Name: _____ NetID: _____

S&DS 355 / 555

Introductory Machine Learning

Final Exam (Practice)

Saturday, December 14, 2019

Complete all of the problems. You are allowed one double-sided (8.5×11) sheet of paper with notes. No electronic devices, including calculators. You have 2.5 hours to complete the exam.

Note: This is not a complete practice final. Rather, it is mainly intended to tell you what to expect for the final. The actual exam will have the same structure: Problem 1 is T/F, Problem 2 is short answer, etc. On this practice exam, the only new problems are Problem 2 (Short Answer) and Problem 6 (Implement a predict function). The other questions are taken from Midterm #2.

Sample solutions for these two new problems are provided; solutions to the midterm were already posted to Canvas.

1	2	3	4	5	6	total
20	15	10	10	10	15	80

1. [From midterm] *True or False, Yes or No?* (20 points)

   Indicate the *best* answer to each of the following statements.

   In a class-based bigram language model, the probability of the next word is

$$p(w_n \mid w_{n-1}) = p(w_n \mid c(w_n))\, p(c(w_n) \mid c(w_{n-1}))$$

   where $c(w)$ is the class of word w.

   In an embedding-based bigram language model, the probability of the next word is

$$p(w_n \mid w_{n-1}) = \frac{\exp(\phi(w_n)^T \phi(w_{n-1}))}{\sum_{v \in \text{Vocabulary}} \exp(\phi(v)^T \phi(w_{n-1}))}$$

   where $\phi(w) \in \mathbb{R}^{100}$ is embedding vector for word w.

YES NO (a) As the number of classes decreases, the language model's perplexity on test data will generally decrease.

YES NO (b) For words to have similar embedding vectors, they must co-occur often.

YES NO (c) The embedding vectors can be obtained by bottom up clustering.

YES NO (d) Each word can belong to multiple word classes.

YES NO (e) The embedding vectors are chosen to solve common word analogies.

The following questions concern latent Dirichlet allocation topic model, using the usual terminology where θ_d are the per-document topic proportions (with Dirichlet prior), $Z_{d,n}$ are the per-word topic assignments, and β_k are the topics (with Dirichlet prior).

YES NO (a) A goal of topic modeling is to automatically find the major semantic themes in a corpus of documents

YES NO (b) This topic model is commonly used as a language model to predict the next word.

YES NO (c) The documents in a corpus are generated independently under this model.

YES NO (d) In practice, reordering the words in each document in the corpus gives a different model.

YES NO (e) Conditioned on all of the topic assignments $Z_{d,n}$, the posterior distribution over each topic β_k is a single Dirichlet distribution.

The following questions concern general Bayesian inference.

YES NO (a) Bayesian inference involves three things: a model for assigning a likelihood to data, a prior distribution over the parameters of the model, and a method for computing the conditional probability of those parameters after observing data, using Bayes rule.

YES NO (b) Bayesian inference cannot be used for logistic regression, because it is a discriminative model.

YES NO (c) Bayesian inference is based on the entire posterior distribution of the parameters, rather than estimating a specific value for those parameters.

YES NO (d) The posterior mean is less affected by the choice of prior as the sample size increases.

YES NO (e) Under a Bayesian treatment of models, the parameters θ are fixed numbers.

The following questions concern basic feedforward neural networks.

YES NO (a) If the activation function $\sigma(x) = x$ is the identity, a neural network is equivalent to a linear model.

YES NO (b) The number of parameters in a neural network is proportional to the total number of neurons.

YES NO (c) Training a neural network for classification uses the same objective function on the output as is used for classical logistic regression.

YES NO (d) The hidden neurons can be seen as latent variables in a probabilistic model.

YES NO (e) Training the network automatically determines the optimal network architecture.

2. *Short Answer* (15 points)

For each of the following five concepts, provide a short definition or explanation.

(a) *Bayesian inference*. Describe the three elements of any use of Bayesian inference.

The three elements of Bayesian inference are:

- prior distribution $\pi(\theta)$
- generative model or likelihood function $p(x|\theta)$
- posterior distribution $p(\theta|\mathcal{D}_n)$ where $\mathcal{D}_n = \{x_1, \ldots, x_n\}$ is observed data.

(b) *Logistic regression*. Define the logistic regression model for binary classification and give the loss function for maximum likelihood training.

The logistic regression model for binary classification is defined as

$$P(Y = 1|X) = \frac{1}{1 + e^{-X\beta}}, \quad P(Y = 0|X) = \frac{e^{-X\beta}}{1 + e^{-X\beta}}.$$

The loss function (to minimize) for maximum likelihood training is

$$l(\beta) = \sum_i \left(\log(1 + e^{-x_i^T \beta}) - y_i x_i^T \beta \right)$$

(c) *Bias and variance*. Define the squared bias and variance of an estimator $\widehat{\theta}$ of a parameter θ. Explain the importance of these quantities.

$$\text{Bias}^2(\widehat{\theta}) \;=\; \Big(\mathbb{E}[\widehat{\theta}] - \theta\Big)^2$$

The bias tells us how much our estimate will tend to over-, or under-, estimate the truth on average. This is important because we want our estimator to be flexible enough to adapt to different sources of data, each with a different value of θ.

$$\text{Variance}(\widehat{\theta}) \;=\; \mathbb{E}\left[\Big(\widehat{\theta} - \mathbb{E}[\widehat{\theta}]\Big)^2\right]$$

The variance tells us how much our estimate will vary across independent data sets from the same source. This is important because we want to have an estimator that gives consistent results for different data sets with the same θ.

2. *Short Answer (continued)*

(d) *Principal components analysis*. Give the steps used to carry out PCA on a data set.

- center the data by subtracting off the sample mean, $\widetilde{x}_i = x_i - \bar{x}$ (where each x_i is a vector of length p
- calculate the sample covariance matrix $\widehat{\Sigma} = \widetilde{x}^T \widetilde{x}$
- find the first k eigenvectors $v_1, \ldots, v_k$ of $\widehat{\Sigma}$ (in decreasing order of eigenvalues)
- represent each data point as $x_i = \bar{x} + \sum_{j=1}^{k} (\widetilde{x}_i^T v_j) v_j$

(e) *SGD*. Give the stochastic gradient descent algorithm for linear least squares regression.

initialize $\beta_0 = 0$ and choose step size η

until convergence:

sample k from $\{0, 1, 2, ..., n\}$

update $\beta_{i+1} = \beta_i + \frac{\eta}{\sqrt{i}}(Y_k - X_k^T \beta_i) X_k$

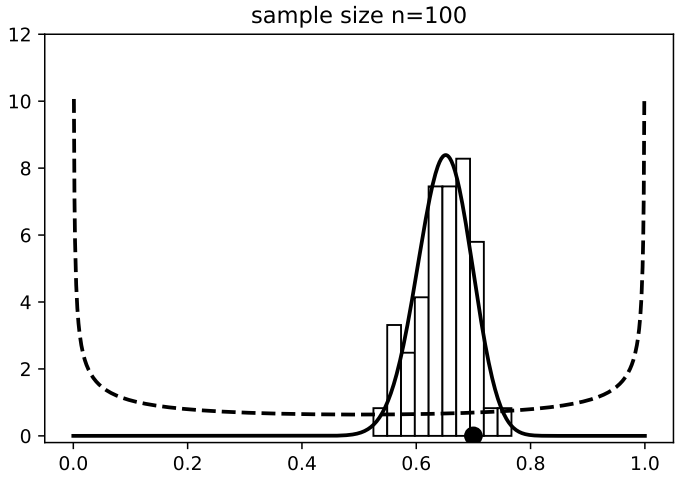
3. [From midterm] *Bayesian inference* (10 points)

Suppose X is a random variable corresponding to a roll of a 6-sided die, with probability $\theta = (\theta_1, \theta_2, \ldots, \theta_6)$ that each of the six faces comes up on any toss. We observe data $D_n = \{x_1, \ldots, x_n\}$ where $x_i \in \{1, 2, 3, 4, 5, 6\}$, and the rolls are independent. Suppose the prior distribution on θ is Dirichlet$(\alpha, \alpha, \alpha, \alpha, \alpha, \alpha)$ where $\alpha > 0$. Let $s_k = \sum_{i=1}^{n} \mathbb{1}(x_i = k)$ be the number or rolls that land on k.

(a) What is the probability (likelihood) $\mathbb{P}((x_1, x_2, x_3, x_4) = (5, 3, 6, 3) \,|\, \theta)$?

(b) What is the posterior distribution of θ given D_n?

(c) In the demo of Bayesian inference in class, we showed "movies" where one frame looked like this:



Briefly describe what each of the following elements of this plot is showing:

(1) The horizontal axis

(2) The black dot at 0.7

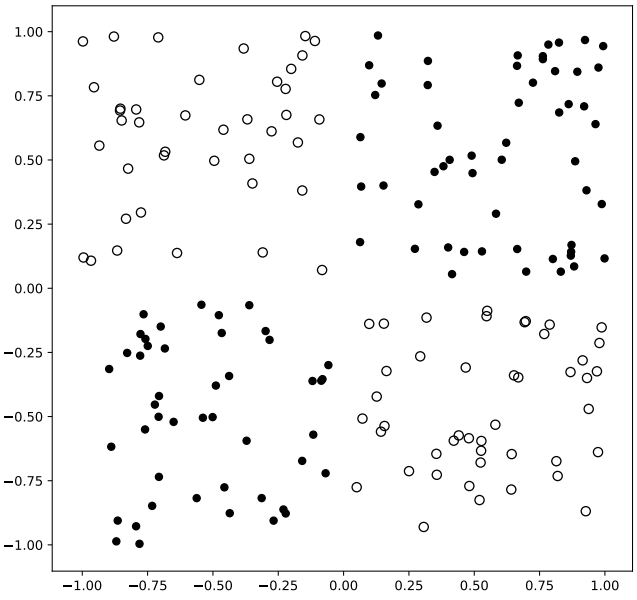
(3) The dashed curve that rises up to around 10 at the endpoints 0.0 and 1.0

(4) The histogram

(5) The solid bell-shaped curve

4. [From midterm] *Neural nets* (10 points)

Consider a function $f(x) = w_2\sigma(w_{11}x_1 + w_{12}x_2 + b_1) + b_2$ where σ is an activation function and $x = (x_1, x_2)$ is a 2-dimensional input. So, $w_{11}, w_{12}, w_2, b_1, b_2$ are numbers. This can be thought of as a neural network with a single hidden layer, consisting of one neuron. It gives a binary classification model by $\mathbb{P}(Y = 1 \,|\, x) = \dfrac{e^{f(x)}}{1 + e^{f(x)}}$. Suppose that we have a data set that looks like this:



The open white points are labeled $Y = 0$ and the solid black points are labeled $Y = 1$. For this problem, you are asked to consider whether or not this simple neural network will give an accurate classifier. *For the purpose of this problem, we will call a classifier "accurate" if the training error rate is less that* 1%.

(a) Can this data set be accurately classified with a linear decision rule? Give a brief justification for your answer.

(b) Suppose the neural network is trained four different ways, using the activation functions ReLU, tanh, identity, and sigmoid. Which of these four neural networks will give accurate classifiers? Briefly explain your answer.

(c) Suppose the neural network is now changed to have input $x_1 x_2$; so the function is now $\widetilde{f}(x) = w_2\sigma(w_1 x_1 x_2 + b_1) + b_2$ with parameters w_1, b_1, w_2, b_2. Four new networks are trained, using each of the activation functions ReLU, tanh, identity, and sigmoid. Which of the resulting four neural networks will give accurate classifiers? Briefly explain your answer.

5. [From midterm] *Code: What does this do?* (10 points)

(a) What is the value of the following Python expression?

```
np.tanh([x for x in np.linspace(-10, 10, 3)])
```

(b) Explain in a couple of sentences what the function `mystery` defined below does:

```
def mystery(X, Y):
  n, d = X.shape
  W = np.random.randn(d,1)
  b = np.random.randn(1,1)

  for i in range(10000):
      f = np.dot(X, W) + b
      phat = np.exp(f)/(1+np.exp(f))
      dloss = phat - Y
      dloss /= num_examples
      dW = np.dot(X.T, dloss)
      db = np.sum(dloss)
      W += -.1 * dW
      b += -.1 * db
  return W, b
```

6. *Coding: Implement a prediction function* (15 points)

   In this problem you are asked to implement a prediction function in Python. Please try to make the code as syntactically correct as possible; partial credit will be given when it is not.

   Recall the training function in our "numpy-complete" implementation of a 2-layer neural network was a function `train_2_layer_network` that started and ended like this:

```
def train_2_layer_network(H1=100):
    # initialize parameters randomly
    # H1 = 100 # size of hidden layer
    W1 = np.random.randn(D,H1)
    b1 = np.zeros((1,H1))
    W2 = np.random.randn(H1,K)
    b2 = np.zeros((1,K))

    # some hyperparameters
    step_size = 1e-1

    # gradient descent loop
    num_examples = X.shape[0]
    for i in range(20000):

      #implementation of backpropagation here
      ...

      # perform a parameter update
      W1 += -step_size * dW1
      b1 += -step_size * db1
      W2 += -step_size * dW2
      b2 += -step_size * db2

  return W1, b1, W2, b2
```

   Suppose that we have a call to this function:

```
W1, b1, W2, b2 = train_2_layer_network(100)
```

   (a) Write a function `predict(X, W1, b1, W2, b2)` that takes a matrix of inputs `X` and makes predictions for the class labels. Your function should return an array of length `X.shape[0]`, the number of rows of `X`. You should assume that the rectified linear activation was used during training.

6. *Coding: Implement a prediction function (continued)*

```
def predict(X, W1, b1, W2, b2):
  hidden_layer = np.maximum(0, np.dot(X, W1) + b1)
  scores = np.dot(hidden_layer, W2) + b2
  exp_scores = np.exp(scores)
  probs = exp_scores/np.sum(exp_scores, axis=1, keepdims=True)
  return np.argmax(probs, axis=1)
```

Note: This is a bit of a fancy way of doing the computation. Longer (but correct!) code that uses loops is certainly acceptable.

6. *Coding: Implement a prediction function (continued)*

   (b) Now write code that takes test data `X` and `y`, computes the predicted labels for X by calling the function from (a), and then prints out the error rate of the predictions.

```
yhat = predict(X, W1, b1, W2, b2)
error_rate = np.sum(yhat != y) / len(y)
print('Error rate: %.2f%%' % (error_rate*100))
```