Q1. [90 pts] Multiple Choice

Check the boxes for **ALL** CORRECT CHOICES. Every question should have at least one box checked. NO PARTIAL CREDIT: the set of all correct answers (only) must be checked.

(1) [3 pts] What strategies can help reduce overfitting in decision trees?

- ■ Pruning
- ☐ Make sure each leaf node is one pure class
- ■ Enforce a minimum number of samples in leaf nodes
- ■ Enforce a maximum depth for the tree

(2) [3 pts] Which of the following are true of convolutional neural networks (CNNs) for image analysis?

- ■ Filters in earlier layers tend to include edge detectors
- ☐ They have more parameters than fully-connected networks with the same number of layers and the same numbers of neurons in each layer
- ■ Pooling layers reduce the spatial resolution of the image
- ☐ A CNN can be trained for unsupervised learning tasks, whereas an ordinary neural net cannot

(3) [3 pts] Neural networks

- ☐ optimize a convex cost function
- ■ can be used for regression as well as classification
- ☐ always output values between 0 and 1
- ■ can be used in an ensemble

(4) [3 pts] Which of the following are true about generative models?

- ■ They model the joint distribution $P(\text{class} = \text{C AND sample} = \mathbf{x})$
- ■ They can be used for classification
- ☐ The perceptron is a generative model
- ■ Linear discriminant analysis is a generative model

(5) [3 pts] Lasso can be interpreted as least-squares linear regression where

- ■ weights are regularized with the ℓ_1 norm
- ☐ weights are regularized with the ℓ_2 norm
- ☐ the weights have a Gaussian prior
- ☐ the solution algorithm is simpler

(6) [3 pts] Which of the following methods can achieve zero training error on *any* linearly separable dataset?

- ■ Decision tree
- ■ Hard-margin SVM
- ☐ 15-nearest neighbors
- ■ Perceptron

(7) [3 pts] The kernel trick

- ☐ can be applied to every classification algorithm
- ☐ changes ridge regression so we solve a $d \times d$ linear system instead of an $n \times n$ system, given n sample points with d features
- ☐ is commonly used for dimensionality reduction
- ■ exploits the fact that in many learning algorithms, the weights can be written as a linear combination of input points

(8) [3 pts] Suppose we train a hard-margin linear SVM on $n > 100$ data points in $\mathbb{R}^2$, yielding a hyperplane with exactly 2 support vectors. If we add one more data point and retrain the classifier, what is the maximum possible number of support vectors for the new hyperplane (assuming the $n+1$ points are linearly separable)?

- ☐ 2
- ☐ n
- ☐ 3
- [✓ $n+1$]

(9) [3 pts] In latent semantic indexing, we compute a low-rank approximation to a term-document matrix. Which of the following motivate the low-rank reconstruction?

- [✓ Finding documents that are related to each other, e.g. of a similar genre]
- ☐ The low-rank approximation provides a lossless method for compressing an input matrix
- [✓ In many applications, some principal components encode noise rather than meaningful structure]
- ☐ Low-rank approximation enables discovery of nonlinear relations

(10) [3 pts] Which of the following are true about subset selection?

- ☐ Subset selection can substantially decrease the bias of support vector machines
- [✓ Subset selection can reduce overfitting]
- ☐ Ridge regression frequently eliminates some of the features
- [✓ Finding the true best subset takes exponential time]

(11) [3 pts] In neural networks, nonlinear activation functions such as sigmoid, tanh, and ReLU

- ☐ speed up the gradient calculation in backpropagation, as compared to linear units
- [✓ help to learn nonlinear decision boundaries]
- ☐ are applied only to the output units
- ☐ always output values between 0 and 1

(12) [3 pts] Suppose we are given data comprising points of several different classes. Each class has a different probability distribution from which the sample points are drawn. We do not have the class labels. We use k-means clustering to try to guess the classes. Which of the following circumstances would undermine its effectiveness?

- ☐ Some of the classes are not normally distributed
- ☐ The variance of each distribution is small in all directions
- [✓ Each class has the same mean]
- [✓ You choose $k = n$, the number of sample points]

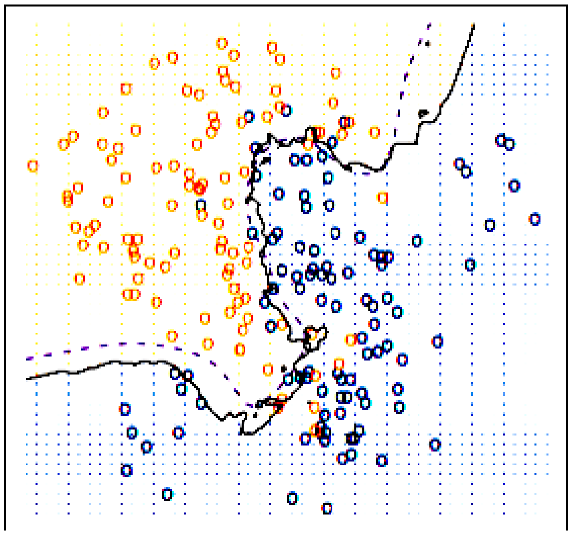
(13) [3 pts] Which of the following are true of spectral graph partitioning methods?

- ☐ They find the cut with minimum weight
- ☐ They minimize a quadratic function subject to one constraint: the partition must be balanced
- [✓ They use one or more eigenvectors of the Laplacian matrix]
- ☐ The Normalized Cut was invented at Stanford

(14) [3 pts] Which of the following can help to reduce overfitting in an SVM classifier?

- [✓ Use of slack variables]
- ☐ High-degree polynomial features
- ☐ Normalizing the data
- ☐ Setting a very low learning rate

(15) [3 pts] Which value of k in the k-nearest neighbors algorithm generates the solid decision boundary depicted here? There are only 2 classes. (Ignore the dashed line, which is the Bayes decision boundary.)



☐ $k = 1$ ☐ $k = 2$

🟥 $k = 10$ ☐ $k = 100$

(16) [3 pts] Consider one layer of weights (edges) in a convolutional neural network (CNN) for grayscale images, connecting one layer of units to the next layer of units. Which type of layer has the fewest parameters to be learned during training? (Select one.)

☐ A convolutional layer with 10 3×3 filters ☐ A convolutional layer with 8 5×5 filters

🟥 A max-pooling layer that reduces a 10×10 image to 5×5 ☐ A fully-connected layer from 20 hidden units to 4 output units

(17) [3 pts] In the kernelized perceptron algorithm with learning rate $\epsilon = 1$, the coefficient a_i corresponding to a training example x_i represents the weight for $K(x_i, x)$. Suppose we have a two-class classification problem with $y_i \in \{1, -1\}$. If $y_i = 1$, which of the following can be true for a_i?

☐ $a_i = -1$ 🟥 $a_i = 1$

🟥 $a_i = 0$ 🟥 $a_i = 5$

(18) [3 pts] Suppose you want to split a graph G into two subgraphs. Let L be G's Laplacian matrix. Which of the following could help you find a good split?

☐ The eigenvector corresponding to the second-largest eigenvalue of L ☐ The left singular vector corresponding to the second-largest singular value of L

🟥 The eigenvector corresponding to the second-smallest eigenvalue of L 🟥 The left singular vector corresponding to the second-smallest singular value of L

(19) [3 pts] Which of the following are properties that a kernel matrix always has?

☐ Invertible ☐ All the entries are positive

☐ At least one negative eigenvalue 🟥 Symmetric

(20) [3 pts] How does the bias-variance decomposition of a ridge regression estimator compare with that of ordinary least squares regression? (Select one.)

- ☐ Ridge has larger bias, larger variance
- ☐ Ridge has smaller bias, larger variance
- [✓ Ridge has larger bias, smaller variance]
- ☐ Ridge has smaller bias, smaller variance

(21) [3 pts] Both PCA and Lasso can be used for feature selection. Which of the following statements are true?

- [✓ Lasso selects a subset (not necessarily a strict subset) of the original features]
- ☐ PCA and Lasso both allow you to specify how many features are chosen
- [✓ PCA produces features that are linear combinations of the original features]
- ☐ PCA and Lasso are the same if you use the kernel trick

(22) [3 pts] Which of the following are true about forward subset selection?

- ☐ $O(2^d)$ models must be trained during the algorithm, where d is the number of features
- ☐ It finds the subset of features that give the lowest test error
- [✓ It greedily adds the feature that most improves cross-validation accuracy]
- [✓ Forward selection is faster than backward selection if few features are relevant to prediction]

(23) [3 pts] You've just finished training a random forest for spam classification, and it is getting abnormally bad performance on your validation set, but good performance on your training set. Your implementation has no bugs. What could be causing the problem?

- [✓ Your decision trees are too deep]
- [✓ You have too few trees in your ensemble]
- [✓ You are randomly sampling too many features when you choose a split]
- [✓ Your bagging implementation is randomly sampling sample points *without* replacement]

(24) [3 pts] Consider training a decision tree given a design matrix $X = \begin{bmatrix} 6 & 3 \\ 2 & 7 \\ 9 & 6 \\ 4 & 2 \end{bmatrix}$ and labels $y = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$. Let f_1 denote feature 1, corresponding to the first column of X, and let f_2 denote feature 2, corresponding to the second column. Which of the following splits at the root node gives the highest information gain? (Select one.)

- ☐ $f_1 > 2$
- ☐ $f_2 > 3$
- [✓ $f_1 > 4$]
- ☐ $f_2 > 6$

(25) [3 pts] In terms of the bias-variance decomposition, a 1-nearest neighbor classifier has ______________ than a 3-nearest neighbor classifier.

- [✓ higher variance]
- ☐ higher bias
- ☐ lower variance
- [✓ lower bias]

(26) [3 pts] Which of the following are true about bagging?

- [✓ In bagging, we choose random subsamples of the input points with replacement]
- ☐ The main purpose of bagging is to decrease the bias of learning algorithms.
- ☐ Bagging is ineffective with logistic regression, because all of the learners learn exactly the same decision boundary
- [✓ If we use decision trees that have one sample point per leaf, bagging never gives lower training error than one ordinary decision tree]

(27) [3 pts] An advantage of searching for an approximate nearest neighbor, rather than the exact nearest neighbor, is that

- ☐ it sometimes makes exhaustive search much faster
- ☐ the nearest neighbor classifier is sometimes much more accurate
- [✓ it sometimes makes searching in a k-d tree much faster]
- ☐ you find all the points within a distance of $(1+\epsilon)r$ from the query point, where r is the distance from the query point to its nearest neighbor

(28) [3 pts] In the derivation of the spectral graph partitioning algorithm, we *relax* a combinatorial optimization problem to a continuous optimization problem. This relaxation has the following effects.

- [✓ The combinatorial problem requires an exact bisection of the graph, but the continuous algorithm can produce (after rounding) partitions that aren't perfectly balanced]
- ☐ The combinatorial problem requires finding eigenvectors, whereas the continuous problem requires only matrix multiplication
- ☐ The combinatorial problem cannot be modified to accommodate vertices that have different masses, whereas the continuous problem can
- [✓ The combinatorial problem is NP-hard, but the continuous problem can be solved in polynomial time]

(29) [3 pts] The firing rate of a neuron

- ☐ determines how strongly the dendrites of the neuron stimulate axons of neighboring neurons
- [✓ is more analogous to the output of a unit in a neural net than the output voltage of the neuron]
- ☐ only changes very slowly, taking a period of several seconds to make large adjustments
- ☐ can sometimes exceed 30,000 action potentials per second

(30) [3 pts] In algorithms that use the kernel trick, the Gaussian kernel

- [✓ gives a regression function or predictor function that is a linear combination of Gaussians centered at the sample points]
- ☐ is equivalent to lifting the d-dimensional sample points to points in a space whose dimension is exponential in d
- [✓ is less prone to oscillating than polynomials, assuming the variance of the Gaussians is large]
- ☐ has good properties in theory but is rarely used in practice

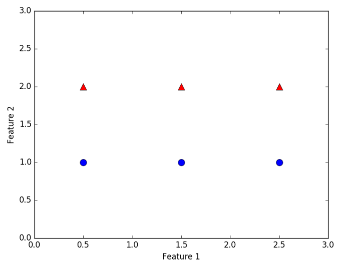
(31) **3 bonus points!** The following Berkeley professors were cited in this semester's lectures (possibly self-cited) for specific research contributions they made to machine learning.

- ☐ David Culler
- [✓ Michael Jordan]
- [✓ Jitendra Malik]
- [✓ Leo Breiman]
- [✓ Anca Dragan]
- ☐ Jonathan Shewchuk

Q2. [8 pts] Feature Selection

A newly employed former CS 189/289A student trains the latest Deep Learning classifier and obtains state-of-the-art accuracy. However, the classifier uses too many features! The boss is overwhelmed and asks for a model with fewer features.

Let's try to identify the most important features. Start with a simple dataset in $\mathbb{R}^2$.

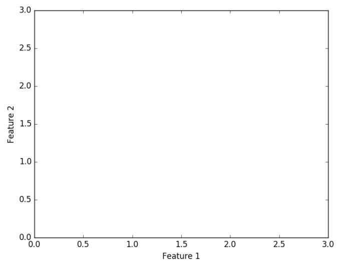


(1) [4 pts] Describe the training error of a Bayes optimal classifier that can see only the first feature of the data. Describe the training error of a Bayes optimal classifier that can see only the second feature.

The first feature yields a training error of 50% (like random guessing). The second feature offers a training error of zero.

(2) [4 pts] Based on this toy example, the student decides to fit a classifier on each feature individually, then rank the features by their classifier's accuracy, take the best k features, and train a new classifier on those k features. We call this approach *variable ranking*. Unfortunately, the classifier trained on the best k features obtains horrible accuracy, unless k is very close to d, the original number of features!

Construct a toy dataset in $\mathbb{R}^2$ for which variable ranking fails. In other words, a dataset where a variable is useless by itself, but potentially useful alongside others. Use + for data points in Class 1, and O for data points in Class 2.



An XOR Dataset is unpredictable with either feature. (This extends to n-dimensions, with the n-bit parity string.)

Q3. [10 pts] Gradient Descent for k-means Clustering

Recall the loss function for k-means clustering with k clusters, sample points $x_1, ..., x_n$, and centers $\mu_1, ..., \mu_k$:

$$L = \sum_{j=1}^{k} \sum_{x_i \in S_j} \|x_i - \mu_j\|^2,$$

where S_j refers to the set of data points that are closer to μ_j than to any other cluster mean.

(1) [4 pts] Instead of updating μ_j by computing the mean, let's minimize L with **batch** gradient descent while holding the sets S_j fixed. Derive the update formula for μ_1 with learning rate (step size) ϵ.

$$\begin{aligned}\frac{\partial L}{\partial \mu_1} &= \frac{\partial}{\partial \mu_1} \sum_{x_i \in S_1} (x_i - \mu_1)^\top (x_i - \mu_1) \\ &= \sum_{x_i \in S_1} 2(\mu_1 - x_i).\end{aligned}$$

Therefore the update formula is

$$\mu_1 \leftarrow \mu_1 + \epsilon \sum_{x_i \in S_1} (x_i - \mu_1).$$

(Note: writing 2ϵ instead of ϵ is fine.)

(2) [2 pts] Derive the update formula for μ_1 with **stochastic** gradient descent on a single sample point x_i. Use learning rate ϵ.

$\mu_1 \leftarrow \mu_1 + \epsilon(x_i - \mu_1)$ if $x_i \in S_1$, otherwise no change.

(3) [4 pts] In this part, we will connect the batch gradient descent update equation with the standard k-means algorithm. Recall that in the update step of the standard algorithm, we assign each cluster center to be the mean (centroid) of the data points closest to that center. It turns out that a particular choice of the learning rate ϵ (which may be different for each cluster) makes the two algorithms (batch gradient descent and the standard k-means algorithm) have identical update steps. Let's focus on the update for the first cluster, with center μ_1. Calculate the value of ϵ so that both algorithms perform the same update for μ_1. (If you do it right, the answer should be very simple.)

In the standard algorithm, we assign $\mu_1 \leftarrow \sum_{x_i \in S_1} \frac{1}{|S_1|} x_i$.

Comparing to the answer in (1), we set $\sum_{x_i \in S_1} \frac{1}{|S_1|} x_i = \mu_1 + \epsilon \sum_{x_i \in S_1} (x_i - \mu_1)$ and solve for ϵ.

$$\begin{aligned}\sum_{x_i \in S_1} \frac{1}{|S_1|} x_i - \sum_{x_i \in S_1} \frac{1}{|S_1|} \mu_1 &= \epsilon \sum_{x_i \in S_1} (x_i - \mu_1) \\ \sum_{x_i \in S_1} \frac{1}{|S_1|} (x_i - \mu_1) &= \epsilon \sum_{x_i \in S_1} (x_i - \mu_1).\end{aligned}$$

Thus $\epsilon = \frac{1}{|S_1|}$.

(Note: answers that differ by a constant factor are fine if consistent with answer for (1).)

Q4. [10 pts] Kernels

(1) [2 pts] What is the primary motivation for using the kernel trick in machine learning algorithms?

If we want to map sample points to a very high-dimensional feature space, the kernel trick can save us from having to compute those features explicitly, thereby saving a lot of time.

(Alternative solution: the kernel trick enables the use of infinite-dimensional feature spaces.)

(2) [4 pts] Prove that for every design matrix $X \in \mathbb{R}^{n \times d}$, the corresponding kernel matrix is positive semidefinite.

For every vector $\mathbf{z} \in \mathbb{R}^n$,

$$\mathbf{z}^\top K \mathbf{z} = \mathbf{z}^\top X X^\top \mathbf{z} = |X^\top \mathbf{z}|^2,$$

which is clearly nonnegative.

(3) [2 pts] Suppose that a regression algorithm contains the following line of code.

$$\mathbf{w} \leftarrow \mathbf{w} + X^\top M X X^\top \mathbf{u}$$

Here, $X \in \mathbb{R}^{n \times d}$ is the design matrix, $\mathbf{w} \in \mathbb{R}^d$ is the weight vector, $M \in \mathbb{R}^{n \times n}$ is a matrix unrelated to X, and $\mathbf{u} \in \mathbb{R}^n$ is a vector unrelated to X. We want to derive a dual version of the algorithm in which we express the weights $\mathbf{w}$ as a linear combination of samples X_i (rows of X) and a dual weight vector $\mathbf{a}$ contains the coefficients of that linear combination. Rewrite the line of code in its dual form so that it updates $\mathbf{a}$ correctly (and so that $\mathbf{w}$ does not appear).

$$\mathbf{a} \leftarrow \mathbf{a} + M X X^\top \mathbf{u}$$

(4) [2 pts] Can this line of code for updating $\mathbf{a}$ be kernelized? If so, show how. If not, explain why.

Yes:

$$\mathbf{a} \leftarrow \mathbf{a} + M K \mathbf{u}$$

Q5. [12 pts] Let's PCA

You are given a design matrix $X = \begin{bmatrix} 6 & -4 \\ -3 & 5 \\ -2 & 6 \\ 7 & -3 \end{bmatrix}$. Let's use PCA to reduce the dimension from 2 to 1.

(1) [6 pts] Compute the covariance matrix for the sample points. (Warning: Observe that X is not centered.) Then compute the **unit** eigenvectors, and the corresponding eigenvalues, of the covariance matrix. *Hint:* If you graph the points, you can probably guess the eigenvectors (then verify that they really are eigenvectors).

The covariance matrix is $X^\top X = \begin{bmatrix} 82 & -80 \\ -80 & 82 \end{bmatrix}$.

Its unit eigenvectors are $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ with eigenvalue 2 and $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$ with eigenvalue 162. (Note: either eigenvector can be replaced with its negation.)

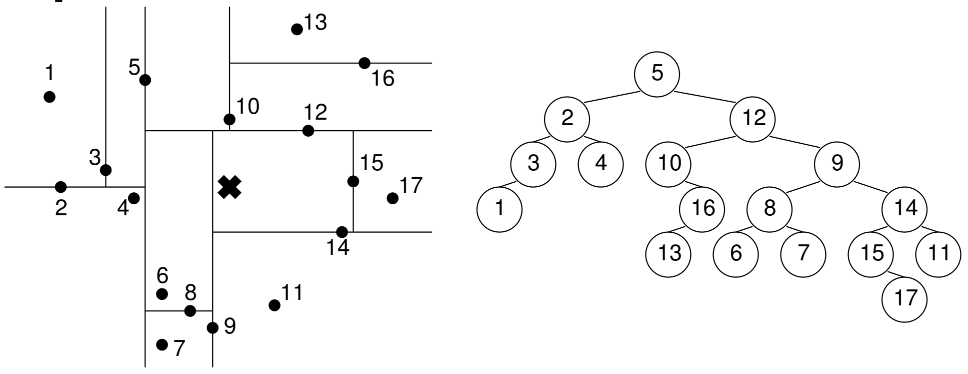
(2) [3 pts] Suppose we use PCA to project the sample points onto a one-dimensional space. What one-dimensional subspace are we projecting onto? For each of the four sample points in X (not the centered version of X!), write the coordinate (in principal coordinate space, not in $\mathbb{R}^2$) that the point is projected to.

We are projecting onto the subspace spanned by $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$. (Equivalently, onto the space spanned by $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$. Equivalently, onto the line $x + y = 0$.) The projections are $(6, -4) \rightarrow \frac{10}{\sqrt{2}}$, $(-3, 5) \rightarrow -\frac{8}{\sqrt{2}}$, $(-2, 6) \rightarrow -\frac{8}{\sqrt{2}}$, $(7, -3) \rightarrow \frac{10}{\sqrt{2}}$.

(3) [3 pts] Given a design matrix X that is taller than it is wide, prove that every right singular vector of X with singular value σ is an eigenvector of the covariance matrix with eigenvalue σ^2.

If v is a right singular vector of X, then there is a singular value decomposition $X = UDV^\top$ such that v is a column of V. Here each of U and V has orthonormal columns, V is square, and D is square and diagonal. The covariance matrix is $X^\top X = VDU^\top UDV^\top = VD^2V^\top$. This is an eigendecomposition of $X^\top X$, so each singular vector in V with singular value σ is an eigenvector of $X^\top X$ with eigenvalue σ^2.

Q6. [10 pts] Trees



(1) [5 pts] Above, we have two depictions of the same k-d tree, which we have built to solve nearest neighbor queries. Each node of the tree at right represents a rectangular box at left, and also stores one of the sample points that lie inside that box. (The root node represents the whole plane $\mathbb{R}^2$.) If a treenode stores sample point i, then the line passing through point i (in the diagram at left) determines which boxes the child treenodes represent.

Simulate running an exact 1-nearest neighbor query, where the bold X is the query point. Recall that the query algorithm visits the treenodes in a smart order, and keeps track of the nearest point it has seen so far.

- Write down the numbers of all the sample points that serve as the "nearest point seen so far" sometime while the query algorithm is running, in the order they are encountered.
- Circle all the subtrees in the k-d tree at upper right that are never visited during this query. (This is why k-d tree search is usually faster than exhaustive search.)

Nearest point seen so far: first 5, then 12, then 10.

The unvisited subtrees are rooted at 2, 13, 7, and 17.

(2) [5 pts] We are building a decision tree for a 2-class classification problem. We have n training points, each having d real-valued features. At each node of the tree, we try every possible univariate split (i.e. for each feature, we try every possible splitting value for that feature) and choose the split that maximizes the information gain.

Explain why it is possible to build the tree in $O(ndh)$ time, where h is the depth of the tree's deepest node. Your explanation should include an analysis of the time to choose one node's split. Assume that we can radix sort real numbers in linear time.

Consider choosing the split at a node whose box contains n' sample points. For each of the d features, we can sort the sample points in $O(n'd)$ time. Then we can compute the entropy for the first split (separating the first sample in the sorted list from the others) in $O(n')$ time, then we can walk through the list and update the entropy for each successive split in $O(1)$ time, summing to a total of $O(n')$ time for each of the d features. So it takes $O(n'd)$ time overall to choose a split.

Each sample point participates in at most h treenodes, so each sample point contributes at most dh to the running time, for a total running time of at most $O(ndh)$.

Q7. [10 pts] Self-Driving Cars and Backpropagation

You want to train a neural network to drive a car. Your training data consists of grayscale 64×64 pixel images. The training labels include the human driver's steering wheel angle in degrees and the human driver's speed in miles per hour. Your neural network consists of an input layer with $64 \times 64 = 4{,}096$ units, a hidden layer with 2,048 units, and an output layer with 2 units (one for steering angle, one for speed). You use the ReLU activation function for the hidden units and no activation function for the outputs (or inputs).

(1) [2 pts] Calculate the number of parameters (weights) in this network. You can leave your answer as an expression. Be sure to account for the bias terms.

$$4097 \times 2048 + 2049 \times 2$$

(2) [3 pts] You train your network with the cost function $J = \frac{1}{2}|\mathbf{y} - \mathbf{z}|^2$. Use the following notation.

- $\mathbf{x}$ is a training image (input) vector with a 1 component appended to the end, $\mathbf{y}$ is a training label (input) vector, and $\mathbf{z}$ is the output vector. All vectors are column vectors.
- $r(\gamma) = \max\{0, \gamma\}$ is the ReLU activation function, $r'(\gamma)$ is its derivative (1 if $\gamma > 0$, 0 otherwise), and $r(\mathbf{v})$ is $r(\cdot)$ applied component-wise to a vector.
- $\mathbf{g}$ is the vector of hidden unit values before the ReLU activation functions are applied, and $\mathbf{h} = r(\mathbf{g})$ is the vector of hidden unit values after they are applied (but we append a 1 component to the end of $\mathbf{h}$).
- V is the weight matrix mapping the input layer to the hidden layer; $\mathbf{g} = V\mathbf{x}$.
- W is the weight matrix mapping the hidden layer to the output layer; $\mathbf{z} = W\mathbf{h}$.

Derive $\partial J/\partial W_{ij}$.

$$\begin{aligned}\frac{\partial J}{\partial W_{ij}} &= (\mathbf{z} - \mathbf{y})^\top \frac{\partial \mathbf{z}}{\partial W_{ij}} \\ &= (\mathbf{z}_i - \mathbf{y}_i)\mathbf{h}_j\end{aligned}$$

(3) [1 pt] Write $\partial J/\partial W$ as an outer product of two vectors. $\partial J/\partial W$ is a matrix with the same dimensions as W; it's just like a gradient, except that W and $\partial J/\partial W$ are matrices rather than vectors.

$$\frac{\partial J}{\partial W} = (\mathbf{z} - \mathbf{y})\mathbf{h}^\top$$

(4) [4 pts] Derive $\partial J/\partial V_{ij}$.

$$\begin{aligned}\frac{\partial J}{\partial V_{ij}} &= (\mathbf{z} - \mathbf{y})^\top \frac{\partial \mathbf{z}}{\partial V_{ij}} \\ &= (\mathbf{z} - \mathbf{y})^\top W \frac{\partial \mathbf{h}}{\partial V_{ij}} \\ &= (\mathbf{z} - \mathbf{y})^\top W \, [0, \ldots, r'(\mathbf{g}_i)\, \mathbf{x}_j, \ldots, 0]^\top \\ &= ((\mathbf{z} - \mathbf{y})^\top W)_i \, r'(\mathbf{g}_i)\, \mathbf{x}_j.\end{aligned}$$