CS229 Final Exam - Summer 2020

Aug 14 2020 7am PDT - Aug 15 2020 7am PDT

This exam is open-notes and open-internet. However, any and all forms of online or offline discussion is prohibited (for instance, posting questions on StackExchange is not permitted). We expect the exam to be completable in roughly 3 hours, but are giving you a 24-hour submission window to accommodate various time zones and upload speeds onto Gradescope. Good luck!

[0 points] The Stanford University Honor Code

At the top of your solution file, please write/type the following Honor Code statement and attest it (i.e. sign or print your name below it), thereby implying your consent to follow the code:

"*I attest that I have not given or received aid in this examination, and that I have done my share and taken an active part in seeing to it that others as well as myself uphold the spirit and letter of the Honor Code.*"

Submissions without the attested Honor Code statement will NOT be graded.

"*I attest that I have not given or received aid in this examination, and that I have done my share and taken an active part in seeing to it that others as well as myself uphold the spirit and letter of the Honor Code.*"

Anh Vu L. Nguyen

1 [10 points] True or False

Each question is worth 1 point. Provide a short justification (one or two lines) for each answer. There will be no negative points, but answers without justification will not receive credit.

For questions 1-3, consider a binary classification task where we have n training examples, $x^{(i)} \in \mathbb{R}^d$, and hypothesis $h_\theta(x)$ is parameterized by θ.

1. **True or False**. Removing half the training data will increase variance.

   Answer: True, this should mean that the model will have less to work with for prediction. The test prediction will likely be larger than before, indicating incrase variance.

2. **True or False**. Removing half of the features of x will increase bias.

   Answer: True, this would increase bias since it forces the model to assume more with less information available.

3. **True or False**. Removing half of the features of x will increase variance.

   Answer: False, removing features leads to increase in bias, which could lead to decrease in variance.

4. **True or False**. Adding a constant to the arguments of Radial Basis Function (RBF) kernel will not change the value of the kernel.

   Answer: True, since RBF kernels rely on calculating squared distance. This squared distance would be unchanged as $((x+c)-(x'+c))^2 = ((x-x')^2$.

5. **True or False**. The following probability density function parameterized by θ belongs to the exponential family.

$$f(x;\theta) = \frac{\theta^x}{2\pi x^3} e^{-\theta} \tag{1}$$

   Answer: True, we can derive the parameters:

$$f(x;\theta) = \frac{\theta^x}{2\pi x^3} e^{-\theta} = \frac{1}{2\pi x^3}(\theta^x\ e^{-\theta})$$

$$= \frac{1}{2\pi x^3} \exp(\log(\theta^x\ e^{-\theta}))$$

$$= \frac{1}{2\pi x^3} \exp(\log(\theta^x) + \log(e^{-\theta}))$$

$$= \frac{1}{2\pi x^3} \exp(x\log(\theta) - \theta)$$

   Rewrite this with y instead of x:

$$f(y;\theta) = \frac{1}{2\pi y^3} \exp(y\log(\theta) - \theta)$$

   The parameters:

$$b(y) = \frac{1}{2\pi y^3};\ T(y) = y;$$

$$\eta^T = \log(\theta) \implies \eta = \log(\theta) \implies \theta = e^\eta$$

$$\alpha(\eta) = \theta = e^\eta$$

6. **True or False**. When training a neural network, it is a good practice to initialize all weights to 0.

   Answer: False, we need random initialization. The reason is discussed in "Parameters Initialization" in Deep Learning Notes, but essentially to avoid the same output for first layer.

7. **True or False**. There is a unique solution to k-means clustering regardless of how we initialize the k centers.

   Answer: False, depending on how we initialize the k centers, it'll impact the distance calculation to determine which group an examples belong to. This can in turn result in a different k-means solution as we run the algorithm each time through the same dataset.

For questions 8-10, suppose we train a SVM classifier to perform binary classification using hinge loss, i.e.

$$L(y_i) = \max\{0, 1 - y_i(w^T x_i + b)\} \tag{2}$$

8. **True or False**. Adding a regularization term to the loss function can change the decision boundary.

   Answer: True, just like in class discussion on regularization and the non-separable case, this technique is used to make decision boundary less sensitive to outliers.

9. **True or False**. Scaling each x_i by constant c can change the decision boundary.

   Answer: False, since scaling all training data point by a constant c means that the distance from the points to decision boundary remains the same. SVM will again give us the same decision boundary.

10. **True or False**. Removing all data points that are not support vectors (i.e., their distance from the decision boundary is greater than 1) can change the decision boundary.

    Answer: If done before introducing a new training, then True, because SVM would try to recreate the decision boundary using only the leftover points. However, if we mean that after we train, we only save the support vectors, and try to make prediction on a new point, then False, decision boundary remains the same.

2 [10 points] Short Answers

Each question is worth 2 points. Provide a short justification (one or two lines) for each answer.

1. **Short Answer**. What 3 components can Mean Squared Error be decomposed into? Which component can be reduced by using a larger training set? Which component can be reduced by adding another hidden layer in a neural net? Which component can be reduced by decreasing the dimension of parameter θ?

   Answer: MSE = Irreducible Error+ Bias^2 + Variance

   Which component can be reduced by using a larger training set? Variance. This leads to the model potentially reduce the gap between cross-validation error and training error through learning from more examples.
   Which component can be reduced by adding another hidden layer in a neural net? Bias. The model is now bigger and more complicated, leading it to learn more pontentially from its current set of features. Training error might be lower, indicating lower bias.
   Which component can be reduced by decreasing the dimension of parameter θ? This action would means that the parameter would have less expressive power. Depending on the data, this might cause it to perform worse on training error. An example if we're reducing parameters to 1 dimension, but we're predicting data that's much better expressed in 3 dimensions. Therefore, this seems to increase bias which might result in increase in variance.

2. **Short Answer**. How many parameters are necessary to specify a *Bernoulli Event Model* Naive Bayes classifier with c classes and vocabulary size v? Express your answer in terms of c and v.

   Answer: We would need $v \times c$ parameters in total.

3. **Short Answer**. Suppose we train a neural network to classify 16 x 16 pixel images into 4 classes. We flatten each image into a vector, and feed the images into a 3-layer neural net with 10 neurons in the first hidden layer, 6 neurons in the second hidden layer, and apply softmax loss at the output layer. How many parameters does this model have? (Don't forget to include biases!)

   Answer: For the first layer, the number of params $= (10 \times (16 \times 16)) + 10 = (10 \times 256) + 10$.
   For the second layer, the number of params $= (6 \times 10) + 6$.
   For the third layer, the number of params $= (4 \times 6) + 4$.
   The model has 2664 parameters in total.

4. **Short Answer**. Consider a Markov Decision Process (MDP) defined as $(S, A, \{P_{sa}\}, \gamma, R)$, where S is a finite set of m states, A is a finite set of n actions, $\{P_{sa}\}$ are the transition probabilities, γ is the discount factor, and R is the reward function. How many distinct deterministic policies $\pi : S \to A$ can be defined for this MDP?

   Answer: We can have up to n^m policies. We have n possible actions and m states, and we can cycle through all m states in the MDP with each state having n possible actions (therefore, n^m).

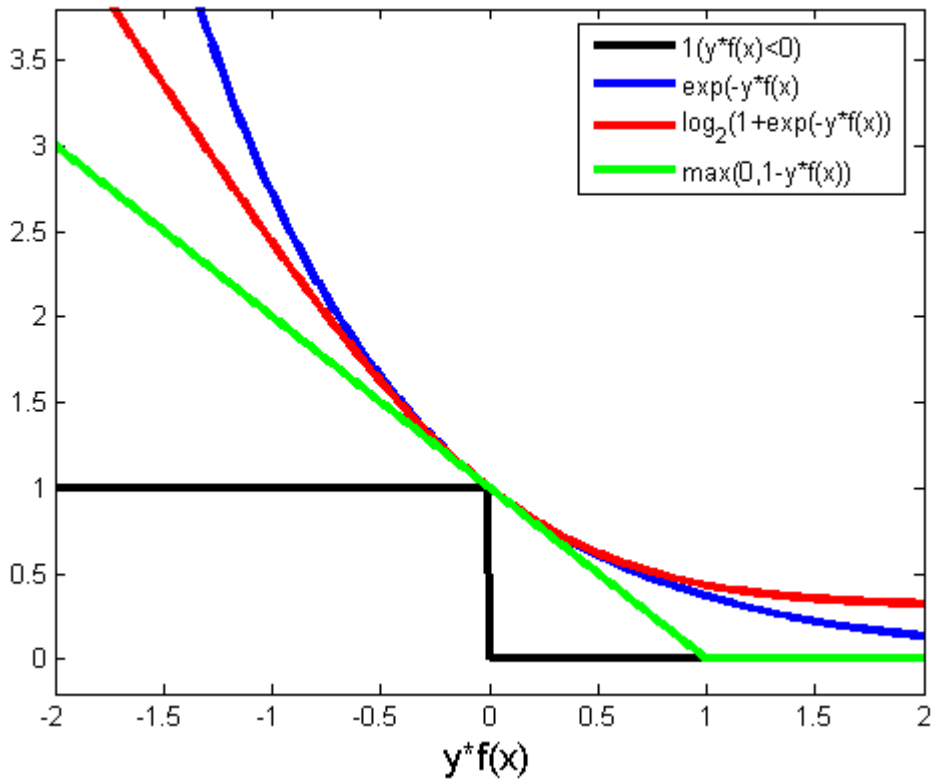
5. **Short Answer**. Suppose we want to minimize a loss function parameterized by $\theta \in \mathbb{R}^d$, where $x \in \mathbb{R}^d$, $y \in \{-1, 1\}$, $z = y\theta^T x$, and the current value of z is -1.5. Which of the following loss functions would gradient descent fail to decrease, regardless of step size?

   (a) zero-one loss: $\varphi(z) = 1\{z \leq 0\}$

   (b) exponential loss: $\varphi(z) = e^{-z}$

   (c) hinge loss: $\varphi(z) = \max\{1 - z, 0\}$

Answer: I found a good graphical plot for all these loss functions (source: https://rohanvarma.me/Loss-Functions):



(a) would not work here as the loss function is a straight line and does not have anything for gradient descent to work with.
(b) should work.
(c) should work since $z = -1.5$ currently.

3 [10 points] Theory *(Kernel Nearest Neighbor for Spam Classification)*

In this problem, we will revisit the spam classification problem with the Kernel method.

Recall that in homework 2 we implemented a Multinomial Event Model Naive Bayes classifier. Now we are going to combine the kernel method with the K-nearest neighbor classification model. For simplicity, we use $k = 1$ in the following scenario: for an incoming email, we calculate its kernel distance with each email in the training set, and assign the incoming email the label of the closest neighbor[1]. The performance of the algorithm highly depends on the kernel we choose, and we are going to compare options in this question.

Let $S_i = [s_1, s_2, \cdots, s_{N_i}]$ be a string sequence (i.e. an email). Vocabulary V is the set of all strings observed in any S_i. For a given sequence S_i, let $g_n(S_i)$ denote the set of all n-grams of S_i. For example,

$$g_2([\text{I, saw, a, saw, that, saw, a, saw}]) = \{\text{I saw, saw a, a saw, saw that, that saw}\}$$

Notice here our 2-gram function only extracts one copy of 'saw a' and 'a saw'. Furthermore, we can define a string comparison function:

$$\delta(s_j, s_k) = \begin{cases} 1, & \text{if } s_j = s_k \text{ (i.e. the two n-gram strings are the same)} \\ 0, & \text{otherwise.} \end{cases}$$

(a) [4 points] We define a kernel function as:

$$K_{V,n}(S_1, S_2) = \sum_{x \in g_n(S_1)} \sum_{z \in g_n(S_2)} \delta(x, z)$$

Express the kernel function using an appropriate feature mapping $\phi(S_i)$. That is, define the feature mapping $\phi(S_i) \in \mathbb{R}^d$ such that $K_{V,n}(S_1, S_2) = \phi(S_1)^\top \phi(S_2)$. What is the dimension of the feature mapping space d in terms of n and V?

Hint: The description of $\phi(S_i)$ should be brief.

Answer:

$$K_{V,n}(S_1, S_2) = \sum_{x \in g_n(S_1)} \sum_{z \in g_n(S_2)} \delta(x, z) = \sum_{x \in g_n(S_1)} \sum_{z \in g_n(S_2)} 1\{x = z\} = \sum_{x \in g_n(S_1)} \sum_{z \in g_n(S_2)} 1\{x = z\}\, 1\{z = x\}$$

The last equality is due the fact that if x=z then z=x as well. The operation here looks like we need to go through and count co-occurence between the two set of n-grams for S_1, S_2. The co-occurence would mean that the n-gram would have to be present in both $g_n(S_i)$ and $g_n(S_j)$. Since we know the vocabulary V, we could construct a high dimensional space for the feature space that combines the lenght of V and also the length of n-grams.

More concretely, given vocab V with v elements in total, we construct a feature map where each row represents whether the n-grams exists in S_j. For example, given a simplified vocab of {"A", "B", "C", "D", "E"} and $n = 2$ for n-grams, we could construct a vector like follow for S_1:

$$\begin{bmatrix} 1\{\{\text{"A, A"}\} \in S_1\} \\ 1\{\{\text{"A, B"}\} \in S_1\} \\ 1\{\{\text{"A, C"}\} \in S_1\} \\ 1\{\{\text{"A, D"}\} \in S_1\} \\ 1\{\{\text{"A, E"}\} \in S_1\} \\ 1\{\{\text{"B, A"}\} \in S_1\} \\ 1\{\{\text{"B, B"}\} \in S_1\} \\ 1\{\{\text{"B, C"}\} \in S_1\} \\ 1\{\{\text{"B, D"}\} \in S_1\} \\ 1\{\{\text{"B, E"}\} \in S_1\} \\ \cdots \end{bmatrix}$$

[1] If $k > 1$, we instead use the majority vote from the k nearest neighbors as the predicted label.

The feature mapping d will has dimension of $(V^n, 1)$ where V is the number of words in dictionary and n is from n-grams.

(b) [3 points] In reality, the vocabulary size is usually large. For example *WikiText-103*, which contains all articles extracted from Wikipedia, has a vocabulary size of 217,646. However, the frequency of words has a long tail distribution and some of the words rarely appear. We can replace those rare words with a designated "unknown token" UNK to effectively reduce the vocabulary size. For example, if we restrict the vocabulary to V' which only the has three tokens $\{\text{I}, \text{saw}, \text{UNK}\}$, our previous example becomes: S_i' = [I, saw, UNK, saw, UNK, saw, UNK, saw], and

$$g_2(S_i') = \{\text{I saw, saw UNK, UNK saw}\}$$

Let V' be the reduced vocabulary, i.e. a subset of $\{V \cup \{\text{UNK}\}\}$. We define the kernel function for V' as:

$$K_{V',n}(S_1, S_2) = \sum_{x \in g_n(S_1')} \sum_{z \in g_n(S_2')} \delta(x, z)$$

Is this still a valid kernel? If yes, express the kernel function using an appropriate feature mapping $\phi'(S_i)$ and state its relation with $\phi(S_i)$. If not, please give a counterexample.

Answer: This is still a valid kernel.

We restrict the vocab to V' and include a out-of-vocab or UNK token. So an immediate impact of restricting Vocabulary would be lower dimensional feature space compared to V.

The Kernel function:

$$K_{V',n}(S_1, S_2) = \sum_{x \in g_n(S_1)} \sum_{z \in g_n(S_2)} \delta(x, z)$$

The feature mapping would be similar to above kernel ($K_{V,n}(S_1, S_2)$). The difference here seems to be that the x has already gone through a previous feature map. This extra feature map just map input to a lower dimensional space (since V' should have fewer vocab than V) where dimension equals to v' where v' is the number of element in V' (including UNK).

On a related note, back in PSET2 Q3, we have discussed constructing Kernels. The feature mapping for V' can be viewed as $K(x, z) = K_3(\phi(x), \phi(x))$ where the initial mapping to restricted V' is the $\phi(x)$. Since we know $K_{V,n}(S_1, S_2)$ is a valid kernel, so by extension, $K_{V',n}(S_1, S_2)$ is also a valid kernel.

(c) [2 points] Which of the following is true for any V'? Give a short explanation.

(A). $K_{V,n}(S_1, S_2) \geq K_{V',n}(S_1, S_2)$
(B). $K_{V,n}(S_1, S_2) \leq K_{V',n}(S_1, S_2)$
(C). $K_{V,n}(S_1, S_2) > K_{V',n}(S_1, S_2)$
(D). $K_{V,n}(S_1, S_2) < K_{V',n}(S_1, S_2)$
(E). None of the above

Answer: The mapping to UNK should account for the common n-grams in the string. We can run through a couple of examples:
Using the string S_1="I saw that that saw an apple", and create 2-grams. For the original case with V, we have:

$$g_2([\text{I, saw, that, that, saw, an, apple}]) = \{\text{I saw, that that, that saw, saw an, an apple}\}$$

For the case with $V' = (\text{I, saw, UNK})$, we have:

$$g_2([\text{I, saw, UNK, UNK, saw, UNK, UNK}]) = \{\text{I saw, UNK UNK, UNK saw, saw UNK}\}$$

Test with a couple of string:

S_2= (that saw) will give us $K_{V,2}(S_1, S_2) = K_{V',2}(S_1, S_2) = 1$
S_2= (an apple) will give us $K_{V,2}(S_1, S_2) = K_{V',2}(S_1, S_2) = 1$
S_2= (an apple, an orange) will give us $K_{V,2}(S_1, S_2) = K_{V',2}(S_1, S_2) = 1$ since both (an apple, an orange) are mapped to a singular (UNK UNK) and will only be counted once.
S_2= (an apple, that saw) will give us $K_{V,2n}(S_1, S_2) = K_{V',2}(S_1, S_2) = 2$ since (an apple, that saw) are mapped to (UNK UNK, UNK saw), which are both counted.
However,
S_2= (an apple, that that) will give us $K_{V,2}(S_1, S_2) = 2 > K_{V',2}(S_1, S_2) = 1$. For V', (an apple, that that) is mapped to (UNK UNK), and is counted once as common string. For V, we count both "an apple" and "that that", giving us 2.
This can be thought of as a consequence of many to one mapping in V'. As to whether $K_{V,n}(S_1, S_2) < K_{V',n}(S_1, S_2)$ is possible, we look at the following example:
S_2= (an orange) will give us $K_{V,2n}(S_1, S_2) = 0 < K_{V',2}(S_1, S_2) = 1$ since (an orange) is mapped to (UNK UNK), which is counted for V'. V maintains the original string, which then means it isn't counted for any common string here.

Therefore, answer is E here.

(d) [1 points] Now we will apply this kernel to our spam classification problem. We hope that the nearest neighbor found through $K_{V',n}$, which is computationally more efficient, is the same as the nearest neighbor from the kernel space defined by $K_{V,n}$.

For any sequence S_i in the training set $\mathcal{S}_{\text{train}}$, denote its ground truth label as $y(S_i)$. The Kernel Nearest Neighbor prediction for S_t in the test set is:

$$f_{V,n}(S_t) = y\left(\arg \max_{S_i \in \mathcal{S}_{\text{train}}} K_{V,n}(S_i, S_t)\right)$$

We hope to observe:

$$f_{V',n}(S_t) = f_{V,n}(S_t) \qquad (*)$$

for any reduced vocabulary V'.

Unfortunately, this is not true. We observe a trade-off between computational cost and accuracy: the smaller the vocabulary, the less computation is needed, but the more likely we do not find the true

nearest neighbor. However, as long as the new vocabulary set V' is not too small, this is still a good approximate classification model.

Give a concrete counterexample to the equation $(*)$. You can choose any vocabularies V, V', and n for your example.

Answer:

$$f_{V,n}(S_t) = y\left(\arg\max_{S_i \in \mathcal{S}_{\text{train}}} K_{V,n}(S_i, S_t)\right)$$

$$f_{V',n}(S_t) = y\left(\arg\max_{S_i \in \mathcal{S}_{\text{train}}} K_{V',n}(S_i, S_t)\right)$$

We want to give counte example to:

$$y\left(\arg\max_{S_i \in \mathcal{S}_{\text{train}}} K_{V,n}(S_i, S_t)\right) = y\left(\arg\max_{S_i \in \mathcal{S}_{\text{train}}} K_{V',n}(S_i, S_t)\right)$$

Let's choose an example where $n = 1$, which means that we're just counting common word between two string for the kernel. Then let's take V to be all the words in the English dictionary, while $V' = \{\text{UNK}\}$, effectively reduce the vocab to 1 words. It's easy to see here that for any string that's actually the same, meaning $y = 1$, V' will map that to just UNK, making the maximum count of common to be 1 (since we also don't extract multiple same copy from the prompt). Let's say we have 2 string to train with S_1=("A B C") and S_2=("A B B C D"). For V, we have 1-gram {"A", "B", "C"} and {"A", "B", "C", "D"}. For V', we have 1-gram {UNK}.

$$f_{V,n}(S_1) = y\left(\arg\max_{S_i \in \mathcal{S}_{\text{train}}} K_{V,n}(S_i, S_1)\right) = y(3)$$

$$f_{V',n}(S_1) = y\left(\arg\max_{S_i \in \mathcal{S}_{\text{train}}} K_{V',n}(S_i, S_1)\right) = y(1)$$

$$\implies f_{V',n}(S_t) \neq f_{V,n}(S_t)$$

4 [10 points] Theory + Coding *(The Poetic Scientist)*

In this question, you will train a neural network to help write poetry. The goal is to find words that sound similar to each other. Weak supervision is one approach to learn how phonetically close or far two words are: your model will learn from and predict whether two inputs are similar or different sounding. Consider a two-layer network parameterized as follows:

$$h^{[1]}(x) = W^{[1]}x + b^{[1]} \tag{3}$$
$$a^{[1]}(x) = \sigma(h^{[1]}(x)) \tag{4}$$
$$h^{[2]}(x) = W^{[2]}a^{[1]}(x) + b^{[2]} \tag{5}$$

Our training examples will now consist of *pairs* of inputs along with a label for whether the inputs are similar or dissimilar: $(x_1^{(i)}, x_2^{(i)}, y^{(i)}) \in \mathbb{R}^d \times \mathbb{R}^d \times \{-1, 1\}$ where -1 indicates a dissimilar pair and 1 a similar pair. Consider the following loss:

$$\ell(x_1, x_2, y; W, b) = \log\left(1 + \exp(-yh^{[2]}(x_1)^\top h^{[2]}(x_2))\right) \tag{6}$$

Theory: [6 points]

(a) [1.5 points] Consider the loss for a single example. Write the expressions for $\frac{\partial \ell}{\partial h^{[2]}(x_1)}$, $\frac{\partial \ell}{\partial h^{[2]}(x_2)}$.

Answer:

$$\frac{\partial}{\partial h^{[2]}(x_1)}\ell(x_1, x_2, y; W, b) = \frac{1}{1 + \exp(-yh^{[2]}(x_1)^\top h^{[2]}(x_2))}\left(0 + \exp(-yh^{[2]}(x_1)^\top h^{[2]}(x_2))(-yh^{[2]}(x_2))\right)$$
$$= \frac{-yh^{[2]}(x_2)e^{(-yh^{[2]}(x_1)^\top h^{[2]}(x_2))}}{1 + e^{-yh^{[2]}(x_1)^\top h^{[2]}(x_2)}}$$

$$\frac{\partial}{\partial h^{[2]}(x_2)}\ell(x_1, x_2, y; W, b) = \frac{1}{1 + \exp(-yh^{[2]}(x_1)^\top h^{[2]}(x_2))}\left(0 + \exp(-yh^{[2]}(x_1)^\top h^{[2]}(x_2))(-yh^{[2]}(x_1))^\top\right)$$
$$= \frac{-yh^{[2]}(x_1)^\top e^{(-yh^{[2]}(x_1)^\top h^{[2]}(x_2))}}{1 + e^{-yh^{[2]}(x_1)^\top h^{[2]}(x_2)}}$$

(b) [1.5 points] Express $\frac{\partial \ell}{\partial W^{[2]}}$, $\frac{\partial \ell}{\partial b^{[2]}}$ in terms of $\frac{\partial \ell}{\partial h^{[2]}(x_1)}$ and $\frac{\partial \ell}{\partial h^{[2]}(x_2)}$ and $a^{[1]}(x_1), a^{[1]}(x_2)$.

Answer:

$$\frac{\partial \ell}{\partial W^{[2]}} = \begin{bmatrix} \frac{\partial \ell}{\partial W_1^{[2]}} \\ \frac{\partial \ell}{\partial W_2^{[2]}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \ell}{\partial h^{[2]}(x_1)}\frac{\partial h^{[2]}(x_1)}{\partial W_1^{[2]}} \\ \frac{\partial \ell}{\partial h^{[2]}(x_2)}\frac{\partial h^{[2]}(x_2)}{\partial W_2^{[2]}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \ell}{\partial h^{[2]}(x_1)}\frac{\partial}{\partial W^{[2]}}W_1^{[2]}a^{[1]}(x_1) + b^{[2]} \\ \frac{\partial \ell}{\partial h^{[2]}(x_2)}\frac{\partial}{\partial W^{[2]}}W_2^{[2]}a^{[1]}(x_2) + b^{[2]} \end{bmatrix} = \begin{bmatrix} \frac{\partial \ell}{\partial h^{[2]}(x_1)}a^{[1]}(x_1) \\ \frac{\partial \ell}{\partial h^{[2]}(x_2)}a^{[1]}(x_2) \end{bmatrix}$$

$$\frac{\partial \ell}{\partial b^{[2]}} = \begin{bmatrix} \frac{\partial \ell}{\partial h^{[2]}(x_1)}\frac{\partial h^{[2]}(x_1)}{\partial b_1^{[2]}} \\ \frac{\partial \ell}{\partial h^{[2]}(x_2)}\frac{\partial h^{[2]}(x_2)}{\partial b_2^{[2]}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \ell}{\partial h^{[2]}(x_1)}\frac{\partial}{\partial b_1^{[2]}}W^{[2]}a^{[1]}(x_1) + b_1^{[2]} \\ \frac{\partial \ell}{\partial h^{[2]}(x_2)}\frac{\partial}{\partial b_2^{[2]}}W^{[2]}a^{[1]}(x_2) + b_2^{[2]} \end{bmatrix} = \begin{bmatrix} \frac{\partial \ell}{\partial h^{[2]}(x_1)}(1) \\ \frac{\partial \ell}{\partial h^{[2]}(x_2)}(1) \end{bmatrix} = \begin{bmatrix} \frac{\partial \ell}{\partial h^{[2]}(x_1)} \\ \frac{\partial \ell}{\partial h^{[2]}(x_2)} \end{bmatrix}$$

(c) [1.5 points] Express $\frac{\partial \ell}{\partial h^{[1]}(x_1)}$, $\frac{\partial \ell}{\partial h^{[1]}(x_2)}$ in terms of model parameters and $\frac{\partial \ell}{\partial h^{[2]}(x_1)}$, $\frac{\partial \ell}{\partial h^{[2]}(x_2)}$.

Answer:

$$\frac{\partial \ell}{\partial h^{[1]}(x_1)} = \frac{\partial \ell}{\partial h^{[2]}(x_1)} \frac{\partial h^{[2]}(x_1)}{\partial W_1^{[2]}} \frac{\partial W_1^{[2]}}{\partial a^{[1]}(x)} \frac{\partial h^{[2]}(x_1)}{\partial a^{[1]}(x)} \frac{\partial a^{[1]}(x)}{\partial h^{[1]}(x_1)} =$$

$$\frac{\partial \ell}{\partial h^{[1]}(x_2)}$$

(d) [1.5 points] Express $\frac{\partial \ell}{\partial W^{[1]}}$, $\frac{\partial \ell}{\partial b^{[1]}}$ in terms of $\frac{\partial \ell}{\partial h^{[1]}(x_1)}$, $\frac{\partial \ell}{\partial h^{[1]}(x_2)}$ and x_1, x_2.

Answer:

Coding: [4 points]

(e) Your friend suggests an alternative to your approach: A single layer, shared by both inputs $x_1^{(i)}$ and $x_2^{(i)}$, that outputs two vectors, $z_1^{(i)} \in \mathbb{R}^{25}$ and $z_2^{(i)} \in \mathbb{R}^{25}$, respectively. Each output represents the input's word embedding: a representation of the input as a point in a vector space that's conducive to further processing. We are interested in the distance $d^{(i)}$ between two embeddings $z_1^{(i)}$ and $z_2^{(i)}$. The weakly supervised labels $y^{(i)} \in \{0, 1\}$ where 0 indicates a similar-sounding pair and 1 a different-sounding one; B is the batch size.

Extract the zip file and run `conda env create -f environment.yml` from inside the extracted directory. This creates a Conda environment called `cs229-final`. Run `source activate cs229-final` to activate this environment. In `train.py`, please fill in the functions corresponding to (e), (f), (g), and include the 3 generated outputs in part (h).

(i) Fill in the code for the `batch_dot` function ($a \cdot b$ denotes the dot product):

$$\text{batch\_dot}(z_1, z_2) = \begin{bmatrix} z_1^{(1)} \cdot z_2^{(1)} \\ \vdots \\ z_1^{(i)} \cdot z_2^{(i)} \\ \vdots \\ z_1^{(B)} \cdot z_2^{(B)} \end{bmatrix} \in \mathbb{R}^B \tag{7}$$

(ii) Fill in the code for the `distance_loss` function:

$$\text{distance\_loss}(y, d) = \frac{1}{B} \sum_{i=1}^{B} \left[(1 - y^{(i)}) \, (d^{(i)})^2 + y^{(i)} \, (1 - d^{(i)})^2 \right] \tag{8}$$

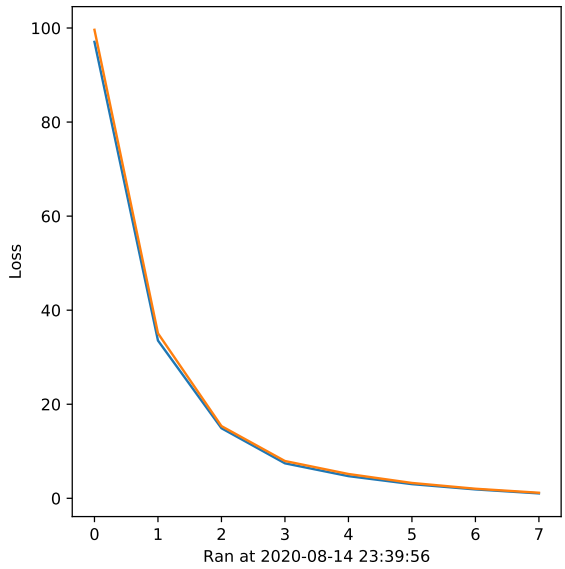
(iii) Use the following to fill in the code for the `accuracy` function:

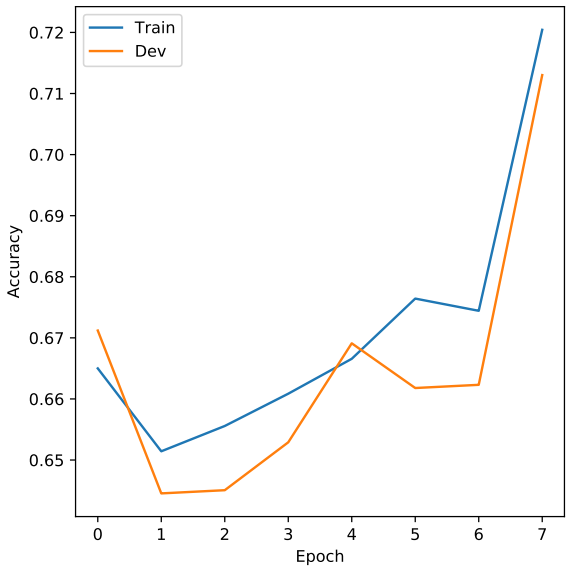
$$\hat{y}^{(i)} = \mathbb{1}\{d^{(i)} > 0.5\} \tag{9}$$

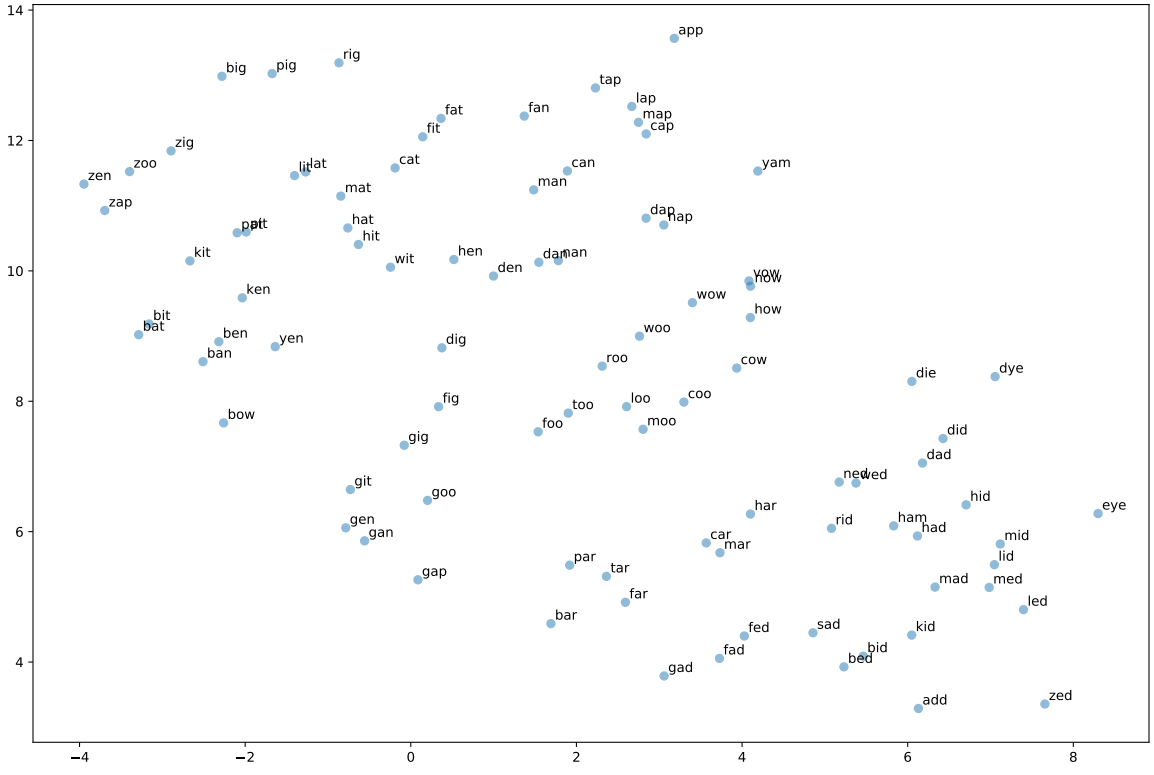
$$\text{accuracy}(y, d) = \frac{1}{B} \sum_{i=1}^{B} \mathbb{1}\{y^{(i)} = \hat{y}^{(i)}\} \tag{10}$$

Run `train.py` and submit the 2 learning curve plots (loss vs epochs, accuracy vs epochs) along with the sample of embeddings projected onto a two-dimensional plot. As usual, pay attention to the documentation in the code and upload your code (`train.py`) to Gradescope. **Hint:** Dev accuracy at the end of training ≈ 0.71.

Answer:







Phonetic Analogies:

"cap" is to "coo" as "zap" is to "zoo"
"how" is to "wow" as "hit" is to "wit"
"yam" is to "ham" as "yen" is to "hen"