CS 189 Introduction to Machine Learning
Fall 2017 Final

1 Potpourri: Part I (20 points)

For the following multiple choice questions, select all that apply. There is a partial credit scheme for some questions. **Bubble in your answers. Do not draw an X or just leave a check.**

(a) Which of the following is/are true? Select all that apply.

◯ In general, generative models have fewer parameters to fit.

◯ SVM is not a generative-model based classification scheme.

◯ Logistic regression is a discriminative model.

◯ Generative models are used for classification by estimating the probability that the observed data came from this model.

Solution:

Generative models tend to have more parameters to fit; for example, LDA and QDA have quadratic number of parameters. They can use these parameters to be able to estimate the probability density that the observed data came from this particular class model, thereby allowing us to pick the class that makes the observed data the most likely.

SVMs learn a decision boundary directly, while logistic regression is discriminative because it implicitly is learning the probability of being each class given the data.

Because some students might have been confused about whether we were referring to generative models being used to generate fresh synthetic data samples, we didn't penalize for not marking the last one as also being true.

(b) Which of the following statements is/are true? Select all that apply.

◯ The dual objective function for SVMs is cubic.

◯ The dual solution for SVMs is usually sparse.

◯ The SVM dual optimization only depends on the training points' positions $\vec{x}_i$ through the Gram matrix consisting of pairwise inner products between training points.

◯ In a hard margin classifier with margin M, all the training points can be perturbed by less than a quarter of the margin ($M/4$) and they still will be correctly classified with the same classifier.

Solution:

The dual objective for SVM is quadratic. The solution is usually sparse since the nonzero elements of the solution correspond to the support vectors. Because the dual only depends on the Gram matrix, we can kernelize SVMs. That was one of the main points of looking at the dual. The last option is correct since the training points will not move to a different side of the hyperplane if we perturb them by $M/4$, they would have to move the entire margin M in a specific direction to get to the other side.

(c) Which of the following can be naturally kernelized? Select all that apply.

◯ SVM

◯ Ridge Regression

◯ Decision Trees

◯ k-means

◯ A 3-layer Neural Network with sigmoid activation functions starting from raw data.

Solution:

SVMs can be kernelized because the dual problem is expressed in terms of the Gram Matrix. In class, we also saw how the ridge regression optimization problem can be expressed in terms of the Gram matrix. On this very exam and discussion, you saw how k-means can be kernelized.

Decision-trees are very feature based by their very nature and so cannot naturally be kernelized. A neural network that starts with the raw data and then processes it via nonlinearities cannot naturally be kernelized since nothing like the Gram matrix appears anywhere.

(d) Which of the following is/are true? Select all that apply.

◯ In random forests you always take an arithmetic mean for the final output.

◯ You can increase diversity in random forests by allowing a different subset of features in each tree.

◯ We want to have diverse trees in random forests to reduce the "bias" of the classifier.

◯ You can increase diversity in random forests by resampling training data subsets to build each tree.

Solution:

In random forests used for classification, we typically take a majority/plurality vote, not the arithmetic average.

The reason we want diverse trees in random forests is to increase the power of averaging (abstractly understood in a way that corresponds to everything for which we have a law of large numbers: max, median, plurality/mean, etc.) to reduce "variance" — our codeword for the data-dependent variability in our overall performance. Random forests don't really change "bias" — our codeword for the ability of the family of models to express the truth.

We can get diverse trees by bagging features (option b) or by bagging the data (option d), or more likely, combining both techniques.

(e) Which of the following can be used to reduce overfitting in Neural Networks? Select all that apply.

◯ Training longer

◯ Adding parameter L_1 norm penalty

◯ Weight sharing

◯ Injecting noise to input training data

◯ Using a Gaussian prior over the weight vector

Solution:

We reduce overfitting by doing things that make the optimization less sensitive to the vagaries of the specific data points. This is done by effectively making the model less expressive and more constrained. This is the general family of things generally called regularization. For example, an L_1 norm penalty.

Weight-sharing does this by literally reducing the degrees of freedom we have.

Injecting noise is something that you saw in the context of total least squares as behaving mathematically like regularization, it continues to do this for neural nets as well. You can also directly see it as making us less dependent on the actual data that we had.

Using a Gaussian prior is the same as an L2 regularizer.

Training longer has the opposite effect — it makes us more faithfully execute the optimization and hence moves the parameters more. Remember, regularization can be understood as putting a leash of sorts on the parameters and preventing them from moving too far. Training longer lets them move further to better (over)fit the data and thus is the opposite of regularization. (Stopping training early would be a kind of regularization.)

(f) Which of the following is/are true about the L_1 and the L_2 penalty? Select all that apply.

◯ L_1 penalty will often result in sparse solutions.

◯ L_2 penalty will often result in sparse solutions.

◯ L_2 penalty is equivalent to using an i.i.d. zero mean Gaussian prior on parameters.

◯ L_1 penalty is equivalent to using an i.i.d. Laplacian prior $\left(p(x) = \frac{1}{2b}\exp(-\frac{|x|}{b})\right)$ on parameters.

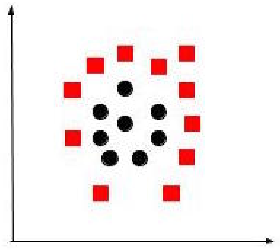
◯ Using L_1 regularization in linear regression will result in a cost function that is not convex.

Solution:

The L_2 penalty does not generally result in sparse solutions (after all, it is coming from an iid Gaussian prior, so why would it? The isocontours of the Gaussian prior are smooth spheres, why would any other optimization surface be more inclined to touch on an axis?), as opposed to the L_1 penalty which does (this is why LASSO works). L_1 regularization in linear regression will still result in a convex optimization problem, albeit one that cannot be minimized by gradient descent.

The fact that L_1 penalties are equivalent to iid Laplacian priors can be seen immediately from the log-likelihood function, although this fact does not give any additional insight into its sparsity-seeking behaviour. (And in fact, might even seem opposed to it since it has no Dirac delta at 0.)

(g) Which of the following classifiers can be used to classify the training data *perfectly*? Methods are not kernelized unless explicitly stated so.



○ Logistic regression

○ Hard margin SVM

○ 3-nearest neighbors

○ LDA

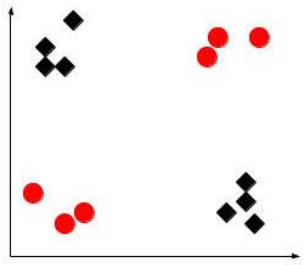
○ SVM with a quadratic kernel

○ QDA

Solution:

We need a classifier that can express a quadratic (circular) decision boundary, and among these, the SVM with quadratic kernel and QDA are capable of doing so. 3-nearest neighbors will not work: some of the squares will be misclassified as circles. LDA has a hyperplane boundary between two classes and so will logistic regression and the hard-margin SVM.

(h) Which of the following classifiers can be used to classify the data *perfectly*? Methods are not kernelized unless explicitly stated so.



◯ Logistic regression without a kernel

◯ Hard margin SVM

◯ 3-nearest neighbors

◯ LDA

◯ SVM with a quadratic kernel

◯ QDA

Solution: 3-nearest neighbors works since the closest 2 neighbors (besides itself) for each training point are of the same class as the training point. Again, SVM with quadratic kernel can express the boundary because a quadratic form can give us a hyperbola for the boundary. QDA is capable of separating the data because long diagonal ellipses can grab each class separately. (They might be tied in the middle, but we have no points there to worry about.)

To get LDA to work, we would have to pretend that there are actually four different classes, but we only have two. Same for logistic regression and hard-margin SVMs.

(i) Which of the following is/are correct about mini-batch stochastic gradient descent? Select all that apply.

◯ One iteration of mini-batch gradient descent over a mini-batch of samples is faster to compute than one iteration of batch gradient descent

◯ Using a smaller mini-batch size always leads to faster convergence in terms of the number of iterations.

◯ A larger mini-batch size reduces the variance in the estimate of the gradient.

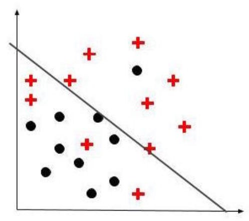
◯ If the mini-batch size is 1 you will process all the training samples before adjusting the parameters.

Solution:

Mini-batch gradient descent evaluates the gradient on fewer examples than batch gradient descent per iteration. So it is faster. But this estimate of the gradient is noisy, which will tend to lead to more iterations before convergence. Having a larger mini-batch size will result in a less noisy estimate of the gradient. So, a smaller one will have a more noisy estimate, which will in turn require us to slow down the step sizes to get more implicit averaging for convergence.

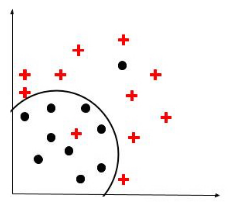
The whole point of mini-batch stochastic gradient descent is that we don't have to wait to see all the training samples before adjusting the parameters. We take steps based on the estimated gradient instead.

(j) For each classifier below, choose whether the classifier has high "bias," high "variance," or it is good enough for the given problem. Select all that apply.



◯ High bias ◯ High variance ◯ Reasonable

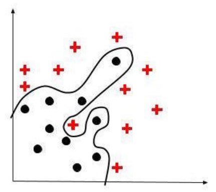
Solution: (a) The model (a straight line) has high bias but low variance. Depending on how you feel, you might also consider this reasonable given the number of data points.



◯ High bias ◯ High variance ◯ Reasonable

Solution:

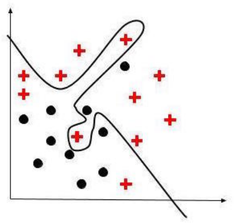
The decision boundary appears very regular (a circle) — suggesting low variance — and capable of separating most of the data, suggesting low bias. This makes is reasonable.



◯ High bias ◯ High variance ◯ Reasonable

Solution:

The decision boundary is irregular, following the quirks of the sampled data points, and this is a signature of "high variance". However, it classifies the points correctly, so there is no evidence of high bias. (for example, 1-NN can produce similar decision boundaries).



◯ High bias ◯ High variance ◯ Reasonable

Solution:

he decision boundary is irregular and fails to separate the data adequately, thereby suggesting that it both has high variance (follows the quirks of the data) and high bias at the same time.

2 Love thy Neighbors (20 points)

In class (and a short homework problem), we covered the `k-Nearest-Neighbors` algorithm. To classify a test point x_i, we looked at the k closest points in the training set and took a plurality vote among their classes. In this problem, we will consider a variation where instead of classifying x_i based on the k closest points, we classify it based on the plurality vote of all points within a radius of R. This is the `R-Radius-Neighbors` algorithm.

Furthermore, recall that the k-Nearest-Neighbors algorithm (and the R-Radius-Neighbors algorithm) are slow at test time, since they required computing a lot of distances. To speed things up, we are going to use the following idea. Rather than checking all the points in the training set, we will randomly sample 1% of the training data. In particular, for every data point i, we will toss an independent, biased coin with probability of heads equal to 0.01, and include data point i in our sampled dataset if the coin comes up heads.

(a) (4 points) Let x_i be a test point that had n radius-R neighbors in the original dataset (i.e., n points within radius R). **What is that probability that it has no neighbors within radius R after the sampling procedure above?** Express your answer in terms of n. Please write your *final* answer in the box below.

Solution: Each point is in the sample with probability 1%, and out otherwise, independently. The probability that all are out is therefore 0.99^n

(b) (5 points) Assume that we are in a binary classification setting, where the test point x_i had n^+ neighbors in the positive class and n^- neighbors in the negative class ($n = n^+ + n^-$). Let N^+ be a random variable which is the number of positive class neighbors of x_i after sampling the training data. Likewise, N^- is the number of negative class neighbors of x_i. **What is $E[N^+ - N^-]$?** Express your answer in terms of n^+ and n^-. Please write your *final* answer in the box below.

Solution: N^+ is a sum of n^+ Bernoulli trials with success probability 0.01, and N^- is similarly a sum of n^- Bernoulli trials with success probability 0.01. The answer is threfore $0.01(n^+ - n^-)$.

(c) (7 points) **Show that the variance** $\mathrm{var}(N^+ - N^-) = 0.0099n$**.**

Hint: Recall that a single Bernoulli trial with probability p has variance $p(1-p)$.

Solution: Variance of sum of n^+ (and n^-) Bernoullis. These are also independent of each other, and so the variance is $0.01 \cdot (1-0.01)n^+ + 0.01 \cdot (1-0.01)n^- = 0.099n$, where have used the fact that $n^+ + n^- = n$.

(d) (4 points) Say that x_i is an element of the test set with a positive label. We say that x_i will be classified correctly *with high probability* if $E[N^+ - N^-]$ is at least two standard deviations above 0. Mathematically, we write

$$E[N^+ - N^-] - 2\sqrt{\mathrm{var}(N^+ - N^-)} > 0.$$

In a huge data set, say that some x_i has $n = 10^4$ neighbors within radius R. Assume that the correctly label for x_i is the positive label. **How large does n^+ need to be in order for x_i to be classified correctly *with high probability* after sampling the training data?** Your answer should be an integer. You may assume that $\sqrt{0.0099} = 0.1$; recall that $n^+ + n^- = n$. Please write your *final* answer in the box below.

Solution: Plugging in expressions from the answers above, we have classification with high probability provided

$$\begin{aligned} 0.01(n^+ - n^-) > & \quad 2\sqrt{0.099n} \\ = & \quad 0.2\sqrt{n}. \end{aligned}$$

Noting that $n^- = n - n^+$ we can conclude that this is true provided

$$\begin{aligned} 2n^+ > & \quad 20\sqrt{n} + n \\ n^+ > & \quad 6000, \end{aligned}$$

where the last step follows by substituting $n = 10^4$.

The integer answer is thus 6001.

3 The k-Meaning of Life (32 points)

Recall that in k-means clustering we are attempting to minimize an objective defined as follows:

$$\min_{C_1, C_2, \ldots, C_k} \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|_2^2, \text{ where}$$

$$\mu_i = \mathrm{argmin}_{\mu_i \in \mathbb{R}^d} \sum_{x_j \in C_i} \|x_j - \mu_i\|_2^2 = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j, \quad i = 1, 2, \ldots, k.$$

The samples are $\{x_1, \ldots, x_n\}$, where $x_j \in \mathbb{R}^d$, and C_i is the set of samples assigned to cluster i. Each sample is assigned to exactly one cluster, and clusters are non-empty.

(a) (3 points) **What is the minimum value of the objective when $k = n$ (the number of clusters equals the number of samples)?** Please write your *final* answer in the box below.

Solution: The value is 0 since every point can get its own cluster.

(b) (5 points) **Prove that the best clustering on $k+1$ clusters is at least as good as the best clustering on k clusters (in terms of the objective function).** We assume $k \le n-1$.

Solution: Take the best clustering on k clusters. Take any cluster and assign one point in it to its own cluster. The objective function either improves (if that point had positive loss) or at worst stays the same (if that point was at the cluster center). Thus, the optimal solution on $k+1$ must be no worse.

(c) (8 points) Suppose we add a regularization term to the above objective. That is, the objective now becomes

$$\sum_{i=1}^{k}\left(\lambda\|\mu_i\|_2^2 + \sum_{x_j \in C_i}\|x_j - \mu_i\|_2^2\right).$$

Show that the optimum of

$$\min_{\mu_i \in \mathbb{R}^d} \lambda\|\mu_i\|_2^2 + \sum_{x_j \in C_i}\|x_j - \mu_i\|_2^2$$

is obtained at $\mu_i = \frac{1}{|C_i|+\lambda}\sum_{x_j \in C_i} x_j$.

Solution:

Method 1: (tricky)

We know from the homework that this is equivalent to adding λ points (may be fractional) at the origin to each cluster.

$$(\sum_{x_j \in C_i}\|x_j - \mu_i\|_2^2) + \lambda\|0 - \mu_i\|_2^2$$

Then the optimum is obtained at $\mu_i = \frac{1}{|C_i|+\lambda}\sum_{x_j \in C_i} x_j$, parallel to k-means.

Method 2:

Taking gradient w.r.t. μ_i for the function

$$f(\mu_j) = (\sum_{x_j \in C_i}\|x_j - \mu_i\|_2^2) + \lambda\|\mu_i\|_2^2,$$

we get

$$\begin{aligned}\nabla_{\mu_j} f(\mu_j) &= (2\sum_{x_j \in C_i}(\mu_i - x_j)) + 2\lambda\mu_i \\ &= 2((|C_i| + \lambda)\mu_i - \sum_{x_j \in C_i} x_j).\end{aligned}$$

Setting it to zero, we get $\mu_i = \frac{1}{|C_i|+\lambda}\sum_{x_j \in C_i} x_j$. As the function f is convex, the minimum is obtained at $\mu_i = \frac{1}{|C_i|+\lambda}\sum_{x_j \in C_i} x_j$.

Motivation: Here is an example where we would want to regularize clusters. The following bonus question was removed from the final for length considerations.

Suppose there are n students who live in a $\mathbb{R}^2$ Euclidean world and who wish to share rides efficiently to Berkeley for their final exam in CS189. The university permits k vehicles which may be used for shuttling students to the exam location. The students need to figure out k good locations to meet up. The students will then each *drive* to the closest meet up point and then the shuttles will deliver them to the exam location. Define x_j to be the location of student j, and let the exam location be at $(0,0)$. Assume that we can drive as the crow flies, i.e. by taking the shortest paths between two points. **Write an appropriate objective function to solve this ridesharing problem and minimize the total distance that all vehicles need to travel.**

Hint: We are not asking for the squared distance.

Solution:

$$\sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|_2 + \|\mu_i\|_2$$

To learn more about this problem, see the following article:

Cathy Wu, Ece Kamar, Eric Horvitz. *Clustering for Set Partitioning with a Case Study in Ridesharing*. IEEE Intelligent Transportation Systems Conference (ITSC), 2016.

(d) (10 points) Now we add a feature (lifting) map $\phi : \mathbb{R}^d \to \mathbb{R}^p$ with $p \gg d$, with the inner product in $\mathbb{R}^p$ expressed by a kernel function k, as

$$k(x,y) = \langle \phi(x), \phi(y) \rangle.$$

The objective of regularized k-means now becomes:

$$\sum_{i=1}^{k} \left(\lambda \|\mu_i\|_2^2 + \sum_{x_j \in C_i} \|\phi(x_j) - \mu_i\|_2^2 \right).$$

As in the previous part, the representative of the ith cluster is $\mu_i = \frac{1}{|C_i|+\lambda} \sum_{x_j \in C_i} \phi(x_j)$. **Show that** $\|\phi(x_j) - \mu_i\|_2^2$ **can be expressed only in terms of kernel functions** k**, input data** $x_j, j = 1, \dots, n$**, and partitions** C_i**, as:**

$$\|\phi(x_j) - \mu_i\|_2^2 = k(x_j, x_j) - \frac{2}{|C_i| + \lambda} \sum_{x_s \in C_i} k(x_s, x_j) + \frac{1}{(|C_i| + \lambda)^2} \sum_{x_s \in C_i} \sum_{x_t \in C_i} k(x_s, x_t).$$

Solution:

$$\begin{aligned}
\|\phi(x_j)-\mu_i\|_2^2 &= \langle \phi(x_j)-\mu_i, \phi(x_j)-\mu_i\rangle \\
&= \langle \phi(x_j), \phi(x_j)\rangle - 2\langle \mu_i, \phi(x_j)\rangle + \langle \mu_i, \mu_i\rangle \\
&= k(x_j,x_j) - 2\langle \frac{1}{(|C_i|+\lambda)} \sum_{x_s\in C_i} \phi(x_s), \phi(x_j)\rangle \\
&\quad + \langle \frac{1}{(|C_i|+\lambda)} \sum_{x_s\in C_i} \phi(x_s), \frac{1}{(|C_i|+\lambda)} \sum_{x_t\in C_i} \phi(x_t)\rangle \\
&= k(x_j,x_j) - \frac{2}{(|C_i|+\lambda)} \sum_{x_s\in C_i} \langle \phi(x_s), \phi(x_j)\rangle \\
&\quad + \frac{1}{(|C_i|+\lambda)^2} \sum_{x_s\in C_i} \sum_{x_t\in C_i} \langle \phi(x_s), \phi(x_t)\rangle \\
&= k(x_j,x_j) - \frac{2}{(|C_i|+\lambda)} \sum_{x_s\in C_i} k(x_s,x_j) \\
&\quad + \frac{1}{(|C_i|+\lambda)^2} \sum_{x_s\in C_i} \sum_{x_t\in C_i} k(x_s,x_t).
\end{aligned}$$

(e) (6 points) Based on the above part, we can design a regularized kernel k-means algorithm which avoids computation of μ_j in a higher dimensional space. We use c to denote the number of clusters. **Fill in the underlined portion of this algorithm.** No need to explain; you will only be evaluated on your answer.

Algorithm 1 Kernel K-means

```
Require: Data matrix $X \in \mathbb{R}^{n\times d}$; Number of clusters $c$, regularization $\lambda$.
Ensure: Cluster id $i(j)$ for each sample $x_j$.
  function KERNEL-K-MEANS($X, c$)
      Randomly initialize $i(j)$ to be an integer in $1, 2, \dots, k$ for each $x_j$.
      while not converged do
          for i=1,...,c do
              Set $C_i = \{j \in \{1, 2, \dots, n\} : i(j) = i\}$.
          end for
          for j=1,...,n do
              Set $i(j) = \text{argmin}_i$ Fill in the objective function here composed of kernel evaluations only.
          end for
      end while
      Return $C_i$ for $i = 1, 2, \dots, c$.
  end function
```

Solution:

The objective is the same as in the previous part, since we are minimizing the squared distance between $\phi(x_j)$ and μ_i over i:

$$k(x_j,x_j) - \frac{2}{|C_i|+\lambda} \sum_{x_s\in C_i} k(x_s,x_j) + \frac{1}{(|C_i|+\lambda)^2} \sum_{x_s\in C_i} \sum_{x_t\in C_i} k(x_s,x_t).$$

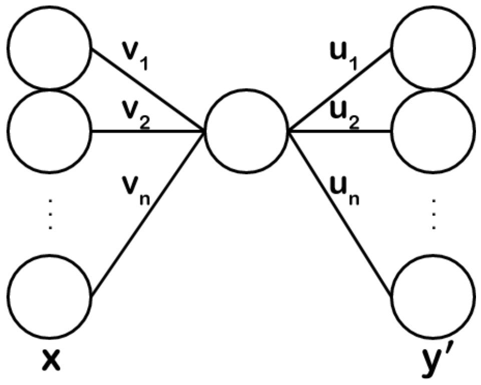
It is fine to omit the term $k(x_j, x_j)$ in the above solution, since it does not depend on i and will therefore not influence the arg min.

4 Neural networks and autoencoders (27 points)

To gain a better understanding of what neural networks learn, let's consider a simple network with two dense layers, linear activations, and no biases. This should remind you also of the simple linear autoencoder you saw in class, although here we will not be interested in making the input equal to the output. We will also restrict to the case of one neuron in the hidden layer. Because we use a linear activation (no nonlinearity between each layer), the output of the network can be given by

$$y' = uv^\top x$$

where the input $x \in \mathbb{R}^d$, the output $y' \in \mathbb{R}^d$, and the linear layers are given by $u \in \mathbb{R}^{d\times 1}$ and $v^\top \in \mathbb{R}^{1\times d}$. An example of the network is shown in the diagram below:



Given n samples, form an input matrix $X \in \mathbb{R}^{d\times n}$ by stacking samples in its columns, and the corresponding output matrix $Y \in \mathbb{R}^{d\times n}$ that is generated by stacking up the outputs as columns, but with noise. Assume $n \geq d$. We can express the squared error loss function as

$$L(u,v) = ||Y - uv^\top X||_F^2.$$

In this problem, we will assume that our input is whitened, i.e. $XX^\top = I$, and so the derivatives are given by

$$\frac{1}{2}\frac{\partial L}{\partial u} = -(Y - uv^\top X)(v^\top X)^\top = -YX^\top v + \|v\|_2^2 u, \text{ and}$$
$$\frac{1}{2}\frac{\partial L}{\partial v^\top} = -u^\top (Y - uv^\top X)X^\top = -u^\top YX^\top + \|u\|_2^2 v^\top.$$

In the following parts, you will:

- Characterize all points where the loss function $L(u,v)$ has derivative zero.

- Show that the optimal u and v that minimizes $L(u,v)$ is related to a low-rank approximation of $YX^\top$.

Let us use the notation $Z \triangleq YX^\top$ for convenience.

(a) (10 points) To start, let's identify a set of stationary points in the loss. We will represent the matrix of interest via its SVD as

$$Z \triangleq YX^\top = U\Sigma V^\top = \sum_{i=1}^{d} \sigma_i u_i v_i^\top$$

Assume that there are d nonzero singular values so that all the matrices $U, \Sigma, V^\top \in \mathbb{R}^{d\times d}$, with $\sigma_1 > \sigma_2 > \ldots > \sigma_d > 0$. The matrix Σ is thus a diagonal matrix of singular values.

Show that the choice $u' = u_i$ and $v' = \sigma_i v_i$ is a stationary point of the function L (i.e. having zero derivative with respect to both its arguments), for every choice of the index $i = 1, 2, \ldots, d$.

[What is true but won't be shown in this problem is that these are the *only* stationary points of the loss function.]

Solution: Plugging into the derivatives of the loss, we see that

$$\frac{1}{2}\frac{\partial L}{\partial u} = -Zv' + \|v\|_2^2 u' = -Z\sigma_i v_i + \sigma_i^2 u_i = \left(\sum_{j=1}^{d} \sigma_j u_j v_j^\top\right)(\sigma_i v_i) + \sigma_i^2 u_i = \sigma_i^2 u_i - \sigma_i^2 u_i = 0,$$

where we have used the fact that the vectors are orthonormal. A similar calculation with the derivative $\frac{1}{2}\frac{\partial L}{\partial v^\top}$ yields the result for the derivative with respect to $v^\top$.

(b) (12 points) Among all the stationary points from the previous part, let us now compute the minimum. By evaluating the loss function L explicitly at all the stationary points $(u', v') = (u_i, \sigma_i v_i)$ for all $i \in \{1, 2, \ldots, d\}$, **show that the optimal argument is given by $(u^*, v^*) = (u_1, \sigma_1 v_1)$.**

Hint: You may use the fact that the loss function can be written as

$$L(u,v) = \text{trace}(Y^\top Y) - 2\text{trace}(YX^\top vu^\top) + \|u\|_2^2 \text{trace}(vv^\top),$$

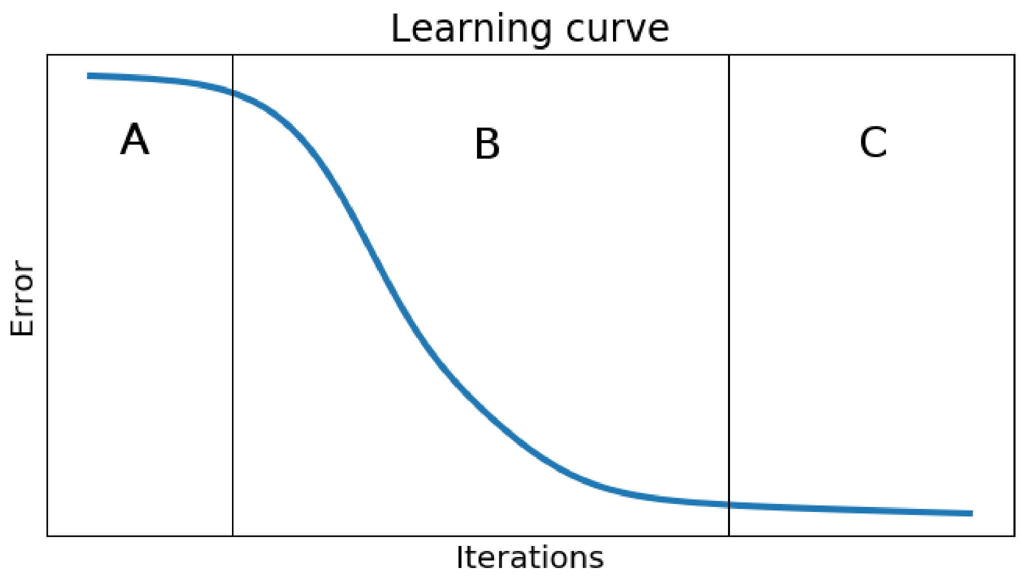
where trace denotes the sum of diagonal entries of a square matrix.

Solution: Using the hint, we have

$$L(u,v) = \text{trace}(Y^\top Y) - 2\text{trace}(Zvu^\top) + \|u\|_2^2 \text{trace}(vv^\top).$$

We know that stationary points take the form $(u_i, \sigma_i v_i)$, so evaluating the loss at these points yields

$$\begin{aligned} L(u_i, \sigma_i v_i^\top) &= \text{trace}(Y^\top Y) - 2\text{trace}(\sum_{i=1}^{d} \sigma_j u_j v_j^\top)\sigma_i v_i u_i^\top) + \sigma_i^2 \text{trace}(v_i v_i^\top) \\ &= \text{trace}(Y^\top Y) - 2\sigma_i^2 \text{trace}(u_i u_i^\top) + \sigma_i^2 \text{trace}(v_i v_i^\top) \\ &= \text{trace}(Y^\top Y) - 2\sigma_i^2 \|u_i\|_2^2 + \sigma_i^2 \|v_i\|_2^2 \\ &= \text{trace}(Y^\top Y) - \sigma_i^2. \end{aligned}$$



This is minimized when σ_i is maximized at σ_1, and so the optimal argument is given by $(u^*, v^*) = (u_1, \sigma_1 v_1)$.

(c) (5 points: BONUS) While we didn't show it in this problem, all stationary points except for the global minimum are saddle points, i.e., there is some descent direction for these points, and they are not local minima. With this in mind, answer the following question.

You are given the following learning curve when running gradient descent on this problem with the "correct" step-size, guaranteeing that you won't diverge. At the end, u and v converge to the optimal solution. **Indicate which of the choices below describes what is happening in regions (A), (B), and (C) of the plot.** Each answer will be used at most once, and only one choice describes each region. Please bubble in your *final* answer in the corresponding blanks.

1. u, v has essentially converged to the global maximum $\arg\max_{u'', v''} L(u'', v'')$.
2. u, v has essentially converged to the global minimum $\arg\min_{u'', v''} L(u'', v'')$.
3. u, v is near a saddle point.
4. u, v is not close to any stationary point.

Solution:

A. 3: Here, the proximity to saddle point(s) is causing the descent to be slow because the gradient is small.

B. 4: Here, there is no stationary point nearby and so we are being strongly attracted by one of them.

C. 2: Here, we have essentially converged to the global minimum since we know that there is only one local minimum and it is global.

5 Dimensionality Reduction and Classification (57 points)

In this problem, you will explore different methods to transform data onto a lower-dimensional space. A good dimensionality reduction procedure is one that allows us to simultaneously preserve the structure in the dataset that we want to discover, while affording some form of compression that can then be used to speed up computations downstream.

We will work with a matrix $S \in \mathbb{R}^{s \times d}$ that transforms data points in d-dimensions to s-dimensions, i.e., for a point $x \in \mathbb{R}^d$, its transformation is given by the vector $Sx \in \mathbb{R}^s$. Assume that our dataset consists of n data points $x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(n)}$, each in d dimensions; half the data are from class C and the other half from class D. The labels are given by $y^{(i)} = 1$ if data point i has label C, and -1 otherwise. For concreteness, you should think of each data point as an image in high-dimensions, and we have half the images labeled $C =$ "*cats*" and the other half labeled $D =$ "*dogs*".

We have the following **three** assumptions on our data:

- For some $\varepsilon \geq 0$, the mean of data points in each of the classes is given by

$$\mu_C = (-\varepsilon, 0, 0, \ldots, 0) \in \mathbb{R}^d, \text{ and}$$
$$\mu_D = (\varepsilon, 0, 0, \ldots, 0) \in \mathbb{R}^d.$$

- We have the empirical covariances

$$\Sigma_C = \Sigma_D = \mathsf{diag}\left(\frac{1}{d^2}, \frac{1}{(d-1)^2}, \ldots, \frac{1}{2^2}, 1\right),$$

  where $\mathsf{diag}(v)$ denotes a $d \times d$ diagonal matrix having $[\mathsf{diag}(v)]_{ii} = v_i$.

- The maximum Euclidean distance of any point from its class mean is at most 5, i.e.,

$$\|x^{(i)} - \mu_C\|_2 \leq 5 \text{ for all } i \text{ labeled "}C = cat\text{"}$$
$$\|x^{(i)} - \mu_D\|_2 \leq 5 \text{ for all } i \text{ labeled "}D = dog\text{"}.$$

It may be helpful to draw yourself a schematic to visualize the data.

In parts (a) and (b), we will explore some standard data-dependent transformation algorithms that you have studied. In parts (c), (d), and (e), you will study some classification approaches in the original d-dimensional space. In the bonus part (f), you will be introduced to a data-independent transformation procedure, and you will see that classification can now be performed in a lower dimensional space, more quickly than before.

(a) (5 points) First, let us set the transformation matrix S such that we perform PCA on the *entire dataset* (including both classes C and D together), but ignoring labels. In effect, we want to transform the data onto the top s eigenvectors of the covariance matrix Σ of the entire dataset.

Show that if $\varepsilon = 0$, the ith row of S is given by $S_i^\top = e_{d-i+1}^\top$, where $e_i \in \mathbb{R}^d$ is the ith standard basis vector in d-dimensions (e.g. $e_1 = (1, 0, 0, \ldots, 0) \in \mathbb{R}^d$).

Solution: When $\varepsilon = 0$, the covariance of the entire dataset is equal to the covariance of each of the classes, so PCA will simply pick out the top s eigenvectors of the matrix Σ_C. That is given by the answer above since the relevant matrix is diagonal (so the eigenvectors are the standard basis) with sorted eigenvalues (so the relevant eigenvectors are the last ones).

(b) (6 points) Let us now bring back the labels as one-hot vectors in $\mathbb{R}^2$, i.e., $\ell^{(i)} = (1,0)$ if $y^{(i)} = 1$, and $\ell^{(i)} = (0,1)$ otherwise. For any fixed $\varepsilon > 0$ (no longer equal to zero), let us now perform CCA for predicting ℓ from x. This is run on the dataset by transforming all the data $\{x^{(i)}\}_{i=1}^n$ onto one vector by computing $Sx^{(i)}$, but $S \in \mathbb{R}^{1\times d}$ is now a row vector. **Which vector S would CCA choose, if we constrained the vector to be unit norm?** Please write your *final* answer in the box below.

Hint: Recall that LDA is a useful technique when the class covariances are the same. Also recall from homework that we established a precise connection between LDA and CCA.

Solution: The LDA on two classes with equal covariances is implicitly defined by the difference of their two means (viewed through the lens of the covariance), and this is equivalent to the first CCA direction, as you saw in the homework. Hence, we have that this direction is given by $\Sigma_C^{-1}(\mu_C - \mu_D) = d^2\varepsilon e_1$. Normalizing, we have that the direction is equal to e_1.

(c) (8 points) **What is the minimum value of ε such that a 2-nearest neighbour algorithm can be guaranteed to incur zero error when a test data point is drawn from one of the training examples $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$?**. Please write your *final* answer in the box below.

Hint: Recall all three assumptions on our data.

Solution: The actual answer is 7.6, for any dimension d. However, we will accept any justified solution of ε between 7.5 and 10. The two corner points are given below, and the rigorous solution follows.

We know that all the action happens along the first dimension. We want the set of 2 nearest neighbors to be preserved, so $\varepsilon > 10$ suffices. This should be clear in one dimension, where we have points at $-\varepsilon-5, -\varepsilon+5, \varepsilon-5, \varepsilon+5$. Setting $\varepsilon < 10$ for this example changes the two nearest neighbours for the points at $-\varepsilon+5$ and $\varepsilon-5$.

Notice that the above solution does not incorporate the fact that the data must preserve a covariance of 1 along the first dimension. In order to do this, we now have $n/2$ points at $\varepsilon - \varepsilon'$, and one point at $-\varepsilon+5$, all labeled C. (Here ε' is assumed tiny.) Take a reflection about 0 and call all those points D. By taking $\varepsilon' \to 0$ and making n large enough, it is possible to additionally preserve the covariance constraint and well as force the mean of the class C to be $-\varepsilon$. The distance between the point at $-\varepsilon+5$ to its closest neighbor in the class is now 5, so in order to preserve this set, it suffices for the distance between $-\varepsilon+5$ and $\varepsilon-5$ to be greater than 5. In particular, this means that we must have $\varepsilon > 7.5$.

The rigorous argument goes as follows: Again, first restrict to one dimension. The worst case for the nearest neighbors algorithm is if the closest point within the class is far away, and the closest point of the opposite class is close by. In particular, the second condition means that there is a point of class C at $-\varepsilon+5$, and a point of class D at $\varepsilon-5$. How do we make sure that the variance along this dimension is 1 while making the rest of the points far away from the point at $-\varepsilon+5$?

You need to add n points at $-\varepsilon-t$ for some $t>0$. Solving for n and t by setting the mean and variance to $-\varepsilon$ and 1 leads to the choice $t=0.2$, and $n=25$. This makes the intra class distance 5.2, and the distance between classes $2\varepsilon-10$. Making these equal (in the worst case), we are led to the solution $\varepsilon=7.6$.

Now what happens in two dimensions? The variance along the first dimension now becomes $1/4$, so repeating the above argument yields that we must have 100 points at $-\varepsilon-0.05$. Since we also need to make sure the variance along the second dimension is 1, we place 50 points each at $(-\varepsilon-0.05,1)$, and $(-\varepsilon-0.05,-1)$. This makes the intra class distance $\sqrt{5.05^2+1}\approx 5.15$ (less than before), while preserving the inter-class distance. We thus see that moving to two dimensions only makes the problem easier than before. This was not special, since the same argument applies inductively to any dimension. Consequently, staying in $d=1$ dimension is optimal, which should have struck you intuitively, since you can maximize the variance in this case.

We didn't want you to make as intricate an argument as this, so we decided that ignoring the variance was fine as long as you realized that restricting to one dimension was the right way to go. Hence the officially accepted range of ε.

(d) (13 points) Let $\varepsilon=20$. **Show that there must exist a hyperplane separator having margin (distance between the hyperplane and the closest point of any one class) at least** 15. Proof by picture is fine.

With $\varepsilon=20$, **which of the following hyperplanes could be a max-margin separator of the two classes? Bubble in and justify your answer.** (Here, x_i denotes the ith coordinate of the d-dimensional space, e.g. $i=1$ corresponds to the first coordinate.)

Hint: Recall all three assumptions on our data.

- $x_1=0$.

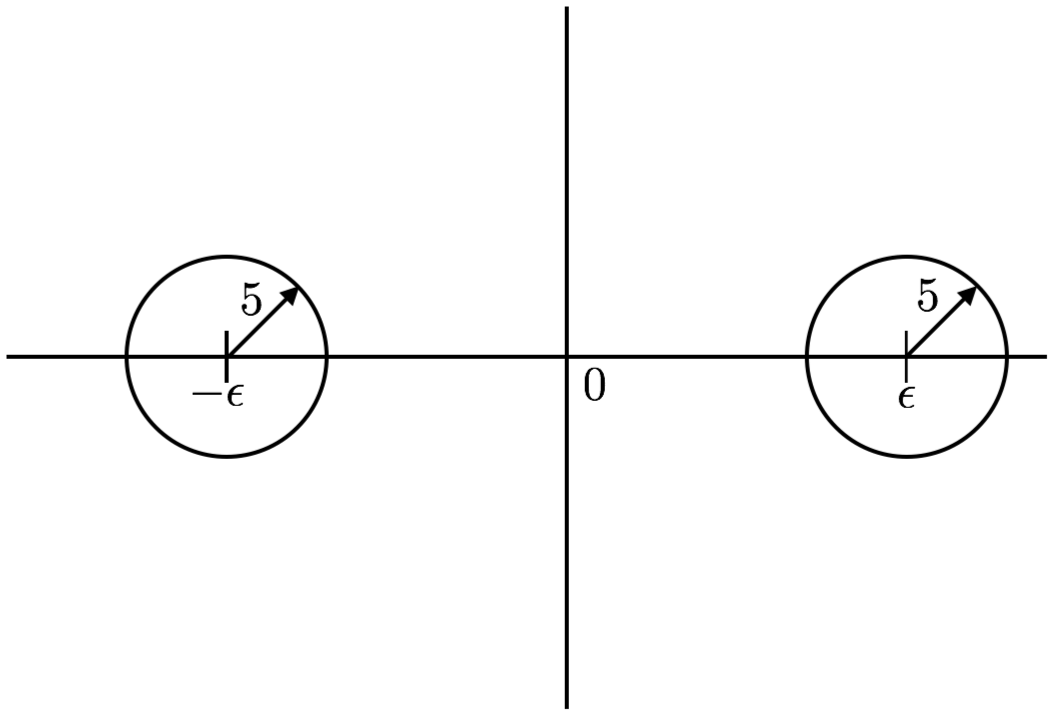
  ◯ Yes ◯ No

  Justification:

- $x_2=0$.

  ◯ Yes ◯ No

  Justification:



- $x_1 = 10$.

  ◯ Yes ◯ No

  Justification:

Solution: The fact that a hyperplane separator exists when $\varepsilon = 20$ should be clear from the following figure in two dimensions. A similar figure is true in d dimensions.

Let us now think about which of the axes could be a max-margin separator. $x_1 = 0$ certainly can be, since it corresponds to the y-axis in the above plot, and is the separator when there are points at $-\varepsilon + 5$ and $\varepsilon = 5$.

$x_2 = 0$ cannot, since since it does not separate the two classes in the first place.

$x_1 = 10$ is a linear separator but cannot be a max-margin separator, since the closest point of class D must be at distance less than $\varepsilon - 10 = 10$, but the closest point of class C is at distance at least $10 - (-\varepsilon + 5) = 25$.

(e) (10 points) With $\varepsilon = 20$, **which of the following methods would find a perfect (not necessarily max-margin) linear separator between the points? Again, bubble in and justify your answer.** Remember to recall all the assumptions on our data.

Notice that any linear separator can be written as $\widehat{w}^\top x + \widehat{b} = 0$. We set $\widehat{b} = 0$, and $\widehat{w}$ is obtained via

- $\widehat{w} = \arg\min_w \|w\|_2^2$ s.t. $y^{(i)} w^\top x^{(i)} \geq 1$ for all $i = 1, 2, \ldots, n$.

  ◯ Yes ◯ No

  Justification:

- Let the optimal value of the previous problem be denoted by

$$p^* = \min_w \|w\|_2^2 \text{ s.t. } y^{(i)} w^\top x^{(i)} \geq 1 \text{ for all } i = 1, 2, \ldots, n, \text{ and define}$$

$$\widehat{w} = \arg \min_{w : \|w\|_2^2 \geq p^*} \|w\|_2^2 + \sum_{i=1}^{n} \max(0, 1 - y^{(i)} w^\top x^{(i)}).$$

  ◯ Yes ◯ No

  Justification:

Solution: We established that there is a linear separator, so method 1, corresponding to the hard margin SVM, will find it.

Method 2 uses the hinge loss, but only searches for w such that $\|w\|_2$ is larger than the corresponding norm for the hard-margin SVM. This is also guaranteed to find a linear separator.

To justify this, fix the norm of w to be some value. We know that since this value is greater than that of the hard margin SVM, all terms in the hinge loss can be make zero by simply scaling the w^* found by the hard margin SVM. In particular, this means that we obtain a perfect classifier (linear separator) for every value of $\|w\|_2^2$. In order to minimize $\|w\|_2^2 + \sum_{i=1}^n \max(0, 1 - y^{(i)} w^\top x^{(i)})$, it therefore suffices to minimize $\|w\|_2$ alone, which yields the same separator as in the hard margin case.

In some exams, the problem was printed with a bug. No constraint was provided on w. For those, the argument becomes trickier and we will accept any well reasoned answer. The thing we fear is that a much larger margin is possible if we now misclassify some points.

(f) (15 points: BONUS) In parts (a) and (b), we saw two data-dependent dimensionality reduction techniques. Let us now choose (in a *data-independent* manner) the entries of the transformation matrix $S \in \mathbb{R}^{s \times d}$ i.i.d. as

$$S_{ij} \sim \mathcal{N}(0, 1/s).$$

By a miracle of probability, it turns out that provided we choose $s \geq \frac{3 \log n}{\delta^2}$, we can guarantee that all n points $\{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$ can be transformed onto s dimensions such that

$$(1-\delta)\|x^{(i)} - x^{(j)}\|_2 \leq \|Sx^{(i)} - Sx^{(j)}\|_2 \leq (1+\delta)\|x^{(i)} - x^{(j)}\|_2$$

for all pairs $i \neq j$ simultaneously with probability greater than $1 - 1/n$.

Let us now set $s = 12 \log n$. Use the fact above to show that the resulting dataset $Sx^{(1)}, Sx^{(2)}, \ldots, Sx^{(n)}$ is still linearly separable in s-dimensional space with probability greater than $1 - 1/n$.

Is the 2 nearest neighbor algorithm still guaranteed to work after the random transformations?

Hint: Think about the value of δ that setting $s = 12 \log n$ corresponds to.

Solution: As clarified during the exam, we were assuming here that $\varepsilon = 20$.

The value of s here corresponds to setting $\delta = 1/2$. In other words, all pairwise distances are between $1/2$ and $3/2$ of the original pairwise distance. What does this mean? The new means of the datasets can now be as small as $\varepsilon/2 = 10$ and $-\varepsilon/2 = -10$, the radius of each class can increase to at most 7.5. In particular, this means that the closest points of the two classes are still on opposite sides of the origin, so a linear separator exists.

A 2-NN algorithm, now, however, is no longer guaranteed to succeed, since by the logic above, we may now have points in the cat class at -2.5 and $n/2$ points close to -10; however, the closest point in the other class is at 2.5. One can force this to succeed by setting $\delta \ll 1/2$, but this would require a correspondingly larger number of transformation directions s.

6 Potpourri: Part II (10 points)

For the following multiple choice questions, select all that apply. There is a partial credit scheme for some questions. You will be graded on your best *five* out of the ten questions in this section, so feel free to skip things you're unsure about.

(a) Which of the following statements is/are true about SVMs? Select all that apply.

◯ We find at least one support vector from each class.

◯ Soft margin SVM effectively optimizes hinge loss plus ridge regularization.

◯ If a training point has a positive slack, then it is a support vector.

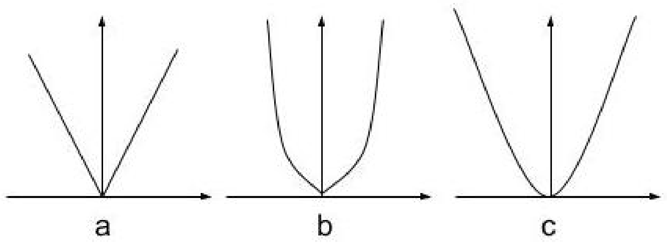
◯ Hinge loss is convex.

Solution:

If there are no support vectors from class +, say, then the hyperplane can be shifted further away from the - support vectors to increase the margin. Training points with positive slack have to correspond to $\alpha_i^* = C$ in the dual formulation, and such points lie inside the margin (or on the wrong side) and thus must be support vectors.

Hinge loss is convex and soft-margin SVM can be formulated as a regularized hinge loss optimization problem, allowing us to use convex optimization algorithms to solve it. The slack variables are basically coming from hinge-loss itself. For a slack variable to be nonzero, that particular point must be incurring some hinge loss relative to the classifier. (i.e. it is inside the margin or on the wrong side)

(b) Which of the following loss functions tends to result in a sparse solution? Select all that apply.



◯ a ◯ b ◯ c ◯ None

Solution:

(a) and (b) have a sharp corner at 0, so like the L_1 loss, these will tend to induce sparsity. (c) is smooth, and as with L_2 loss, will not induce sparsity.

(c) Which of the following is true about decision trees? Select all that apply.

◯ Decision trees are less interpretable as compared to other classifiers.

◯ Decision trees are not prone to overfitting.

◯ Decision trees are sensitive to the choice of feature ordering.

◯ As compared to other classifiers, it is easier to deal with missing features in training points.

Solution:

Decision trees are known for interpretability (we can explain why a point is classified in a particular way based on a sequence of judgements), but are prone to overfitting on their own (since a deep tree can approximate any boundary). The order in which the features are explored can determine how the tree turns out, because the splits are chosen greedily and the previous splits influence the quality of subsequent ones. Decision trees can be easily modified to deal with missing features - for example, during a split one can only consider those examples which have defined values for that feature and distribute the entries with missing data to both sets of child nodes.

(d) Which of the following is/are true about AdaBoost? Select all that apply.

◯ The classifier is a linearly weighted combination of different classifiers.

◯ AdaBoost stops if it finds a classifier with zero training error.

◯ AdaBoost at step t reweights the training data so the boosted classifier found at step t is not better than a random guess.

◯ We can only use AdaBoost ideas on decision trees.

Solution:

The ultimate classifier that AdaBoost builds linearly weights the classifiers it finds along the way with parameters relating to each classifer's training error. This is why it is a kind of matching pursuit for classification. Because it is not trying to optimize classification loss directly (instead looking at an exponential loss), it can still update the weights and increase generalization performance even when the training error reduces to zero. Because this is classification and not regression, the concept of the "residual" in matching pursuit is accomplished by reweighting the training data. The concept of the "error being orthogonal to the last basis vector found" shows up as reweighting the training data so that the classifier just found is not better than a random guess.

Because it is a kind of matching pursuit that treats classifiers as features, one can use AdaBoost effectively with any classifier that performs better than random guessing. It doesn't have to be used with trees.

(e) Which of the following is/are true about GANs? Select all that apply.

◯ GAN is a supervised learning algorithm.

◯ In GAN we have two neural networks competing with each other.

◯ During GAN training, the generator's effective loss function changes.

◯ In GAN training, the gradient backpropagates through the generator into the discriminator.

Solution:

Standard GAN requires no labels and is consequently unsupervised. (A conditional GAN would use the labels the same way that LDA or QDA use the labels.) GANs are trained adversarially with competing generator and discriminator nets, and the generator's effective loss function is a function of the discriminator's loss and thus changes during training as the discriminator evolves. This is important because of how the generator is trained — the discriminator weights are frozen during the generator updates and the gradient propagates through the discriminator into the generator for the generator's updates. In other words, the generator tries to improve so as to better fool this particular discriminator. The reverse is not true. The discriminator is trained with examples presented from the generator and from the actual training data. The generator net is not involved in the loss function at all and no gradients are propagated through it.

(f) Assume that we are computing weights for a regression problem without any regularization and the weight vector is $w = [102, 431, 95]^T$. We tried different norm penalties to reduce variance and the resulting weight vectors are $w_1 = [4.35, 8, 1.65]^T$, $w_2 = [0, 1.87, 2.34]^T$, $w_3 = [0, 15.87, 23.7]^T$, and $w_4 = [10.31, 18, 10.65]^T$. Which of the following could be true? λ is the regularization hyperparameter.

◯ $w_1 = [4.35, 8, 1.65]^T$ can be the result of applying L_2 (ridge) regularization.

◯ The weights $w_3 = [0, 15.87, 23.7]^T$ result in a predictor that is more sensitive to perturbations of the input x as compared to the weights $w_2 = [0, 1.87, 2.34]^T$.

◯ $w_3 = [0, 15.87, 23.7]^T$ could be the result of applying L_1 regularization

◯ We applied a smaller λ to compute $w_1 = [4.35, 8, 1.65]^T$ compared to the λ used to compute $w_4 = [10.31, 18, 10.65]^T$

Solution:

All are true except the last one (d).

In reality, a larger λ penalizes the weights more heavily, and hence results in a decrease in the norm of the weights.

We know the L_1 loss is sparsity seeking and hence more likely to favor weight vectors where some entries are zero.

In general, we know that regularization is a way to shrink the weights and thereby make ourselves less sensitive to input perturbations.

(g) Which of the following is/are true about LDA and QDA? Select all that apply.

◯ In general, QDA is more prone to overfitting than LDA.

◯ QDA and LDA are generative approaches for classification.

◯ In QDA the decision boundaries are always hyperplanes.

◯ LDA has fewer parameters to learn than QDA.

Solution:

QDA learns a more complex, nonlinear decision boundary and thus is more prone to overfit. QDA has more parameters and that is what lets it express more complex boundaries.

QDA and LDA are both generative classification approaches that fit Gaussian distributions to the data, but QDA estimates a covariance matrix for each class, while LDA assumes the class-conditional covariance matrices are equal. The equal class-conditional covariances force the decision boundaries for LDA to locally be hyperplanes.

(h) Which is/are true about k-nearest neighbors (kNN)? Select all that apply.

◯ 1NN would have zero training error on the training set itself.

◯ Increasing k in kNN tends to result in less overfitting.

◯ You can use different distance measures for kNN.

◯ kNN always learns linear decision boundaries.

◯ kNN cannot be used with dimensionality reduction.

Solution: 1NN will perfectly classify the training set, since each queried point would have itself as its nearest neighbor. Increasing k represents a kind of averaging, and hence it decreases variance. This makes the boundaries more regular and this tends to reduce overfitting.

kNN is flexible as to the choice of distance metric but regarless of the metric, kNN typically learns nonlinear decision boundaries. It can be used very effectively with dimensionality reduction to alleviate the curse of dimensionality as well as to reduce the variance.

(i) Which of the following function classes can be represented exactly by a neural network with one hidden layer with threshold activation functions (i. e. step function) and a final output that is linear? Select all that apply.

◯ Polynomials of degree two

◯ Piecewise linear functions

◯ All continuous functions

◯ Piecewise constant functions

Solution:

Neural nets with step nonlinearities can universally approximate any function, but are only capable of exactly representing piecewise constant functions.

If you interpreted threshold activation function using the definition of "thresholding activation functions" given in the homework, even there, the assumption was that the functions saturate and do not grow unboundedly. This means that there is no way to do piecewise linear exactly since that would require a ReLU style activation function that is not saturating.

(j) Assume that k_1 is a valid kernel taking $\mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$. Which of the following is/are necessarily a valid kernel for all valid choices of k_1 and k_2? Select all that apply.

○ $k(x,z) = k_1(x,z) + C$ where $C > 0$.

○ $k(x,z) = k_1(f(x), f(z))$ where $f : \mathbb{R}^d \mapsto \mathbb{R}^d$.

○ $k(x,z) = h(k_1(x,z))$ where $h : \mathbb{R} \mapsto \mathbb{R}$ is a degree 2 polynomial.

○ $k(x,z) = k_1(f(x), g(z))$ where $f : \mathbb{R}^d \mapsto \mathbb{R}^d$ and $g : \mathbb{R}^d \to \mathbb{R}^d$ are different functions.

Solution:

The first one is true - the Gram matrix is $K + CI_n$, which is PSD if K is. The second is also true: if Φ is the induced feature map for k_1, then the induced feature map for k is $\Phi \circ f$.

The last two are not true in general because we can have negative C for general quadratics, and a lack of symmetry for the last.