STANFORD UNIVERSITY
CS 229, Autumn 2019
Midterm Examination

Tuesday, November 5, 6:00pm-9:00pm

Question	Points
1 Short answers	/19
2 Multi-class GDA	/14
3 K-means	/11
4 Universal approximation	/21
5 L_2 regularization	/22
6 Representer theorem	/22
Total	/109

Name of Student: _____

SUNetID: _____@stanford.edu

The Stanford University Honor Code:

I attest that I have not given or received aid in this examination, and that I have done my share and taken an active part in seeing to it that others as well as myself uphold the spirit and letter of the Honor Code.

Signed: _____

Instructions

Logistics:

1. You are allowed 3 double-sided 8.5x11 inch pages of notes (handwritten or typed).
2. This exam has 26 pages in total. Please check that this is the case. If not, please let the teaching staff know immediately.
3. The use of electronic devices (except pre-approved medical devices) is **not** allowed during the entire duration of this exam.

***Additional rules for proofs in Problems 5 and 6*:**

1. **For questions asking for more than a brief explanation, we will check whether your reasoning is correct, whether you prove the desired result, and whether all your intermediary steps are valid.**
2. **As such, you must prove that multiplicative constants are non-negative when you move them through inequalities, that an argument is the global minimizer of a function if said function is convex, that inverses exist when you take them, etc.**

Please address any clarification you may need (technical or otherwise) to the teaching staff.

Formulas

- *Multivariate Gaussian probabiltity density function:*

$$\mathcal{N}(\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

- *Second-order multivariate Taylor expansion of f about x_0, for all $x \in domf$:*

$$f(x) = f(x_0) + (x-x_0)^\top \nabla_x f(x)\Big|_{x_0} + \frac{1}{2}(x-x_0)^\top \nabla_x^2 f(x)\Big|_{x_0}(x-x_0) + o\left(\|x-x_0\|_2^2\right)$$

- *Indicator function over an arbitrary set $\mathcal{S}$:*

$$\mathbb{1}_{\mathcal{S}} : x \mapsto \begin{cases} 1, & \text{if } x \in \mathcal{S} \\ 0, & \text{otherwise} \end{cases}$$

- *Probability mass function (pmf):*

$$p_X : x \mapsto \mathbb{P}(X = x)$$

  for a discrete random variable X, e.g.

$$p_X(x) = p^x(1-p)^{1-x}$$

  if $X \sim \text{Bernoulli}(p)$.

- *Probability density function (pdf):* f_X such that

$$\mathbb{P}[a \leq X \leq b] = \int_a^b f_X(x)dx$$

  for a continuous random variable X, e.g.

$$f_X(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

  if $X \sim \mathcal{N}(0,1)$.

Notations

- $\mathbb{R}_+$: the set of non-negative real numbers.
- $\mathbb{S}_+^d$: the set of $d \times d$ PSD matrices.
- $[\![1, n]\!]$: $\{1, 2, \ldots, n\}$

1. **[19 points] Short answers**

   (a) **[4 points] Loss functions for different tasks**
   Consider the following loss functions:

   A. $-(y \log \hat{y} + (1-y)\log(1-\hat{y}))$
   B. $\frac{1}{2}(\hat{y}-y)^2$
   C. for $k > 2$, $-\sum_{j=1}^{k} \mathbf{1}\{y = j\} \log \hat{y}_j$
   D. $\max(0, 1 - y\hat{y})$

   For each of the given tasks below, *circle one or more* letters, corresponding to the list above, that would be most appropriate to use in a learning algorithm to solve the task.
   If you deem none of the above options adequate, please circle **None**. For each choice, you gain 1 point if and only if your answer is *entirely* correct.

   i. Estimate the number of hours that a problem set will take.

   A **B** **C** **D** **None**
   Answer: **B**.
   Because we are predicting a continuous quantity (as opposed to a class), we formulate it as a regression problem, for which only squared loss is applicable here.

   ii. Group similar students together based on what classes they've taken.

   A **B** **C** **D** **None**
   Answer: **None**.
   A lot of students put **A** as a loss function, but **A** is the loss function for logistic regression, a binary classification problem. In this situation, we don't know how what the groupings are, or how many of them there are. Therefore, we need to apply a method like k-means, whose loss function is not listed.

   iii. Predict whether or not the average house price in SF will increase this year.

   A **B** **C** **D** **None**
   Answer: We accepted both {**A, D**} and {**A, B, D**} as correct answers.
   A is the loss for logistic regression, a binary classification problem. **D** is the hinge loss, and is used in SVM classification. We accepted **B** because you could create a regression on the house price, and then just make a binary decision on that, but the intent of the problem was to frame the situation as binary classification.

   iv. Predict whether tomorrow will be a rainy, cloudy or sunny day.

A **B** **C** **D** **None**

Answer: **C**.

This is a classification problem with more than two classes. Therefore, **A** and **D** are not sufficient, even though they are used for classification.

(b) **[10 points] SVM and kernels**
We generate 100 examples using three features that are sampled independently from a standard Gaussian distribution. For the i^{th} example, denote those features $x_1^{(i)}$, $x_2^{(i)}$ and $x_3^{(i)}$. The labels are binary, i.e. the i^{th} label, $y^{(i)}$, is either 0 or 1. Specifically, for 20 training examples $i \in [\![1, 20]\!]$:

$$y^{(i)} = \begin{cases} 1, & \text{if } \sum_{j=1}^{3} {x_j^{(i)}}^2 \geq c \\ 0, & \text{otherwise} \end{cases}$$

i. **[2 points]** Qualitatively assess the performance in terms of *separability* of using SVM in the original feature space. Briefly explain your answers.
Answer: Vanilla SVM gives a linear decision boundary and will perform poorly. The data was created using quadratic features so a hyperplane is not a good decision boundary.
(+1 for the evaluation of performance, +1 for explanation)

ii. **[2 points]** Qualitatively assess the performance in terms of *generalizability* of using SVM with a 15-degree polynomial kernel. Briefly explain your answers.
Answer: SVM with a 15-degree polynomial kernel will not give a good decision boundary either, since it will likely overfit the training data.
(+1 for the evaluation of performance, +1 for explanation)

iii. [**4 points**] Which kernel(s) would you choose to parameterize SVM to separate above two classes to obtain both *separability* and *generalizability*? Select all that apply and briefly explain your answers.

A 2-degree polynomial kernel
B 20-degree polynomial kernel
C 200-degree polynomial kernel
D dot product kernnel

Answer: A.
SVM with a 2-degree polynomial kernel will map features to quadratic feature spaces, which is what we used to generate the dataset. poly3 and poly6 kernels will overfit the training data. SVM with dot product kernel gives linear decision boundary.
(+1 for each correct explanation.)

iv. [**2 points**] Suppose we augment the features by adding five additional noisy features sampled from a standard Gaussian distribution. How would augmenting feature space as such affect the test error of your kernelized SVM? Briefly explain your answers.
Answer: Including 5 more noise features will increase the test error of kernerlized SVM.
Kernels would not easily find the subspace where class separation occurred, i.e. it would not weight the informative subspace higher than the noisy subspace. It would suffer from having many dimensions to search over.
(+1 for answering the increase of test error, +1 for correct explanation.)

(c) **[5 points] Non-exponential family distribution**
Although we have seen many examples of exponential family distributions, there exist distributions which cannot be written in the exponential family form $p(x;\eta) = b(x)\exp\left(\eta^T T(x) - a(\eta)\right)$.
Consider the following options for distributions:

(1) Uniform distribution of parameters a, b, with pdf $f(x) = \frac{1}{b-a}\mathbb{1}_{[a,b]}(x)$.

(2) Exponential distribution of parameter λ, with pdf $f(x) = \lambda e^{-\lambda x}\mathbb{1}_{\mathbb{R}_+}(x)$.

(3) Binomial distribution of parameters n, p, with pmf

$$p(x) = \binom{n}{x} p^x (1-p)^{n-x}\mathbb{1}_{[\![1,n]\!]}(x)$$

where $\binom{n}{x} = \frac{n!}{x!(n-x)!}$.

(4) Binomial distribution of parameter p and known n, with pmf being the same form as the one in (3).

(5) Multivariate Gaussian of parameters μ, Σ, with pdf

$$f(x) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

Circle one or more of the above options corresponding to distributions that do *not* belong to the exponential family, and briefly explain why *not*. No explanation is needed for those that do belong to the exponential family. For each choice, you gain 1 point if and only if your answer is *entirely* correct, i.e. circled with explanation if the distribution is *not* in the exponential family and not circled with no explanation otherwise.

Answer: Notice that the support of an exponential family distribution is completely determined by $b(x)$ since $\exp\left(\eta^T T(x) - a(\eta)\right)$ will always be non-zero for any finite $T(x)$ and η. However, by formulation, $b(x)$ cannot be a function of its parameters. Therefore, **(1) and (3) do not belong to the exponential family** because its support depend on parameters.
The support of (1) depends on both a and b; the support of (3) depends on n. (4) belongs to the exponential family because n is a known constant for it.

2. **[14 points] Multiclass GDA**

In this problem we will explore a multi-class (k-class) extension of Gaussian Discriminant Analysis (GDA). Let us consider the model which has the following data generative process:

$$\begin{aligned} y &\sim \text{Categorical}(\phi) \\ x \mid y = 1 &\sim \mathcal{N}(\mu_{[1]}, \Sigma) \\ &\vdots \\ x \mid y = k &\sim \mathcal{N}(\mu_{[k]}, \Sigma), \end{aligned}$$

where $y \in \{1, \ldots, k\}$ denotes the class label, $\phi \in \mathbb{R}^k$ are the parameters of the class prior categorical distribution such that $0 < \phi_j < 1$ for all j and $\sum_{j=1}^k \phi_j = 1$, $x, \mu_{[1]}, \ldots, \mu_{[k]} \in \mathbb{R}^d$ denote the input data and class means, and $\Sigma \in \mathbb{S}_+^d$ is the shared covariance matrix across all the k classes.

Let us suppose that the parameters ϕ, $\mu_{[1]}, \ldots, \mu_{[k]}$ and Σ have been learned from the training data, and now we want to predict y for a new input x. Derive the form of the posterior distribution $p(y|x)$ for the above model. In particular, show that the posterior distribution takes the form of the softmax function:

$$p(y = j \mid x) = \frac{\exp\left\{b_j + \theta_{[j]}^T x\right\}}{\sum_{i=1}^k \exp\left\{b_i + \theta_{[i]}^T x\right\}}$$

where $\theta_{[i]} \in \mathbb{R}^d$ and $b_i \in \mathbb{R}$ for all $i = 1, \ldots, k$.

(a) **[6 points]** Provide expressions for b_i and $\theta_{[i]}$ (for any i, since they are all symmetric) in terms of $\phi, \mu_{[1]}, \ldots, \mu_{[k]}$ and Σ.

Answer:

$$\begin{aligned} p(y = j|x) &= \frac{p(x|y = j)p(y = j)}{\sum_i p(x|y = i)p(y = i)} \quad \textit{(+2 pt)} \\ &= \frac{\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(x - \mu_{[j]})^T\Sigma^{-1}(x - \mu_{[j]})\}\phi_j}{\sum_i \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(x - \mu_{[i]})^T\Sigma^{-1}(x - \mu_{[i]})\}\phi_i} \quad \textit{(+2 pt)} \\ &= \frac{\exp\left\{-\frac{1}{2}(x - \mu_{[j]})^T\Sigma^{-1}(x - \mu_{[j]})\right\}\phi_j}{\sum_i \exp\left\{-\frac{1}{2}(x - \mu_{[i]})^T\Sigma^{-1}(x - \mu_{[i]})\right\}\phi_i} \quad \textit{(+1 pt)} \\ &= \frac{\exp\left\{-\frac{1}{2}x^T\Sigma^{-1}x - \frac{1}{2}\mu_{[j]}^T\Sigma^{-1}\mu_{[j]} + x^T\Sigma^{-1}\mu_{[j]}\right\}\phi_j}{\sum_i \exp\left\{-\frac{1}{2}x^T\Sigma^{-1}x - \frac{1}{2}\mu_{[j]}^T\Sigma^{-1}\mu_{[j]} + x^T\Sigma^{-1}\mu_{[j]}\right\}\phi_i} \\ &= \frac{\exp\left\{-\frac{1}{2}\mu_{[j]}^T\Sigma^{-1}\mu_{[j]} + \mu_{[j]}^T\Sigma^{-1}x\right\}\phi_j}{\sum_i \exp\left\{-\frac{1}{2}\mu_{[j]}^T\Sigma^{-1}\mu_{[j]} + \mu_{[j]}^T\Sigma^{-1}x\right\}\phi_i} \\ &= \frac{\exp\left\{\log\phi_j - \frac{1}{2}\mu_{[j]}^T\Sigma^{-1}\mu_{[j]} + \mu_{[j]}^T\Sigma^{-1}x\right\}}{\sum_i \exp\left\{\log\phi_i - \frac{1}{2}\mu_{[j]}^T\Sigma^{-1}\mu_{[j]} + \mu_{[j]}^T\Sigma^{-1}x\right\}} \end{aligned}$$

$$\Rightarrow \boxed{b_i = \log \phi_i - \frac{1}{2}\mu_{[i]}^T \Sigma^{-1} \mu_{[i]}, \quad \theta_{[i]} = \Sigma^{-1}\mu_{[i]}}$$

(+1 for each)

(b) **[3 points]** Suppose an input data point $x \in \mathbb{R}^d$ lies on the decision boundary between class j_1 and class j_2, where $(j_1, j_2) \in [\![1, k]\!]^2$ and $j_1 \neq j_2$. Write the condition x satisfies in terms of $\theta_{[j_1]}$, b_{j_1}, $\theta_{[j_2]}$, b_{j_2} and $\theta_{[j]}$, b_j for all $j \neq j_1, j_2$. As such, is the decision boundary between two classes affine?

Answer: A point x lying on the decision boundary between classes j_1 and j_2 satisfies:

$$\forall\ j \neq j_1, j_2, \quad p(y = j_1|x) = p(y = j_2|x) > p(y = j|x)$$

which can be re-expressed as:

$$\forall\ j \neq j_1, j_2, \quad b_{j_1} + \theta_{[j_1]}^\top x = b_{j_2} + \theta_{[j_2]}^\top x > b_j + \theta_{[j]}^\top x$$

(+2 for correct equality (+ inequality), +1 for attempt)
This is linear/affine *(+1)*.

(c) **[5 points]** Now suppose the covariance matrix for the prior of the first class becomes $c\Sigma$ with $c > 1$, and further suppose $\phi_j > 0$ for all $j \in [\![1, k]\!]$. Write the new decision boundary between class 1 and class 2 in terms of Σ, c,$\mu_{[1]}$,$\mu_{[2]}$, ϕ_1 and ϕ_2. Compared to when the covariance matrix was the same Σ across all classes, does the current posterior $p(y = 1|x)$ increase, decrease or stay unchanged for x such that $\|x-\mu_{[1]}\|_2 \approx 0$? What about for x such that $\|x-\mu_{[1]}\|_2$ is very large?

Answer: A point x lying on the decision boundary between classes 1 and 2 satisfies:

$$\frac{1}{(2\pi c)^{d/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2c}(x-\mu_{[1]})^T\Sigma^{-1}(x-\mu_{[1]})\right\}\phi_1$$
$$=\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_{[2]})^T\Sigma^{-1}(x-\mu_{[2]})\right\}\phi_2$$

(+3 for expression)

- $\|x-\mu_{[1]}\|_2 \approx 0$:
  When c is larger, the probability distribution is more spread-out. Given class $j = 1$, the probability that x is close to the mean (i.e. $\|x-\mu_{[1]}\|_2 \approx 0$) thus decreases. Correspondingly, this decreases the probability of x being in class 1 for x close to the mean.
- $\|x-\mu_{[1]}\|_2$ is very large:
  Again, when c is larger, the probability distribution is more spread-out. Thus, given class $j = 1$, the probability that x is away from the mean (i.e. $\|x-\mu_{[1]}\|_2$ is very large) increases. Correspondingly, this increases the probability of x being in class 1 for x away from the mean.

(+1 for each case)

3. **[11 points] K-means**

Recall the k-means clustering algorithm on a set of data points $X = \{x^{(1)}, \ldots, x^{(n)}\}$:

- Initialize the cluster centroids $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^d$ to be k randomly chosen points from X.
- Repeat until convergence: {

  For every i, set
  $$c^{(i)} := \arg\min_j \|x^{(i)} - \mu_j\|^2.$$

  For each j, set
  $$\mu_j := \frac{\sum_{i=1}^n \mathbf{1}\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^n \mathbf{1}\{c^{(i)} = j\}}.$$

  }

Also, recall in the notes that we defined a distortion function as follows:

$$J(c, \mu) = \sum_{i=1}^n \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

(a) **[3 points]** k-means is not guaranteed to converge to the global minimum. Provide an example of non-global convergence for $k = 2$. Specify (1) a set $X = \{x^{(i)} \in \mathbb{R}\}$ (i.e. for $d = 1$), (2) a set of converged cluster centroids and their cost J_1, and (3) a different set of converged cluster centroids and their cost $J_2 \neq J_1$.

Answer: Example solution:

- $X = [0, 4, 10]$
- centroid set 1 [0, 7], $J_1 = 18$
- centroid set 2 [2, 10], $J_2 = 8$

Points

- *+3 valid centroids and attempt at costs*
- *+2 minor mistake eg. no cost*
- *+1 some attempt*
- *0 no answer or completely incorrect*

(b) [**4 points**] Give a brief explanation why k-means will always converge to a locally optimal clustering.

Answer:

- We argue that J decreases monotonically after each iteration of the inner loop.
- The c_i assignment step minimizes J for every point.
- The μ_j assignment step minimizes J for every centroid, since the mean of a set of points minimizes the sum of distance squared to them (proof below).
  - let $f(y) = \sum_{i=1}^{n} \|x_i - y\|^2$
  - $\nabla_y f(y) = \sum_{i=1}^{n} 2(x_i - y) = 0$
  - $ny = \sum_{i=1}^{n} x_i$
  - $y = \frac{1}{n} \sum_{i=1}^{n} x_i$
- Because the minimum of J is 0 and every step of the algorithm decreases J, it must eventually converge.

Points

- *+4 fully correct solution, such as above*
- *+3 makes a minor mistake (such as not showing that J is bounded from below, or there are finite number of possible clusters)*
- *+2 correct idea but missing substantial detail, or the solution only gives an intuitive description of how k-mean converges to a locally optimal clustering, but lack any rigorous proof to support that.*
- *+2 If the solution only explains why the result will be locally optimal but not globally optimal, it will only get 2 points, because it is not really answering the question of why k-mean will converge and why the clusters are locally optimal after convergence.*
- *+1 makes a major mistake*
- *0 unattempted*

(c) [**4 points**] Because k-means converges to local optima, a standard technique is to run the algorithm many times and pick the clusters of the run that yielded the lowest J. It turns out that with better initialization, we can greatly improve the performance of k-means, having it run in fewer iterations and converge to lower J. The k-means++ algorithm presents an initialization technique that generally performs much better than random initialization:

i. Choose an initial centroid $\mu_1 \in \mathbb{R}^d$ uniformly at random from X.

ii. With $D(x)$ defined as the shortest distance from a data point x to its closest centroid that has already been chosen, choose the next centroid μ_i by randomly sampling $x' \in X$ with probability

$$\frac{D(x')^2}{\sum_{x \in X} D(x)^2}$$

iii. Repeat step (ii) until all k centroids have been chosen.

iv. Proceed with regular k-means using these centroids.

In 2-3 sentences, provide the intuition for why k-means++ will generally perform better than regular k-means.

Answer:

- The probability distribution will cause points far away from existing clusters to more likely be chosen, meaning k-means++ will tend to produce a distribution of clusters that is spread out over the dataset.
- Because the centroids are more spread out, they are less likely to be redundantly close to each other (since two centroids that are close together will tend to drift apart).
- Because k-means++ will generally produce a more 'natural' initial clustering with much lower J than regular k-means, it will also likely converge in fewer iterations and yield lower final J.
- Beyond the scope of what's expected for full credit, but for an in-depth example, consider the following: if there are N true clusters of equal size that are well-separated, regular k-means will place initial one centroid in each true cluster with probability $\approx \frac{N!}{N^N}$ which quickly approaches 0, while k-means++ will more likely provide better coverage.

Points

- *+4 Explains how initial clusters will be spread out in and motivates* **why** *this leads to faster convergence / lower cost J than regular k-means.*
- *+3 Explains that the intial clustroids will be more spread out, but doesn't give deeper justification for why that is better than regular k-means, or justification is partially incorrect*
- *+2 Correctly understands (ii) assigns higher probability to further away points, but doesn't indicate what this implies (initial cluster distribution is more spread out than random) or makes a non-trivial incorrect claim*

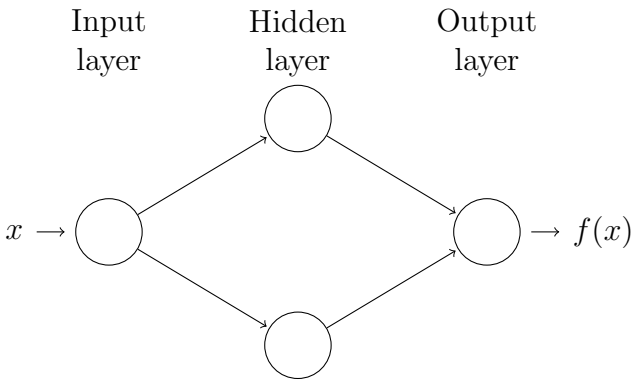
- *+1 Unclear, makes largely incorrect claim, and/or missing key point about initial clustroids being more evenly distributed.*
- *0 Unattempted or barely attempted*

4. **[21 points] Universal approximation for neural networks**

   In this problem, we develop an intuition for the flexibility of neural networks.

   (a) **[7 points]** Define a neural network with one hidden layer consisting of two hidden neurons to approximate the unit pulse function

$$f(x) = \begin{cases} 1 & -\frac{1}{2} \leq x < \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}, x \in \mathbb{R}$$



   Using:

   - the sign function $g(z) = \mathbb{1}\{z \geq 0\}$ as your activation function in the hidden layer,
   - and the identity function as the activation function in the output layer,

   fully specify the dimensions and explicitly write the values of $W^{[i]}$ and $b^{[i]}$ for $i \in \{1, 2\}$, where $W^{[i]}$ and $b^{[i]}$ are the i^{th} layer's weight matrix and bias vector, respectively, such that the output of this neural network is exactly f.

   Answer: The neural network is as thus:

$$Z^{[1]} = W^{[1]}x + b^{[1]}$$

$$A^{[1]} = g(Z^{[1]}) = \begin{bmatrix} \mathbb{1}\{z_1^{[1]} \geq 0\} \\ \mathbb{1}\{z_2^{[1]} \geq 0\} \end{bmatrix}$$

$$f(x) = a^{[2]} = Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$

   such that

$$W^{[1]} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$b^{[1]} = \begin{bmatrix} \frac{1}{2} \\ -\frac{1}{2} \end{bmatrix}$$

$$W^{[2]} = \begin{bmatrix} 1 & -1 \end{bmatrix}$$

$$b^{[2]} = 0$$

   To arrive at the above, note that:

$$f(x) = Z^{[2]} = w_1^{[2]}\mathbb{1}\{w_1^{[1]}x + b_1^{[1]} \geq 0\} + w_2^{[2]}\mathbb{1}\{w_2^{[1]}x + b_2^{[1]} \geq 0\} + b^{[2]}$$

(b) **[2 points]** Let ϕ_{ϵ,h,x_0} be the pulse function centered around x_0, of width 2ϵ and height h. Write the expression of ϕ_{ϵ,h,x_0} as a piecewise function.

Answer:

$$\phi_{\epsilon,h,x_0}(x) = \begin{cases} h & (x_0 - \epsilon) \leq x < (x_0 + \epsilon) \\ 0 & o.w. \end{cases}, x \in \mathbb{R}$$

(+0.5 for correct bounds, +0.5 for correct inequalities, +1 for correct height)

(c) **[7 points]** Explicitly write the values of the weight matrices and bias vectors of the network shown in (a) such that the output is now **not** f, but ϕ_{ϵ,h,x_0}. Use the same activation functions as in (a). Express these values in terms of ϵ, h, and x_0.

Answer: The neural network is as thus:

$$\begin{aligned} Z^{[1]} &= W^{[1]}x + b^{[1]} \\ A^{[1]} &= g(Z^{[1]}) = \begin{bmatrix} \mathbb{1}\{z_1^{[1]} \geq 0\} \\ \mathbb{1}\{z_2^{[1]} \geq 0\} \end{bmatrix} \\ \phi(x) = a^{[2]} &= Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]} \end{aligned}$$

such that

$$\begin{aligned} W^{[1]} &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ b^{[1]} &= \begin{bmatrix} \epsilon - x_0 \\ -\epsilon - x_0 \end{bmatrix} \\ W^{[2]} &= \begin{bmatrix} h & -h \end{bmatrix} \\ b^{[2]} &= 0 \end{aligned}$$

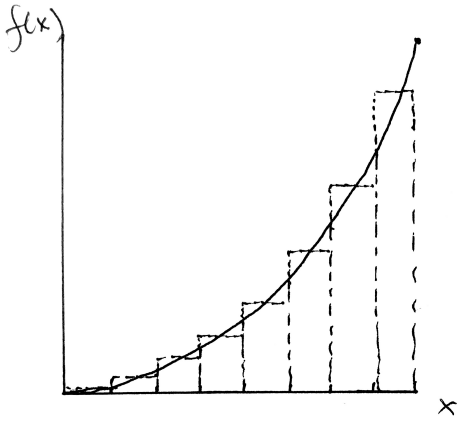
To arrive at the above, note that:

$$\phi(x) = Z^{[2]} = w_1^{[2]}\mathbb{1}\{w_1^{[1]}x + b_1^{[1]} \geq 0\} + w_2^{[2]}\mathbb{1}\{w_2^{[1]}x + b_2^{[1]} \geq 0\} + b^{[2]}$$

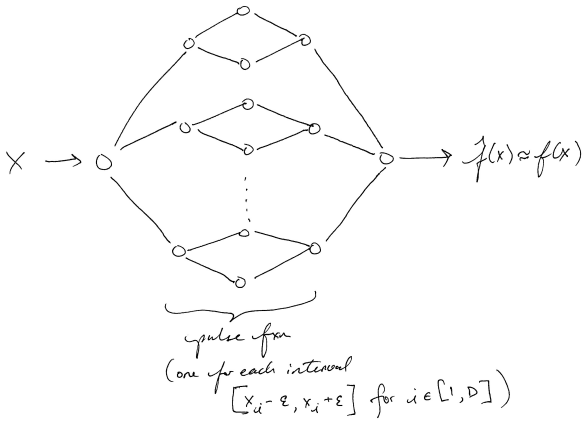
$$= h\mathbb{1}\{x \geq x_0 - \epsilon\} - h\mathbb{1}\{x \geq x_0 + \epsilon\}$$

(d) **[5 points]** Describe how the output of the generalized pulse function in part (c) can be used to approximate some arbitrary function $f : \mathbb{R} \to \mathbb{R}$ over a finite interval $[x_{\min}, x_{\max}]$ using a neural network. Draw a graph comparing $f(x)$ with the output of the neural network. Draw the structure of the neural network. No formal proof is required: a straightforward description in words, equations, and diagrams would suffice.

Answer: The key insight here is to use piecewise pulse functions to numerically approximate the target function, as shown in the following example:

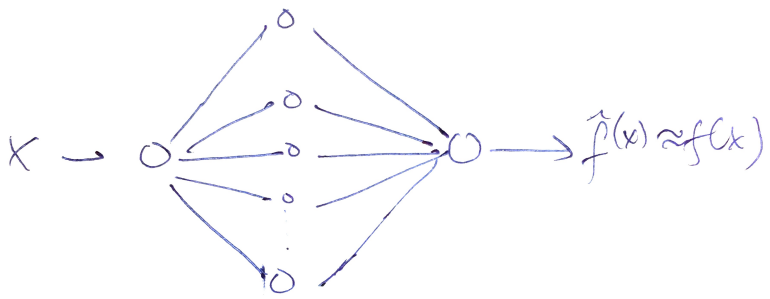


Within a neural network, this can be done by defining pulse functions over consecutive intervals $[x_i - \epsilon, x_i + \epsilon]$, $i \in [\![1, D]\!]$ where D is the number of intervals in $[x_{min}, x_{max}]$, then learning the weights for each of these functions corresponding to the value of $f(x_i)$ within the i-th interval (i.e. the height of the pulse function $h_i = f(x_i)$). These pulse functions can be arranged in parallel, and the resultant network outputs the sum of the pulse function outputs. Then, feeding some given value of x would only activate the pulse function for the i-th interval such that $x \in [x_i - \epsilon, x_i + \epsilon]$, which then returns the learned weight h_i corresponding to the value of $f(x_i)$, which closely approximates $f(x)$.



Additional note: Another way of constructing this NN is as such, with hidden neurons indicating whether the given x were greater than some threshold (the interval between thresholds defined by ϵ). Then, the 0-th neuron would have weight $f(x_0^0)$,

and the i-th neuron would have weight $f(x_0^i) - f(x_0^{i-1})$. This way, for a given x input, all the neurons with threshold less than x will activate, and the resulting sum of the hidden outputs telescopes to $f(x_0^i)$ for i being the largest index of a neuron that fires.



(+1 for interval explanation, +1 for using height of pulse function to approximate $f(x)$, +1 for drawing correct graph comparing $f(x)$ to consecutive pulse function output, +1 for summing pulse function outputs in the output layer of NN, +1 for drawing correct NN structure)

5. **[23 points] L_2 regularization under quadratic approximation**

   Consider a twice-differentiable and convex objective function, $J : \mathbb{R}^d \to \mathbb{R}$. Since J is twice-differentiable, you may assume the following result:

$$\forall\ \theta \in \mathbb{R}^d,\ \forall\ (i,j) \in [\![1,d]\!]^2, \quad \frac{\partial^2 J(\theta)}{\partial\theta_i \partial\theta_j} = \frac{\partial^2 J(\theta)}{\partial\theta_j \partial\theta_i}$$

   As you have seen in class, a common technique to prevent overfitting is to add a regularization term to this objective that "shrinks" the weights. Suppose that we are applying L_2 regularization.

   (a) **[1 points]** Let $\theta \in \mathbb{R}^d$. Write the regularized objective, $\tilde{J}$, evaluated at θ, as a function of J and θ, denoting the regularization parameter by $c \geq 0$, and assuming a multiplicative factor of $\frac{1}{2}$ (for convenience).

   Answer:

$$\tilde{J}(\theta) = J(\theta) + c\frac{1}{2}\theta^\top\theta$$

   (b) **[2 points]** Derive the update rule if we were to use gradient descent for the *regularized* objective $\tilde{J}$ with a learning rate $0 < \alpha < 1/c$. Derive the multiplicative term in front of θ as function of α and c. Does θ decrease in this update rule?

   Answer:

$$\begin{aligned}\theta &\leftarrow \theta - \alpha\left(\nabla_\theta J(\theta) + c\theta\right)\\ &\leftarrow \theta(1-\alpha c) - \alpha\nabla_\theta J(\theta)\end{aligned}$$

   (+1.5 for correct multiplicative factor)

   Since the update rule contains a gradient term, there is no guarantee just from this update rule on whether the magnitude of θ shrinks or not *(+0.5)*.

(c) Note that the above is for one single step. Let's try to characterize what happens over the entire course of training. Recall that the goal in the *unregularized* case is to find the weights that minimize the objective J, i.e. find θ^* such that:

$$\theta^* = \arg\min_\theta J(\theta)$$

which is a different minimizer than

$$\hat{\theta} = \arg\min_\theta \tilde{J}(\theta)$$

Let's try to characterize the change from θ^* to $\hat{\theta}$ over the entire course of training.

i. **[7 points]** Let $H_{\theta^*} := \nabla_\theta{}^2 J(\theta)\Big|_{\theta=\theta^*}$.

Under the second order Taylor approximation of the *regularized* objective $\tilde{J}$ about θ^*, solve for $\hat{\theta}$ as a function of H_{θ^*}, θ^* and c.

Answer: The gradient term $\nabla_\theta J(\theta)\Big|_{\theta=\theta^*}$ is equal to zero by definition of θ^* and thus:

$$\begin{aligned}\tilde{J}(\theta) &\underset{\theta\to\theta^*}{\approx} J(\theta^*) + \frac{1}{2}(\theta-\theta^*)^\top \nabla_\theta^2 J(\theta)\Big|_{\theta=\theta^*}(\theta-\theta^*) \\ &\quad + \frac{c}{2}\theta^{*\top}\theta^* + c(\theta-\theta^*)^\top\theta^* + c(\theta-\theta^*)^\top(\theta-\theta^*) \qquad (1)\end{aligned}$$

Differentiating Eq. (??) wrt θ yields:

$$\begin{aligned}\nabla_\theta \tilde{J}(\theta) &\approx H_{\theta^*}(\theta-\theta^*) + c\theta^* + c(\theta-\theta^*) \\ &= H_{\theta^*}(\theta-\theta^*) + c\theta \qquad (2)\end{aligned}$$

Since J is convex, the Hessian H_{θ^*} is psd and setting Eq. (??) to 0 yields the minimizer of $\tilde{J}$:

$$\hat{\theta} = \underbrace{(H_{\theta^*} + cI)}_{\text{non-singular since pd}}{}^{-1} H_{\theta^*}\theta^*$$

- *+0: Not attempted or completely copied the given Taylor expansion.*
- *+1: Attempted the problem, but got the wrong loss function.*
- *+2: Correct loss function $\tilde{J}(\theta)$. Attempted Taylor expansion.*
- *+3: Correct Taylor Expansion/Correct justification that $\hat{\theta}$ is the minimizer.*
- *+4: Correct derivation of $\nabla_\theta \tilde{J}(\theta)$.*
- *+5: Correct derivation and calculation, but failed to incorporate the definition of θ^* and included $\nabla_\theta J(\theta^*)$ (which should be 0) in the final answer.*
- *+6: Correct derivation. Minor calculation error (e.g. simplifying the final answer, missing 1/2).*
- *+7: Correct $\hat{\theta}$.*

ii. **[7 points]** Let $(o_1, \ldots, o_d)$ denote the orthonormal basis of eigenvectors of H_{θ^*}. Find the decomposition of $\hat{\theta}$ on $(o_1, \ldots, o_d)$, i.e. explicitly find the α_i's such that:

$$\hat{\theta} = \sum_{i=1}^{d} \alpha_i o_i$$

The expression of α_i's should **not** contain $\hat{\theta}$.
Hint 1: Any symmetric matrix S can be expressed as $O^\top \Lambda O$, where O is an orthonormal matrix and Λ is a diagonal matrix whose diagonal values are the eigenvalues of S (spectral theorem).
Hint 2: If O is orthonormal, then $O^\top = O^{-1}$.
Answer: Since H_{θ^*} is symmetric *(+1)*, then by the spectral theorem there exists O orthonormal and Λ such that $H_{\theta^*} = O^{-1}\Lambda O$ with $\Lambda = \mathbf{diag}(\lambda_i, i \in [\![1, d]\!])$ and $O = \begin{bmatrix} — & o_1^\top & — \\ & \ldots & \\ — & o_d^\top & — \end{bmatrix}$ *(+1)*.

$$\begin{aligned}
\hat{\theta} &= (O^{-1}\Lambda O + cI)^{-1} O^{-1}\Lambda O\theta^* \\
&= (O^{-1}(\Lambda + cI)O)^{-1} O^{-1}\Lambda O\theta^* \\
&= O^{-1}(\Lambda + cI)^{-1} O O^{-1}\Lambda O\theta^* \\
&= O^{-1}(\Lambda + cI)^{-1}\Lambda O\theta^* \qquad (3) \\
&= \sum_{i=1}^{d} \frac{\lambda_i}{\lambda_i + c} o_i^\top \theta^* o_i \qquad (4)
\end{aligned}$$

(+3 to get to Eq. (??) and +2 to get to Eq. (??))

iii. **[4 points]** Let λ_i denote the i^{th} eigenvalue for $i \in [\![1, d]\!]$. Give an interpretation of the above result for when $\lambda_i \gg c$ and when $\lambda_i \ll c$.
Answer: Thus the weight $\hat{\theta}$ is θ^* aligned with the orthonormal basis of eigenvectors of H_{θ^*}, with its i^{th} component rescaled to $\frac{\lambda_i}{\lambda_i + c}$ for $i \in [\![1, d]\!]$ *(+1)*. Since all the λ_i's are positive, this has the consequence of shrinking the weights along the directions where the eigenvalues of H_{θ^*} are small ($\lambda_i \ll c$) *(+1)* and leaving them unaffected along the directions where the eigenvalues are large ($\lambda_i \gg c$) *(+1)*. This analysis holds because the λ_i's are all non-negative since H_{θ^*} is psd *(+1)*.

iv. **[2 points]** For which objective function J (seen in class) is the above approximation exact and why?
Answer: The quadratic approximation is exact in linear regression *(+1)* since the objective is quadratic (and thus higher order derivates are all zero) *(+1)*.

6. **[22 points] Representer theorem**

In the supervised learning setting, we have input data $x \in \mathbb{R}^n$ and label $y \in \mathcal{Y}$.

(a) Specify the set $\mathcal{Y}$ in the case of:

i. **[1 points]** linear regression
Answer: $\mathcal{Y} = \mathbb{R}$

ii. **[1 points]** binary classification
Answer: $\mathcal{Y} = \{-1, 1\}$ or $\mathcal{Y} = \{0, 1\}$

iii. **[1 points]** multi-class (k-class) classification
Answer: $\mathcal{Y} = [\![1, k]\!]$.

(b) *(Representer theorem)*
For each of the above problems, we constructed a loss function $\ell : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$ with $\ell(\theta^T x, y)$ measuring the loss we suffer for predicting $\theta^T x$. For example, linear regression uses the squared residual for the loss: $\ell(\theta^\top x, y) = \frac{1}{2}(\theta^\top x - y)^2$. Let $\Omega : \mathbb{R} \to \mathbb{R}$ be a non-decreasing function. Consider the set of training examples $\{(x^{(i)}, y^{(i)}) \mid i \in [\![1, n]\!]\}$. Let $\tilde{J} : \mathbb{R}^d \to \mathbb{R}$ be the regularized objective across n examples:

$$\tilde{J} : \theta \mapsto \sum_{i=1}^{n} \ell(\theta^\top x^{(i)}, y^{(i)}) + \Omega\left(\|\theta\|_2\right)$$

Let $\theta \in \mathbb{R}^d$. In this question, we would like to show that there exists $\alpha \in \mathbb{R}^n$ such that:

$$\tilde{J}(\hat{\theta}) \leq \tilde{J}(\theta) \tag{5}$$

where

$$\hat{\theta} := \sum_{i=1}^{n} \alpha_i x^{(i)}$$

i. **[6 points]** Recall that for any subspace $\mathcal{S}$ of $\mathbb{R}^d$ (i.e. $S \subset \mathbb{R}^d$) we can define its orthogonal complement $\mathcal{S}^\perp = \left\{u \in \mathbb{R}^d \mid \forall v \in \mathcal{S}, u^T v = 0\right\}$. Also recall that the orthogonal decomposition theorem states that any vector $x \in \mathbb{R}^d$ can be uniquely written as:

$$x = x_{\mathcal{S}} + x_{\mathcal{S}^\perp}$$

where $x_{\mathcal{S}}$ and $x_{\mathcal{S}^\perp}$ are the projections of x onto $\mathcal{S}$ and $\mathcal{S}^\perp$, respectively.

Let $\theta \in \mathbb{R}^d$ and $\mathcal{S}_x := \text{span}\left\{x^{(i)} \mid i \in [\![1, n]\!]\right\}$. Denote $\theta_{\mathcal{S}}$ and $\theta_{\mathcal{S}^\perp}$ the projections of θ onto $\mathcal{S}_x$ and $\mathcal{S}_x^\perp$, respectively. Write the orthogonal decomposition of θ and express $\theta_{\mathcal{S}}$ and $\theta_{\mathcal{S}^\perp}$ in bases of $\mathcal{S}_x$ and $\mathcal{S}_x^\perp$, respectively. The values of the coefficients don't need to be explicitly written, but please do introduce relevant notation and specify the dimensions of each basis in terms of any of the following values: $\dim S_x$, n and d.

Answer: Decomposing θ into its component on $\mathcal{S}_x$ and $\mathcal{S}_x^\perp$ *(+1)*:

$$\theta = \theta_{\mathcal{S}} + \theta_{S^\perp} \tag{6}$$

Let $(u_1, \ldots, u_{d_0})$ be a basis of $\mathcal{S}_x$, with $d_0 = \dim \mathcal{S}_x$ *(+1)*. Similarly, let $(v_1, \ldots, v_{d-d_0})$ *(+1 for correct dimension)* be a basis of $\mathcal{S}_x^\perp$.
Then there exist $\{a_i \in \mathbb{R} \mid i \in [\![1, d_0]\!]\}$ and $\{b_i \in \mathbb{R} \mid i \in [\![1, d - d_0]\!]\}$ such that Eq. (??) can be re-written as:

$$\theta = \underbrace{\sum_{i=1}^{d_0} a_i u_i}_{\theta_{\mathcal{S}}} + \underbrace{\sum_{i=1}^{d-d_0} b_i v_i}_{\theta_{\mathcal{S}^\perp}} \tag{7}$$

(+2 for correct $\theta_{\mathcal{S}}$ and +1 for correct $\theta_{\mathcal{S}^\perp}$)
(Common mistakes: confusing $\dim \mathcal{S}_x$ with n and/or confusing n with d.)

ii. **[5 points]** Show that $\|\theta\|_2 \geq \|\theta_{\mathcal{S}}\|_2$.
Answer: From Eq. (??):

$$\begin{aligned}
\|\theta\|_2^2 &= \|\theta_{\mathcal{S}} + \theta_{\mathcal{S}^\perp}\|_2^2 \\
&= \|\theta_{\mathcal{S}}\|_2^2 + \|\theta_{\mathcal{S}^\perp}\|_2^2 + 2\underbrace{\theta_{\mathcal{S}}^\top \theta_{\mathcal{S}^\perp}}_{=0 \ (+1)} \\
&= \|\theta_{\mathcal{S}}\|_2^2 + \underbrace{\|\theta_{\mathcal{S}^\perp}\|_2^2}_{\geq 0 \ (+1)} \\
&\geq \|\theta_{\mathcal{S}}\|_2^2 \quad (+1)
\end{aligned}$$

Thus $\|\theta\|_2 \geq \|\theta_{\mathcal{S}}\|_2$ *(+1)* since both $\|\theta\|_2$ and $\|\theta_{\mathcal{S}}\|_2$ are non-negative *(+1)*.

iii. [**8 points**] Now prove the theorem.
Answer: We are going to start by showing that $\tilde{J}(\theta_{\mathcal{S}}) \leq \tilde{J}(\theta)$.
Part 1: $\Omega(\|\theta_{\mathcal{S}}\|_2) \leq \Omega(\|\theta\|_2)$ *(+1)* because $\|\theta_{\mathcal{S}}\|_2 \leq \|\theta\|_2$ (from previous part) *(+1)* and that Ω is non-decreasing *(+1)*.
Part 2: $\sum_{i=1}^{n} \ell(\theta^\top x^{(i)}, y^{(i)}) = \sum_{i=1}^{n} \ell(\theta_{\mathcal{S}}^\top x^{(i)}, y^{(i)})$ *(+1)* through the following steps *(+1)*.

$$\sum_{i=1}^{n} \ell(\theta^\top x^{(i)}, y^{(i)})$$
$$\sum_{i=1}^{n} \ell((\theta_{\mathcal{S}} + \theta_{S^\perp})^\top x^{(i)}, y^{(i)})$$
$$\sum_{i=1}^{n} \ell(\theta_{\mathcal{S}}^\top x^{(i)} + \theta_{S^\perp}^\top x^{(i)}, y^{(i)})$$
$$\sum_{i=1}^{n} \ell(\theta_{\mathcal{S}}^\top x^{(i)}, y^{(i)})$$

Note that $\theta_{S^\perp}^\top x^{(i)} = 0$ because by definition $\theta_{S^\perp}$ is orthogonal to every $x^{(i)}$ vector *(+1)*.

From part 1 and 2 we now have $\tilde{J}(\theta_{\mathcal{S}}) \leq \tilde{J}(\theta)$ as we have dealt with each term in $\tilde{J}$ and shown that each term is $\leq$ for $\theta_{\mathcal{S}}$.

Part 3: Let $\hat{\theta} = \theta_{\mathcal{S}}$ *(+1)*.
This is valid due to our work in 6.i. through the fact that $\theta_{\mathcal{S}}$ can be expressed as a combination of $x^{(i)}$'s just like $\hat{\theta}$ *(+1)*.

Thus proves the theorem as we have that $\tilde{J}(\hat{\theta}) \leq \tilde{J}(\theta)$.

The implications of this theorem are far-reaching. In particular, if $\tilde{J}$ can be minimized, then by virtue of the representer theorem, its minimizer admits the following representation:

$$\theta^* = \sum_{i=1}^{n} \alpha_i x^{(i)}$$

and we can replace any occurrence of $\theta^\top x$ with $\theta^\top x = \sum_{i=1}^{n} \alpha_i x^\top x^{(i)}$ which is handy because the kernel trick $\phi(x)^\top \phi(x^{(i)}) := K(x, x^{(i)})$ can then be applied for some high-dimensional mapping ϕ, and we can directly solve for α instead.

Congratulations! You've reached the end of the midterm.