Name: ___________________________________ NetID: _____________

S&DS 365 / 565

# Introductory Machine Learning

Midterm Exam #2 (Practice, sample solution)

Tuesday, November 19, 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.

For your reference, recall the following distributions:

$$\text{Normal}(\mu, \sigma^2): \quad p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}, \quad x \in \mathbb{R}$$

$$\text{Beta}(\alpha, \beta): \quad p(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta^{\alpha-1}(1-\theta)^{\beta-1}, \quad \theta \in [0, 1]$$

$$\text{Dirichlet}(\alpha_1, \ldots, \alpha_K): \quad p(\theta) = \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \theta_k^{\alpha_k - 1}, \quad \sum_{k=1}^K \theta_k = 1, \ \ \theta_k \geq 0$$

$$\text{Bernoulli}(\theta): \quad \mathbb{P}(z) = \theta^z (1-\theta)^{(1-z)}, \quad z \in \{0, 1\}$$

$$\text{Binomial}(n, p): \quad \mathbb{P}(k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad k \in \{0, 1, \ldots, n\}$$

| 1 | 2 | 3 | 4 | 5 | total |
|---|---|---|---|---|---|
|  |  |  |  |  |  |
| 20 | 10 | 10 | 10 | 10 | 60 |

1. *True or False, Yes or No?* (20 points)

Indicate whether each of the following statements is true or false, by circling your answer.

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) The classes can be determined by bottom-up clustering.

[✓ YES] NO (b) As the number of classes decreases, the training likelihood decreases.

YES [✓ NO] (c) Words that co-occur with each other will tend to be placed in the same class.

[✓ YES] NO (d) The embedding vectors can be obtained by maximizing the training likelihood using stochastic gradient descent.

YES [✓ NO] (e) Both of these are mixture models with latent variables.

Class based bigram model:
- Put words in groups/classes
  - classes are formed by words with similar neighbors NOT by words that cooccur with each other
    - we're not trying to predict the next word to be similar to the one before it, but rather the word likely to be neighbor
- There is a loss/likelihood function (the likelihood is the negative of the loss)
- As you decrease number of classes, you probably won't get a good model (decrease likelihood or increase loss). As the model becomes better, likelihood increases/loss decreases

Bottom-up clustering: don't need to know it that well. Almost like big decision tree where you go from the leaves upward

Only LDA and class based models use latent variables

Gradient descent used everywhere where loss function: logistic regression, perplexity

Embeddings vs class based language models

All are designed to predict probability of current word based on previous

Class based model: too many probabilities to estimate

$$p(w2 \mid w1) = p(w2 \mid class(w2)) * p(class(w2) \mid class(w1))$$

E.g. nouns and verbs

The class is a latent variable - class-based n-gram model is a mixture model

Embedding model: NOT a mixture model

$$p(w2 \mid w1) = \frac{e \wedge (\phi(w2)^T * \phi(w1))}{\sum e \wedge (\phi(w)^T * \phi(w1))}$$

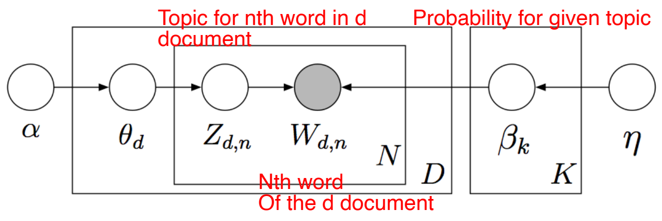
Don’t need to know Gibbs sampling
or Markov chains

Language models: supervised
Topic models: unsupervised

Get to know this diagram
The words are independent unless conditioned on both
Z and β

We don't need to know Θ (the per-document topic proportions) for independence because knowing β is enough

The following questions concern latent Dirichlet allocation (LDA) topic model



where $\theta_d \sim \text{Dirichlet}(\alpha)$ are the per-document topic proportions, $Z_{d,n} \sim \text{Multi}(\theta_d)$ are the per-word topic assignments, $\beta_k \sim \text{Dirichlet}(\eta)$ are the topics, and $W_{d,n} \sim \text{Multi}(\beta_{Z_{d,n}})$ are the observed words.

[✓ YES] NO (a) The objective of posterior inference is to estimate the conditional probability of $\theta_d$, $Z_{d,n}$, and $\beta_k$ given a corpus of documents.

YES [✓ NO] (b) The words in a document are generated independently.

[✓ YES] NO (c) Reordering the documents in the training corpus does not change the posterior distribution over the latent variables.

[✓ YES] NO (d) Reordering the words in each document in the training corpus does not change the posterior distribution over the latent variables.

[✓ YES] NO (e) Conditioned on all of the per-word topic assignments $Z_{d,n}$, the posterior distribution over $\theta_d$ is a single Dirichlet distribution.

The following questions concern the t-SNE method for visualizing word embeddings.

[✓ YES] NO (a) The method is used to visualize high dimensional embeddings.

YES [✓ NO] (b) The goal of the algorithm is to preserve all pairwise distances between words in the high dimensional embedding space. Does it squish/contort these distances?

[✓ YES] NO (c) The algorithm attempts to place words nearby in the 2-dimensional visualization that are very close together in the high dimensional space.

YES [✓ NO] (d) The method gives another representation of the data that is commonly used in other machine learning algorithms.

YES [✓ NO] (e) The method is based on the use of latent variable models.

YES [✓ NO] (f) The method uses Bayesian inference.

[✓ YES] NO (g) The visualization is optimized using stochastic gradient descent.

2. *Bayesian inference* (10 points)

Suppose $X$ is a Bernoulli$(\theta)$ random variable and we observe data $D_n = \{x_1, \ldots, x_n\}$. Suppose the prior distribution on $\theta$ is Beta$(\alpha, \beta)$. Let $s = \sum_{i=1}^n x_i$ be the number of "successes".

(a) What is the posterior distribution of $\theta$ given $D_n$?

Beta$(s + \alpha, n - s + \beta)$

(b) What is the posterior mean?

$$\mathbb{E}(\theta \,|\, x) = \frac{s + \alpha}{n + \alpha + \beta}$$

Perplexity: a perplexity of 1 is good - you're confident. Perplexity of 100 means you've got 100 options to choose from that are just as likely as each other to come next. Lower perplexity is good. It's a kind of loss function. If you minimize perplexity, you can roughly maximize likelihood.

3. *Embeddings* (10 points)

(a) Define the pointwise mutual information

Definition of near can vary e.g. right next to each other or within 5 words of each other.

$$\text{PMI}(w_1, w_2) = \log\left(\frac{p_{\text{near}}(w_1, w_2)}{p(w_1)\,p(w_2)}\right)$$

where $p_{\text{near}}(w_1, w_2)$ is the probability that words $w_1$ and $w_2$ appear near each other in the corpus.

If w1 and w2 is independent, then PMI is 0, i.e. log 1

(b) Consider the analogy "puppy is to dog as ? is to cat" Use $\phi(\text{puppy})$, $\phi(\text{dog})$, and $\phi(\text{cat})$ to denote the embeddings of these three words. Write down the missing word as the solution to an optimizition problem.

$$\widehat{w} = \operatorname*{argmin}_{w \in V} \|\phi(\text{puppy}) - \phi(\text{dog}) + \phi(\text{cat}) - \phi(w)\|$$

where $\text{argmin}_w$ means "the word $w$ that minimizes"

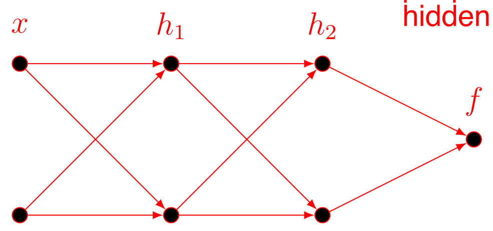
Look at https://piazza.com/class/jzr3shwhvh818k?cid=167 for a correction

4. *Neural nets* (10 points)

Consider a function $f(x) = \sigma(W_2\sigma(W_1x + b_1) + b_2)$ where $\sigma$ is an activation function. Considering $f$ as a neural network, suppose that the input dimension is $d = 2$, and there are two neurons in each hidden layer.

(a) Draw a picture of $f$ as a network.

The output and input layers are sometimes not counted in the number of layers, so people usually just specify the number of hidden layers



(b) Suppose the activation function is $\sigma(x) = \text{relu}(x)$ is the rectified linear unit, and that

$$W_1 = \begin{pmatrix} 2 & 1 \\ 2 & 2 \end{pmatrix} \quad b_1 = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$
$$W_2 = \begin{pmatrix} 1 & -1 \end{pmatrix} \quad b_2 = -1.$$

Compute the value of $f((2, -2)^T)$.

$f((2,-2)^T) = 0$ because

$$\sigma\left(\begin{pmatrix} 2 & 1 \\ 2 & 2 \end{pmatrix}\begin{pmatrix} 2 \\ -2 \end{pmatrix} + \begin{pmatrix} -1 \\ 0 \end{pmatrix}\right) = \sigma\left(\begin{pmatrix} 2 \\ 0 \end{pmatrix} + \begin{pmatrix} -1 \\ 0 \end{pmatrix}\right) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$
$$\sigma\left(\begin{pmatrix} 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} - 1\right) = 0$$

5. *Code* (10 points)

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

```
np.mean([i+1 for i in range(3)])
```

2.0 as a float

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

```
def mystery(X, Y):
  n, d = X.shape

  W1 = .1*np.random.randn(d,10)
  b1 = np.random.randn(1,10)
  W2 = .1*np.random.randn(10,1)
  b2 = np.random.randn(1,1)

  for i in range(1000):
      h = np.dot(X, W1) + b1
      Yhat = np.dot(h, W2) + b2
      dloss = Yhat - Y
      dloss /= num_examples
      dW2 = np.dot(h.T, dloss)
      db2 = np.sum(dloss)
      dh = np.dot(dloss, W2.T)
      dW1 = np.dot(X.T, dh)
      db1 = np.sum(dh)
      W1 += -.01 * dW1
      b1 += -.01 * db1
      W2 += -.01 * dW2
      b2 += -.01 * db2
  return Yhat
```

The function returns the least squares linear regression fits $\widehat{Y} = \widehat{\beta}_1 X + \widehat{\beta}_0$. They are computed using a neural network with linear activation function, trained with gradient descent.