CS 189 Fall 2017 — Introduction to Machine Learning

EXAMFINALREVIEW

1 Multiple Choice

For the following multiple choice questions, select all that apply.

(a) What strategies can help reduce overfitting in decision trees?

◯ Pruning

◯ Enforce a minimum number of samples in leaf nodes

◯ Make sure each leaf node is one pure class

◯ Enforce a maximum depth for the tree

Solution: (a), (c), (d)

(b) Neural Networks...

◯ optimize a convex cost function

◯ always output values between 0 and 1

◯ can be used for regression as well as classification

◯ can be used in an ensemble

Solution: (b),(d)

(c) Which of the following methods can achieve zero training error on *any* linearly separable dataset?

◯ Decision tree

◯ 15-nearest neighbors

◯ Hard-margin SVM

◯ Soft-margin SVM

Solution: (a), (b), (d)

(d) Which of the following can help to reduce overfitting in an SVM classifier?

◯ Use of slack variables

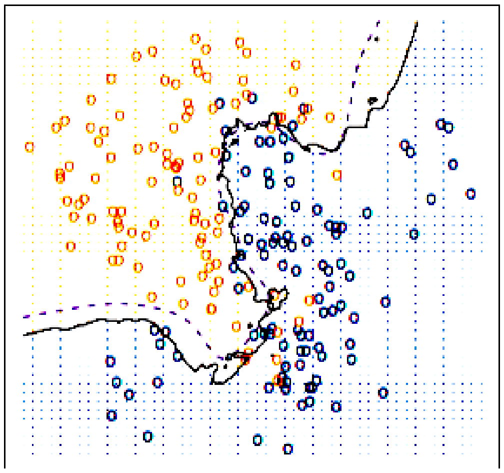
◯ High-degree polynomial features

◯ Normalizing the data

◯ Setting a very low learning rate

Solution: (a)

(e) 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.)



◯ $k = 1$ ◯ $k = 2$

◯ $k = 10$ ◯ $k = 100$

Solution: (b)

(f) Which of the following are properties that a kernel matrix always has?

◯ Invertible ◯ All the entries are positive

◯ At least one negative eigenvalue ◯ Symmetric

Solution: (d)

(g) 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 are randomly sampling too many features when you choose a split

◯ You have too few trees in your ensemble

◯ Your bagging implementation is randomly sampling sample points *without* replacement

Solution: (a), (b),(c),(d)

(h) In terms of the bias-variance decomposition, a 1-nearest neighbor classifier has ____________ than a 3-nearest neighbor classifier.

◯ higher variance

◯ lower variance

◯ higher bias

◯ lower bias

Solution: (a), (d)

(i) Which of the following are true about bagging?

◯ In bagging, we choose random subsamples of the input points with replacement

◯ In bagging, we assign more weight to trees with higher accuracy

◯ The main purpose of bagging is to decrease the bias of learning algorithms.

◯ If we use decision trees that have one sample point per leaf, bagging never gives lower training error than one ordinary decision tree

Solution: (a), (d)

(j) Why is PCA sometimes used as a preprocessing step before regression?

◯ To reduce overfitting by removing poorly predictive dimensions.

◯ To expose information missing from the input data.

◯ To make computation faster by reducing the dimensionality of the data.

◯ For inference and scientific discovery, we prefer features that are not axis-aligned.

Solution: (a), (c)

(k) For which of the following does normalizing your input features influence the predictions?

◯ decision tree (with usual splitting method)

◯ Lasso

◯ neural network

◯ soft-margin support vector machine

Solution: (b), (c), (d)

(l) Why would we use a random forest instead of a decision tree?

◯ For lower training error.

◯ To reduce the variance of the model.

◯ To better approximate posterior probabilities.

◯ For a model that is easier for a human to interpret.

Solution: (b), (c)

(m) A low-rank approximation of a matrix can be useful for

◯ removing noise.

◯ filling in unknown values.

◯ discovering latent categories in the data.

◯ matrix compression.

Solution: (a), (b), (c), (d)

(n) Which of the following statements is true about the standard k-means clustering algorithm?

◯ The random partition initialization method usually outperforms the Forgy method.

◯ It is computationally infeasible to find the optimal clustering of $n = 15$ points in $k = 3$ clusters.

◯ After a sufficiently large number of iterations, the clusters will stop changing.

◯ You can use the metric $d(x,y) = \frac{x \cdot y}{|x| \cdot |y|}$.

Solution: (b)

2 Maximum Likelihood Estimation for Reliability Testing

Suppose we are reliability testing n units taken randomly from a population of identical appliances. We want to estimate the mean failure time of the population. We assume the failure times come from an exponential distribution with parameter $\lambda > 0$, whose probability density function is $f(x) = \lambda e^{-\lambda x}$ (on the domain $x \geq 0$) and whose cumulative distribution function is $F(x) = \int_0^x f(x)\,\mathrm{d}x = 1 - e^{-\lambda x}$.

(a) In an ideal (but impractical) scenario, we run the units until they all fail. The failure times are $t_1, t_2, \ldots, t_n$.

Formulate the likelihood function $\mathscr{L}(\lambda; t_1, \ldots, t_n)$ for our data. Then find the maximum likelihood estimate $\hat{\lambda}$ for the distribution's parameter.

Solution:

$$\mathscr{L}(\lambda; t_1, \ldots, t_n) = \prod_{i=1}^{n} f(t_i) = \prod_{i=1}^{n} \lambda e^{-\lambda t_i} = \lambda^n e^{-\lambda \sum_{i=1}^{n} t_i}$$

$$\ln \mathscr{L}(\lambda) = n \ln \lambda - \lambda \sum_{i=1}^{n} t_i$$

$$\frac{\partial}{\partial \lambda} \ln \mathscr{L}(\lambda) = \frac{n}{\lambda} - \sum_{i=1}^{n} t_i = 0$$

$$\hat{\lambda} = \frac{n}{\sum_{i=1}^{n} t_i}$$

(b) In a more realistic scenario, we run the units for a fixed time T. We observe r unit failures, where $0 \le r \le n$, and there are $n-r$ units that survive the entire time T without failing. The failure times are $t_1, t_2, \ldots, t_r$.

Formulate the likelihood function $\mathscr{L}(\lambda; n, r, t_1, \ldots, t_r)$ for our data. Then find the maximum likelihood estimate $\hat{\lambda}$ for the distribution's parameter.

Hint 1: What is the probability that a unit will not fail during time T? *Hint 2:* It is okay to define $\mathscr{L}(\lambda)$ in a way that includes contributions (densities and probability masses) that are not commensurate with each other. Then the constant of proportionality of $\mathscr{L}(\lambda)$ is meaningless, but that constant is irrelevant for finding the best-fit parameter $\hat{\lambda}$. *Hint 3:* If you're confused, for part marks write down the likelihood that r units fail and $n-r$ units survive; then try the full problem. *Hint 4:* If you do it right, $\hat{\lambda}$ will be the number of observed failures divided by the sum of unit test times.

Solution:

$$\begin{aligned}
\mathscr{L}(\lambda; n, r, t_1, \ldots, t_r) &\propto \left(\prod_{i=1}^{r} f(t_i)\right)(1-F(T))^{n-r} \\
&= \left(\prod_{i=1}^{r} \lambda e^{-\lambda t_i}\right)\left(e^{-\lambda T}\right)^{n-r} \\
&= \lambda^r e^{-\lambda \sum_{i=1}^{r} t_i} e^{-\lambda (n-r) T} \\
\ln \mathscr{L}(\lambda) &= r \ln \lambda - \lambda \sum_{i=1}^{r} t_i - \lambda (n-r) T + \text{constant} \\
\frac{\partial}{\partial \lambda} \ln \mathscr{L}(\lambda) &= \frac{r}{\lambda} - \sum_{i=1}^{r} t_i - (n-r) T = 0 \\
\hat{\lambda} &= \frac{r}{\sum_{i=1}^{r} t_i + (n-r) T}
\end{aligned}$$

3 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.

(a) 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).

Solution: 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.)

(b) 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.

Solution: 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 $(6,-4) \to \frac{10}{\sqrt{2}}$, $(-3,5) \to -\frac{8}{\sqrt{2}}$, $(-2,6) \to -\frac{8}{\sqrt{2}}$, $(7,-3) \to \frac{10}{\sqrt{2}}$.

(c) 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.

Solution: 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.

4 Gradient Descent for k-means Clustering

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

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

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

(a) 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.

Solution:

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

Therefore the update formula is

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

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

(b) Derive the update formula for μ_1 with **stochastic** gradient descent on a single sample point x_i. Use learning rate ε. **Solution:** $\mu_1 \leftarrow \mu_1 + \varepsilon(x_i - \mu_1)$ if $x_i \in S_1$, otherwise no change.

(c) 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.)

Solution: 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 $\sum_{x_i \in S_1} \frac{1}{|S_1|} x_i = \mu_1 + \varepsilon \sum_{x_i \in S_1} (x_i - \mu_1)$ and solve for ε.

$$\sum_{x_i \in S_1} \frac{1}{|S_1|} x_i - \sum_{x_i \in S_1} \frac{1}{|S_1|} \mu_1 = \varepsilon \sum_{x_i \in S_1} (x_i - \mu_1) \sum_{x_i \in S_1} \frac{1}{|S_1|} (x_i - \mu_1) = \varepsilon \sum_{x_i \in S_1} (x_i - \mu_1).$$

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

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

5 Kernels

(a) What is the primary motivation for using the kernel trick in machine learning algorithms? **Solution:** 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.)

(b) Prove that for every design matrix $X \in R^{n \times d}$, the corresponding kernel matrix is positive semidefinite.

Solution: 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.

(c) 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).

Solution:

$$\mathbf{a} \leftarrow \mathbf{a} + MXX^\top \mathbf{u}$$

(d) Can this line of code for updating $\mathbf{a}$ be kernelized? If so, show how. If not, explain why.
Solution: Yes:

$$\mathbf{a} \leftarrow \mathbf{a} + MK\mathbf{u}$$

6 Decision Trees

Consider the design matrix

$$\begin{bmatrix} 4 & 6 & 9 & 1 & 7 & 5 \\ 1 & 6 & 5 & 2 & 3 & 4 \end{bmatrix}^\top$$

representing 6 sample points, each with two features f_1 and f_2.

The labels for the data are

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}^\top$$

In this question, we build a decision tree of depth 2 by hand to classify the data.

(a) What is the entropy at the root of the tree?

Solution:

$$-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

(b) What is the rule for the first split? Write your answer in a form like $f_1 \geq 4$ or $f_2 \geq 3$. Hint: you should be able to eyeball the best split without calculating the entropies.

Solution: If we sort by f_1, the features and the corresponding labels are

$$\begin{bmatrix} 1 & 4 & 5 & 6 & 7 & 9 \\ 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

If we sort by f_2, we have

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

The best split is $f_1 \geq 7$.

(c) For each of the two treenodes after the first split, what is the rule for the second split?

Solution: For the treenode with labels $(1,1)$, there's no need to split again.

For the treenode with labels $(0,1,0,0)$, if we sort by f_1, we have

$$\begin{bmatrix} 1 & 4 & 5 & 6 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

If we sort using f_2, we get

$$\begin{bmatrix} 1 & 2 & 4 & 6 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

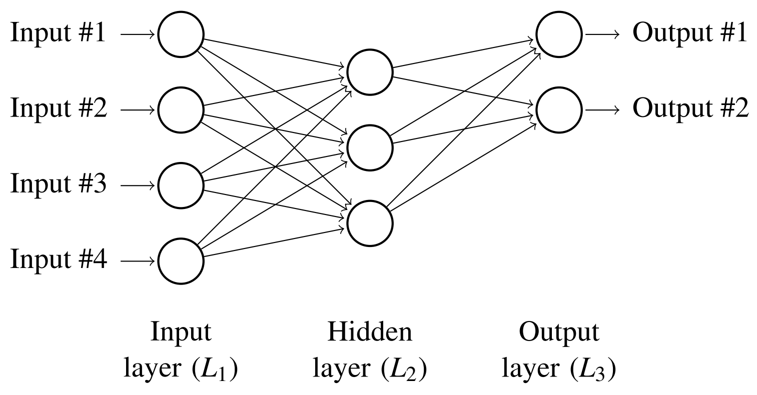
We easily see we should choose $f_2 \geq 2$.

(d) Let's return to the root of the tree, and suppose we're incompetent tree builders. Is there a (not trivial) split at the root that would have given us an information gain of zero? Explain your answer.

Solution: Yes. The rules $f_1 \geq 5$, $f_2 \geq 3$, or $f_2 \geq 5$ would all fail to reduce the weighted average entropy below 1.

7 Neural Networks

Consider a three layer fully-connected network with n_1, n_2, n_3 neurons in three layers respectively. Inputs are fed into the first layer. The loss is mean squared error E, and the non-linearity is a sigmoid function. Let the label vector be t of size n_3. Let each layer output vector be y_i and input vector be z_i, both of size n_i. Let the weight between layer i and layer $i+1$ be W_{ii+1}. The j-th element in y_i is defined by y_i^j, same for z_i^j. The weight connecting k-th and l-th neuron in $i, i+1$ layers is defined by W_{ii+1}^{kl} (You don't need to consider bias in this problem).



Here is a summary of our notation:

- σ denotes the activation function for L_2 and L_3, $\sigma(x) = \frac{1}{1+e^{-x}}$. There is no activation applied to the input layer.
- $z_i^{(j)} = \sum_{k=1}^{P} W_{i-1i}^{kj} x_{i-1}^{(k)}$

- $y_i^{(j)} = \sigma\left(\sum_{k=1}^{P} W_{i-1i}^{kj} x_{i-1}^{(k)}\right)$

Now solve the following problems.

(a) Find $\frac{\partial E}{\partial z_3^j}$ in terms of y_3^j.

Solution:
$-2y_3^j(1-y_3^j)(t^j - y_3^j)$

(b) Find $\frac{\partial E}{\partial y_2^k}$ in terms of elements in W_{23} and $\frac{\partial E}{\partial z_3^j}$.

Solution:
$\sum_{j=1}^{n_3} W_{23}^{kj} \frac{\partial E}{\partial z_3^j}$

(c) Find $\frac{\partial E}{\partial W_{23}^{kj}}$ in terms of y_2^k, y_3^j and t^j.

Solution:
$y_2^k \frac{\partial E}{\partial z_3^j} = -2y_3^j(1-y_3^j)(t^j - y_3^j)y_2^k$

(d) If the input to a neuron in max-pooling layer is x and the output is $y = max(x)$, derive $\frac{\partial y}{\partial x_i}$.

Solution:
$\frac{\partial y}{\partial x_i} = 1$ if and only if $x_i = max(x)$, otherwise $\frac{\partial y}{\partial x_i} = 0$.