sum_i \alpha_i$.

[redacted]

Time-Varying Bayesian Regression

15. (20 points) Consider a data set $\{(x_t, y_t)\}_{t=1}^T = \{(x_1, y_1), ..., (x_T, y_T)\}$, consisting of scalar features $x_1, x_2, ..., x_T$ and scalar observations $y_1, y_2, ..., y_T$. You can think of x_t and y_t as representing the feature and observation at timestep t. There is only one feature and observation at each timestep t. The goal is to find a model that fits this data. Based on prior knowledge about the problem, you decide to use the following time-varying Bayesian model:

- The initial parameter w_0 is distributed as a standard Gaussian. Each other parameter w_t is distributed as w_{t-1} plus independent noise from a standard Gaussian, for $t = 1, ..., T$. (That is, the initial parameter value gets noisier with time.)
- The observation y_t is distributed as $w_t x_t$ plus independent noise from a standard Gaussian, for $t = 1, ...T$.

Formally:

$$w_0 \sim N(0, 1)$$

$$w_t = w_{t-1} + \eta_t, \qquad \eta_t \sim N(0, 1), \qquad t = 1, ..., T$$

$$y_t = w_t x_t + \epsilon_t, \qquad \epsilon_t \sim N(0, 1), \qquad t = 1, ..., T$$

where w_0, $\{\eta_t\}$, and $\{\epsilon_t\}$ are independent. (Note that, if you had $\eta_t = 0$ for all $t = 1, ..., T$, then you would be back to the usual Bayesian regression setting.)

You want to estimate the vector of parameters $\mathbf{w} := [w_0, w_1, ..., w_T]^T$ as the expected value of $\mathbf{w}$ according to the posterior distribution $p(\mathbf{w}|\{(x_t, y_t)\}_{t=1}^T)$. In this setting, the posterior distribution turns out to be a multivariate Gaussian, so its expected value is equal to the vector of parameters that maximizes the probability density function; i.e. the mode of the posterior distribution. Therefore, to estimate $\mathbf{w}$, it suffices to compute the mode of the posterior distribution. This question focuses on computing this mode.

(a) (4 points) Calculate the prior distribution of the parameters, $p(w_0, w_1, ..., w_T)$. Feel free to ignore proportionality constants in your answer.

Solution:

$$\begin{aligned} p(w_0, w_1, ..., w_T) &= p(w_0) \prod_{t=1}^T p(w_t | w_{t-1}) \\ &= N(w_0; 0, 1) \prod_{t=1}^T N(w_t; w_{t-1}, 1) \\ &= C_1 \exp\left(-\frac{1}{2}\left(w_0^2 + \sum_{t=1}^T (w_t - w_{t-1})^2\right)\right) \end{aligned}$$

(b) (4 points) Calculate the set of parameters $w_0, w_1, ..., w_T$ that maximizes $p(w_0, w_1, ..., w_T)$; i.e. the mode of the prior.

[redacted]

Name: ______________________________

Solution:

$$\begin{aligned}\operatorname*{argmax}_{\mathbf{w}} p(w_0, w_1, ..., w_T) &= \operatorname*{argmax}_{\mathbf{w}} \log p(w_0, w_1, ..., w_T) \\ &= \operatorname*{argmin}_{\mathbf{w}} w_0^2 + \sum_{t=1}^{T} (w_t - w_{t-1})^2\end{aligned}$$

The expression above must be non-negative, so it is clearly minimized by $w_0 = w_1 = ... = w_n = 0$, which sets it to zero.

(c) (6 points) Calculate the posterior distribution of the parameters $p(w_0, w_1, ..., w_T | \{(x_t, y_t)\}_{t=1}^{T})$. Feel free to ignore proportionality constants in your answer.

Solution:

$$\begin{aligned}p(w_0, w_1, ..., w_T | \{(x_t, y_t)\}_{t=1}^{T}) &= C_2 p(\{(x_t, y_t)\}_{t=1}^{T} | w_0, w_1, ..., w_T) p(w_0, w_1, ..., w_T) \\ &= C_2 \left(\prod_{t=1}^{T} p(y_t | x_t, w_t) \right) p(w_0, w_1, ..., w_T) \\ &= C_3 \exp\left(-\frac{1}{2} \left(w_0^2 + \sum_{t=1}^{T} [(y_t - w_t x_t)^2 + (w_t - w_{t-1})^2] \right) \right)\end{aligned}$$

[redacted]

(d) (6 points) Assume $x_t > 0$ for all t. Show how to calculate the set of parameters $w_0, w_1, ..., w_T$ that maximizes $p(w_0, w_1, ..., w_T | \{(x_t, y_t)\}_{t=1}^T)$; i.e. the mode of the posterior. You may assume you have access to a linear system solver to solve for v in $Av = b$, for an invertible matrix A and a vector b. If your answer invokes this solver, clearly state what A and b are and why A is invertible.

Hint: If a matrix A is symmetric and for each row i, $A_{ii} > \sum_{j \neq i} |A_{ij}|$, then A is invertible.

Solution:

$$\begin{aligned}\operatorname*{argmax}_{\mathbf{w}} p(w_0, w_1, ..., w_T | \{x_t, y_t\}_1^T) &= \operatorname*{argmax}_{\mathbf{w}} \log p(w_0, w_1, ..., w_T | \{x_t, y_t\}_1^T) \\ &= \operatorname*{argmin}_{\mathbf{w}} w_0^2 + \sum_{t=1}^{T} [(y_t - w_t x_t)^2 + (w_t - w_{t-1})^2]\end{aligned}$$

Denote the last term L. Note that L is convex in each w_t, $0 \leqslant t \leqslant T$. Take derivatives and set them to zero:

$$\frac{\partial L}{\partial w_0} = 2w_0 - 2(w_1 - w_0) = 0 \Longrightarrow 2w_0 - w_1 = 0$$

$$\frac{\partial L}{\partial w_T} = -2(y_T - w_t x_t)x_T + 2(w_T - w_{T-1}) = 0 \Longrightarrow -w_{T-1} + (x_T^2 + 1)w_T = x_T y_T$$

For $t = 1, ..., T-1$,

$$\begin{aligned}\frac{\partial L}{\partial w_t} &= -2(y_t - w_t x_t)x_t + 2(w_t - w_{t-1}) - 2(w_{t+1} - w_t) = 0 \\ &\Longrightarrow -w_{t-1} + (x_t^2 + 2)w_t - w_{t+1} = x_t y_t\end{aligned}$$

Plug this system of equations in $w_0, ..., w_T$ in the solver in order to get a solution. Let A be the matrix corresponding to this system of equations in $w_0, ..., w_T$. For simplicity, let the rows and columns of A be indexed starting at 0. Then the matrix has the following rows:

- On row 0 (corresponding to $\frac{\partial L}{\partial w_0}$): $A_{0,0} = 2$, $A_{0,1} = -1$, all other elements zero.
- On rows 1 ... $T-1$ (corresponding to $\frac{\partial L}{\partial w_t}$ for $0 < t < T$): $A_{t,t-1} = -1$, $A_{t,t} = x_t^2 + 2$, $A_{t,t+1} = -1$, all other elements zero.
- On row T (corresponding to $\frac{\partial L}{\partial w_T}$): $A_{T,T-1} = -1$, $A_{T,T} = x_T^2 + 1$, all other elements zero.

Clearly, if $x_t > 0$ for all t, on every row the diagonal element is larger than the sum of the absolute values of all other elements. A is also symmetric. Therefore, A is invertible, so there exists a unique solution to the system of equations. So it is possible to find $w_0, w_1, ..., w_T$.

[redacted]

Name: ______________________________

Classification (Yes-no Questions)

16. (6 points) For each question in this section, please select the correct answer and provide a *short* explanation in the box.

(a) (2 points) Recall that a risk function is defined as the expected loss of the classifier $h \in \mathcal{H}$ with respect to the data distribution $\mathbb{P}$ over $\mathcal{X} \times \mathcal{Y}$, that is,

$$L(h) := \mathbb{E}[\ell(h, X, Y)]$$

For the 0/1 loss, defined as $\ell_{0/1} = \begin{cases} 1\,, \text{if } h(X) \neq Y \\ 0\,, \text{if } h(X) = Y \end{cases}$, the risk is: $L(h) = \mathbb{P}(h(X) \neq Y)$.

○ Yes ○ No

Solution: Yes.
$L(h) = \mathbb{E}[\ell_{0/1}(h, X, Y)] = 1 \times \mathbb{P}(h(X) \neq Y) + 0 \times \mathbb{P}(h(X) = Y) = \mathbb{P}(h(X) \neq Y)$

(b) (2 points) With enough training data, the training error of a nearest neighbor classifier always goes down to zero.

○ Yes ○ No

Solution: No. While the training error always goes to 0 for 1-nearest neighbor, for the general case, the training error is non-zero. (We also give full credit if the answer explicitly assumes 1-NN and gives correct reasoning.)

(c) (2 points) In general, for small training sets, we are likely to reduce the bias of the classifier by adding a regularization penalty to the loss function.

○ Yes ○ No

Solution: No. Regularization prevents over-fitting. It reduces the variance of the estimator in favor of introducing some bias.

[redacted]

Classification (Multiple Choice)

17. (3 points) For each question in this section, please select the correct answer and provide a *one-sentence* explanation in the box.

Loss functions

(a) (3 points, Check all that apply.) Suppose we are building a linear classification model $h(x) = w^T x$ on linearly separable data, minimizing which of the loss functions below will force $|w| \to \infty$?

○ Logistic Loss: $L(y, h(x)) = \log(1 + \exp(-yh(x)))$

○ Hinge Loss: $L(y, h(x)) = \max\{0, 1 - yh(x)\}$

○ 0 -1 Loss $L(y, h(x)) = \begin{cases} 0\,, \text{if } y = h(x) \\ 1\,, \text{if } y \neq h(x) \end{cases}$

○ Exponential loss: $L(y, h(x)) = \exp(-yh(x))$

Solution: Exponential loss and logistic loss. They don't have a tight lower bound even there's no misclassification.

[redacted]

Name: ______________________________

Bayes Classifier

18. (7 points) Suppose for a given data-set with features X and labels Y, we know the true underlying distribution is: $P(X|Y=0) = \mathcal{N}(\mu_0, \sigma^2)$, $P(X|Y=1) = \mathcal{N}(\mu_1, \sigma^2)$ and $P(Y=1) = P(Y=0)$, what is the Bayes classifier for this data-set?

Solution:

$$\begin{aligned} P(Y=1|X) &= \frac{P(X|Y=1)P(Y=1)}{P(X|Y=0)P(Y=0) + P(X|Y=1)P(Y=1)} \\ &= \frac{\mathcal{N}(\mu_1, \sigma^2)}{\mathcal{N}(\mu_0, \sigma^2) + \mathcal{N}(\mu_1, \sigma^2)} \\ &= \frac{1}{1 + \exp\left(\frac{2(\mu_0 - \mu_1)X - (\mu_0^2 - \mu_1^2)}{2\sigma^2}\right)} \end{aligned}$$

Therefore, the Bayes classifier is

$$h_B(X) = \begin{cases} 1\text{ , if } \frac{1}{1+\exp\left(\frac{2(\mu_0-\mu_1)X-(\mu_0^2-\mu_1^2)}{2\sigma^2}\right)} > 0.5 \\ 0\text{ , otherwise} \end{cases}$$

The condition for $h_B(X) = 1$ can be simplified as

$$\begin{aligned} x &> \frac{\mu_0 + \mu_1}{2}\text{ ,if } \mu_0 < \mu 1 \\ x &< \frac{\mu_0 + \mu_1}{2}\text{ ,if } \mu_0 > \mu 1 \end{aligned}$$

i.e. x is closer to μ_1 than μ_0

[redacted]

Name: ______________________________

Convolutional Neural Networks

19. (15 points) Given a sequence of 6 bits (0 or 1), Bob is trying to create a 1D convolutional network to recognize if the string "101" occurs in this sequence. Bob thinks this task is easy and so he just sets weights for his network instead of learning them. He chooses to use a 1D convolution with weights $w_1 = 1, w_2 = 0, w_3 = 1$, a bias term of $b = -1.5$, a simple step function activation, stride 1, and no padding. For clarity consider the figure below:

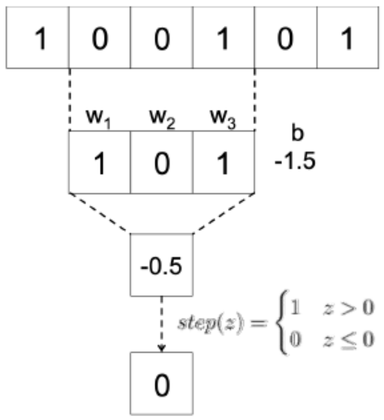


Figure 2: An example of the convolutional layer applied to part of the input

Note that $\text{step}(0) = 0$.

(a) (2 points) What will be the output size of this first convolutional layer?

Solution: Since the stride is 1, we just have $6 - 3 + 1 = 4$ as the output size.

[redacted]

To reduce the output size of the convolutional layer to a 0-1 classification output, Bob decides to use a global average pooling layer (which simply averages all of the outputs from the previous layer), with another step function based activation after the layer.

(b) (5 points) Find a possible input sequence that would make the network output 1 but does not have "101" as a substring.

> **Solution:** Any sequence of 6 bits that has "111" as a subsequence **and** does not have "101" as a subsequence is acceptable. We note that passing in this sequence leads to a preactivation of $0.5 > 0$, which the step function turns to 1. Since all outputs before the global average pooling layer are nonnegative, any positive output from the convolutional layer will propagate through, so this subsequence will be misclassified as containing a "101".

(c) (5 points) Modify one of the weights (or the bias) of the convolution filter so that it works as desired (and no longer fails on the input sequence found in part b).

> **Solution:** If we consider all other sequences of 3 bits, we see that the network has the correct classification for everything other than "111". To fix it's classification for this point, we must penalize against a '1' in the central position, so we should make w_2 negative. Specifically, any modification that changes to w_2 to a value $w_2 \leqslant -0.5$ is acceptable. For part d, we use $w_2 = -1$ as an example.

(d) (3 points) Bob decides that he wants his network to be able to take into account sequences with variable finite lengths (e.g. a 20-bit sequence). Does Bob need to make any changes to his network architecture (which ran on 6-bit inputs)? Write a one sentence explanation as to what the change would be (if needed) or why a change is not needed.

> **Solution:** No change is needed since the global average pooling layer works with any number of inputs. As long as one of the inputs to the average pooling layer is nonzero, the output will be 1 (since all the inputs to the layer are nonnegative).

Regression

20. (20 points) Consider the following datasets consisting of 100 samples indexed by $k = 1, .., 100$.

- Feature vector $x^{(k)} = [1, k-2, (k-1)^2]^T$,
- Label $y^{(k)} = (k+3)^2$

You are to perform a ridgeless (unregularized) linear regression, that is find the best fit

$$\hat{y} = \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3 .$$

(Note that we are not adding an intercept $\hat{\beta}_0$ because its role is played by $\hat{\beta}_1$.)

(a) (10 points) Compute the regression coefficients (ERM solution).

Solution: $\hat{\beta}_1 = 24, \hat{\beta}_2 = 8, \hat{\beta}_3 = 1$. The empirical risk/loss is:

$$\begin{aligned}
&\frac{1}{100}\sum_{k=1}^{100}\left(y^{(k)} - \hat{y}^{(k)}\right)^2 \\
=&\frac{1}{100}\sum_{k=1}^{100}\left[(k+3)^2 - (\hat{\beta}_1 + \hat{\beta}_2(k-2) + \hat{\beta}_3(k-1)^2)\right]^2 \\
=&\frac{1}{100}\sum_{k=1}^{100}\left[(k^2 + 6k + 9 - \hat{\beta}_1 - \hat{\beta}_2 k + 2\hat{\beta}_2 - \hat{\beta}_3 k^2 + 2\hat{\beta}_3 k - \hat{\beta}_3^2\right]^2 \\
=&\frac{1}{100}\sum_{k=1}^{100}\left[(1-\hat{\beta}_3)\,k^2 + (6 - \hat{\beta}_2 + 2\hat{\beta}_3)\,k + (9 - \hat{\beta}_1 + 2\hat{\beta}_2 - \hat{\beta}_3{}^2)\right]^2
\end{aligned} \tag{1}$$

The coefficients of k^2, k, and $1 = (k^0)$ are all linear in the three free parameters we get to adjust. If we can make these coefficients zero, we'd incur zero empirical risk, which is the best we can do when using a squared loss. Indeed, we arrive at $\hat{\beta}_1 = 24, \hat{\beta}_2 = 8, \hat{\beta}_3 = 1$ by solving the three linear equations,

$$\begin{cases} 1 - \hat{\beta}_3 = 0 \\ 6 - \hat{\beta}_2 + 2\hat{\beta}_3 = 0 \\ 9 - \hat{\beta}_1 + 2\hat{\beta}_2 - \hat{\beta}_3^2 = 0 \end{cases}$$

(b) (5 points) Show your prediction on a new input $x^{(new)} = [0, 0, 0]$.

Solution: $24 * 0 + 8 * 0 + 1 * 0 = 0$; hence we predict 0.

(c) (5 points) Suppose that you observe $y^{(new)} = 0$. Compute updated regression coefficients (new ERM solution).

Solution: The empirical risk is already zero with the old coefficients and cannot go lower. Hence we need not update the coefficients.

[redacted]

Bayes Classifier

21. (10 points) Our friend calculated posteriors $Q(y|x)$ for a slightly different dataset than ours where the prior class probabilities $Q(y)$ were different. In other words, our friend's data was generated by $Q(x, y) = P(x|y)Q(y)$ while ours came from $P(x, y) = P(x|y)P(y)$ (note that $P(x|y)$ is the same for both). Given our friend's $Q(y|x)$, their incorrect class priors $Q(y)$, and the true class priors $P(y)$ for our data, how do we get the Bayes classifier for our data?

Solution:

$$\begin{aligned} h^*(x) &= \arg\max_y P(y|x) \\ &= \arg\max_y P(x|y)P(y) \\ &= \arg\max_y P(x|y)Q(y)\frac{P(y)}{Q(y)} \\ &= \arg\max_y \frac{P(x|y)Q(y)}{Q(x)}\frac{P(y)}{Q(y)} \\ &= \arg\max_y Q(y|x)\frac{P(y)}{Q(y)} \end{aligned}$$

[redacted]

Name: ______________________

Logistic regression

22. (11 points) We are interested in regularizing the terms separately in logistic regression.

(a) (2 points) Consider the data in the figure below where we fit the model

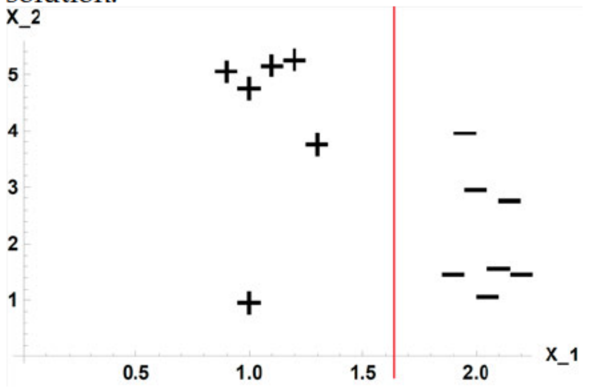
$$P(y = 1 \mid x, w) = \text{Sigmoid}(w_0 + w_1 x_1 + w_2 x_2)$$

Suppose we fit the model by maximum likelihood, that is, we minimize

$$J(w) = \sum_{i=1}^{n} -\log P(y^i | x^i, w)$$

Sketch a possible decision boundary corresponding to w^*, marking the boundary as "(a)"

Solution: The solution is not unique (belows are the same). Here is one possible solution.



(b) (2 points) Is your decision boundary unique?

Solution: No. Weights will want to go to infinity to maximize likelihood but there's room for rotation and translation.

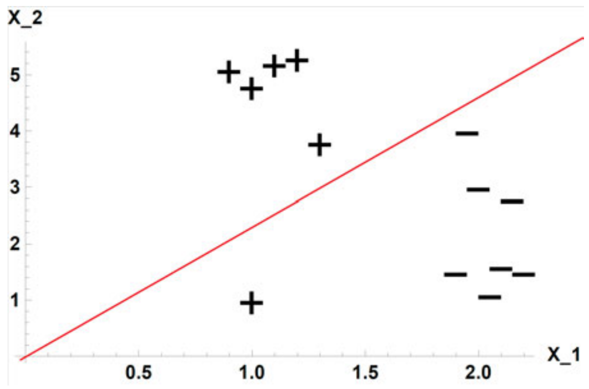
(c) (5 points) Now suppose we regularize only the w_0 parameter; that is, we minimize

$$J(w) = [\sum_{i=1}^{n} -\log P(y^i | x^i, w)] + \lambda w_0^2$$

with λ approaching ∞. Sketch a possible decision boundary corresponding to (the regularized) w^*, marking this boundary as "(c)"

[redacted]

Solution:



(d) (2 points) How many classification errors does it make on the training set?

Solution: 1