CS181 2018 Midterm 1 Review Questions Solution

1. **Linear Regression**

Consider a one-dimensional regression problem with training data $\{x_i, y_i\}$. We seek to fit a linear model with no bias term:

$$\hat{y} = wx$$

a. Assume a squared loss $\frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$ and solve for the optimal value of w^*.

b. What is the prediction for some new observation x, without mention of w?

c. Suppose that we have a generative model of the form $\hat{y} = wx + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and w is known. Given a new x, what is the expression for the probability of $\hat{y}$?
Note: The univariate Gaussian PDF is:

$$\mathcal{N}(a|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-(a-\mu)^2}{2\sigma^2}\right)$$

d. Now assume that w is random and that we have a prior on w with known variance s_0^2:

$$w \sim \mathcal{N}(0, s_0^2)$$

Write down the form of the *posterior* distribution over w. Take logs and drop terms that don't depend on the data and prior parameters, but you do not need to simplify further (i.e. you do not need to complete the square to make it look like a normal).

Solution

a. This question is mostly just math. Take the derivative with respect to w, set it equal to 0, and solve for w:

$$-\sum_{i=1}^{N}(y_i - wx_i)x_i = 0$$

$$-\sum_{i=1}^{N} y_i x_i + w\sum_{i=1}^{N} x_i^2 = 0$$

$$w^* = \frac{\sum_{i=1}^{N} y_i x_i}{\sum_{i=1}^{N} x_i^2}$$

b. Here we just plug in from above:

$$y = x\left(\frac{\sum_{i=1}^{N} y_i x_i}{\sum_{i=1}^{N} x_i^2}\right)$$

c. This is a definitions question. use the form of the univariate Gaussian:

$$p(y|x) = \mathcal{N}(y|wx, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(\frac{-(y - wx)^2}{2\sigma^2}\Big)$$

d. Here we combine everything above.
Prior:

$$p(w) = \mathcal{N}(w|0, s_0^2)$$

Likelihood:

$$p(D|w) = p(\mathbf{y}|\mathbf{x}, w) = \prod_{i=1}^{N} \mathcal{N}(y_i|wx_i, \sigma^2)$$

Posterior:

$$p(w|D) \propto p(w)p(D|w) = \mathcal{N}(w|0, s_0^2) \prod_{i=1}^{N} \mathcal{N}(y_i|wx_i, \sigma^2)$$

Take logs:

$$\ln p(w|D) = const + \frac{-w^2}{2s_0^2} + \sum_{i=1}^{N} \frac{-(y_i - wx_i)^2}{2\sigma^2}$$

Worth noting similarity to ridge regression.

End Solution

2. **Multiclass Classification**

Suppose that we have a K-class classification scenario with training are 1-hot vectors.

We model this problem using a neural network with d units in a single hidden layer, expressed as a column vector $\boldsymbol{\phi}(\mathbf{x}; \mathbf{W}, \mathbf{w}_0) \in \mathbb{R}^d$, which we write as $\boldsymbol{\phi}$. We take a linear combination of these values and pass them to a softmax function to get a final set of K outputs. Let $\mathcal{C}_k$ represent a 1-hot vector with a 1 in the k^{th} index and let $\mathbf{v}_\ell \in \mathbb{R}^d$ be a column vector of weights:

$$p(\mathbf{y} = \mathcal{C}_k | \mathbf{x}; \{\mathbf{v}_\ell\}_{\ell=1}^K, \mathbf{W}, \mathbf{w}_0) = \frac{\exp(\mathbf{v}_k^\top \boldsymbol{\phi})}{\sum_{\ell'=1}^K \exp(\mathbf{v}_{\ell'}^\top \boldsymbol{\phi})}$$

a. Suppose we add the same global bias to each vector of weights in the final layer, i.e. replace $\mathbf{v}_k^\top \boldsymbol{\phi}$ with $\mathbf{v}_k^\top \boldsymbol{\phi} + v_0$ for some scalar v_0 with the same scalar for all k. Does this increase the expressivity of our model? Why or why not?

b. Write down and simplify the log likelihood of a particular observation $(\mathbf{x}_i, \mathbf{y}_i)$, including constants. Assume that we use a sigmoid activation function (don't need to simplify within the sigmoid, just the sums/logs/exps around it)

$$\boldsymbol{\phi}(\mathbf{x}; \mathbf{W}, \mathbf{w}_0) = \boldsymbol{\sigma}(\mathbf{W}\mathbf{x} + \mathbf{w}_0)$$

c. Consider the scenario of drawing items from a distribution and encoding them in binary for communication. An efficient scheme encodes common items with a short code and rare items with longer codes. The *cross-entropy* $\mathbb{E}_{p(x)}[-\ln q(x)]$ can be interpreted as the expected number of required bits to send a randomly chosen item $x \sim p(x)$ using a code optimized for $q(x)$. For classification, we can use the following as a loss:

$$\mathbb{E}_{p(y|x)}[-\ln q(y|x)]$$

where $p(\mathbf{y}|\mathbf{x})$ is 1 for the true class and is 0 otherwise and $q(\mathbf{y} = \mathcal{C}_k|\mathbf{x})$ is your model's prediction (output of the softmax layer for k^{th} class). Write down the expression for the cross-entropy by unpacking the expectation and writing it as a sum of terms and describe its relationship to the log loss in part (b).

d. When running SGD, we need to compute the gradient of the loss for a single example $(\mathbf{x}, \mathbf{y})$ with respect to each of the parameters. Use the above definition of $\boldsymbol{\phi}$ to compute this gradient for the bias vector $\mathbf{w}_0$ using the original negative log likelihood loss.

$$\frac{\partial}{\partial \mathbf{w}_0} - \ln p(\mathbf{y} = \mathcal{C}_k | \mathbf{x}; \mathbf{W}, \mathbf{w}_0, \{\mathbf{v}_\ell\})$$

You may use $\sigma'()$ for the derivative of the sigmoid function and this intermediate term:

$$\frac{\partial}{\partial \mathbf{v}_k^\top \boldsymbol{\phi}} - \ln p(\mathbf{y} = \mathcal{C}_k | \mathbf{x}; \mathbf{W}, \mathbf{w}_0, \{\mathbf{v}_\ell\}) - y_k$$

Solution

a. Normalization:

$$\begin{aligned} p(\mathbf{y} = C_k|\mathbf{x}; \{\mathbf{v}_\ell\}, \mathbf{W}, \mathbf{w}_0) &= \frac{1}{const} \exp(\mathbf{v}_k^\top \boldsymbol{\phi} + v_0) \\ &= \frac{1}{const} \exp(\mathbf{v}_k^\top \boldsymbol{\phi}) \exp(v_0) \\ &= \frac{1}{const'} \exp(\mathbf{v}_k^\top \boldsymbol{\phi}) \end{aligned}$$

So this is effectively the same as the original formula.

b. Assume that y_i is in class k:

$$\begin{aligned} \ln p(\mathbf{y} = C_k|\mathbf{x}; \{\mathbf{v}_\ell\}, \mathbf{W}, \mathbf{w}_0) &= \ln \frac{\exp(\mathbf{v}_k^\top \boldsymbol{\phi})}{\sum_{\ell=1}^K \exp(\mathbf{v}_\ell^\top \boldsymbol{\phi})} \\ &= \mathbf{v}_k^\top \boldsymbol{\phi} - \ln \sum_{\ell=1}^K \exp(\mathbf{v}_\ell^\top \boldsymbol{\phi}) \\ &= \mathbf{v}_k^\top \boldsymbol{\sigma}(\mathbf{W}\mathbf{x} + \mathbf{w}_0) - \ln \sum_{\ell=1}^K \exp(\mathbf{v}_\ell^\top \boldsymbol{\sigma}(\mathbf{W}\mathbf{x} + \mathbf{w}_0)) \end{aligned}$$

c. This is a comprehension question. Here $p(\mathbf{y}|\mathbf{x})$ is our true data, and $q(\mathbf{y}|\mathbf{x})$ is our model, i.e.:

$$p(\mathbf{y} = C_k|\mathbf{x}; \{\mathbf{v}_\ell\}, \mathbf{W}, \mathbf{w}_0)$$

The expectation here is computed empirically as the sum over the data, and the inner term is just applying our model.

$$\begin{aligned} \sum_{i=1}^N \mathbb{E}_{\mathbf{y}_i \sim p(\mathbf{y_i}|\mathbf{x}_i)}[-\ln q(\mathbf{y}_i|\mathbf{x}_i)] = \\ -\sum_{i=1}^N \sum_{\ell=1}^K p(\mathbf{y}_i = C_\ell|\mathbf{x}_i) \ln q(\mathbf{y}_i = C_\ell|\mathbf{x}_i) = \\ -\sum_{i=1}^N \ln q(y_i = C_{true}|\mathbf{x}_i) \end{aligned}$$

The inner sum drops out because p is the true data distribution, which assigns probability 0 to all C_ℓ except for the true class C_{true}. Finally, you should recognize the last term, with q as our estimated distribution, as the same as our loss.

d. We will use three indices in the solution. The correct class is C_{true}. We will use ℓ to sum over classes. We will use j to indicate the index into the basis function: ϕ_j. Also we will drop the parameters of p for space. The key step here is to apply the vector chain rule to get the gradient for the basis vector:

$$\frac{\partial - \ln p(\mathbf{y} = C_{true}|\mathbf{x}; \mathbf{W}, \mathbf{w}_0, \{\mathbf{v}_\ell\})}{\partial \phi_j} = \sum_{\ell=1}^{K} \frac{\partial \mathbf{v}_\ell^\top \boldsymbol{\phi}}{\partial \phi_j} \frac{\partial - \ln p(\mathbf{y} = C_{true}|\mathbf{x})}{\partial \mathbf{v}_\ell^\top \boldsymbol{\phi}}$$
$$= \sum_{\ell=1}^{K} v_{\ell j}(p(\mathbf{y} = C_{true}|\mathbf{x}) - y_\ell)$$

Then we invoke the chain rule once more. Here we (pedantically) start by considering a sum over j', however $\frac{\partial \phi_{j'}}{\partial w_{0j}} = 0$ if $j \neq j'$ since w_{0j} is the bias for basis ϕ_j.

$$\frac{\partial - \ln p(\mathbf{y} = C_{true}|\mathbf{x}; \mathbf{W}, \mathbf{w}_0, \{\mathbf{v}_\ell\})}{\partial w_{0j}} = \sum_{j'} \frac{\partial \phi_{j'}}{\partial w_{0j}} \frac{\partial - \ln p(\mathbf{y} = C_{true}|\mathbf{x})}{\partial \phi_{j'}}$$
$$= \frac{\partial \phi_j}{\partial w_{0j}} \frac{\partial - \ln p(\mathbf{y} = C_{true}|\mathbf{x})}{\partial \phi_j}$$
$$= \frac{\partial \phi_j}{\partial w_{0j}} \sum_{\ell=1}^{K} v_{\ell j}(p(\mathbf{y} = C_{true}|\mathbf{x}) - y_\ell)$$
$$= 1 \times \sigma'(\mathbf{w}_j^\top \mathbf{x} + w_{0j}) \times \sum_{\ell=1}^{K} v_{\ell j}(p(\mathbf{y} = C_k|\mathbf{x}) - y_\ell).$$

Note that you can do this in vector form as well. See CS181 Spring 2017 Neural Network 2 lecture slides 14-24

End Solution

3. **Overfitting and Underfitting**

Harvard Insta-Ice Unit (HI2U) has built a robot that can deliver 24-hour shaved ice to student houses. To prevent collisions, they train three different approaches to classify camera images as containing nearby tourists or open space; if the robot identifies a tourist in its path, it is programmed to halt. The performances of the classifiers are:

	Training Accuracy	Testing Accuracy
Classifier A	75.3%	74.8%
Classifier B	80.3%	77.8%
Classifier C	90.2%	60.0%

where Classifier B has a more expressive model class than A, and classifier C has both a more expressive model class and more features than A. All the classifiers have closed-form solutions, so HI2U is pretty sure that the inference is not hindering performance.

a. If you had to choose one: might Classifier A be overfitting or underfitting? Explain your reasoning.

b. If you had to choose one: might Classifier C be overfitting or underfitting? Explain your reasoning.

c. If you had to guess yes or no: might more training examples significantly boost the test-time performance of Classifier A? Classifier C? Explain your reasoning.

Hint: try to relate your reasoning to model bias and model variance.

Solution

a. Likely it is underfitting, demonstrated by the results of Classifier B. This is likely a bias issue, as we have evidence that a richer model can improve on training accuracy.

b. Likely it is overfitting. 90% training accuracy indicates very little bias, but poor test accuracy shows variance issues.

c. It seems unlikely that more training will help A. There is no indication of a variance issue (train/test accuracy are similar). However for classifier C, more training data would reduce the variance of the rich model.

End Solution

4. **Neural Networks**

Consider the following non-linearity for use in a neural network:

$$f_{0/1}(z) = \left\{ \begin{array}{ll} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{array} \right\}$$

Let $\mathbf{x}$ be a binary feature vector of length 4: $\mathbf{x} \in \{0,1\}^4$. Define neural network A as follows:

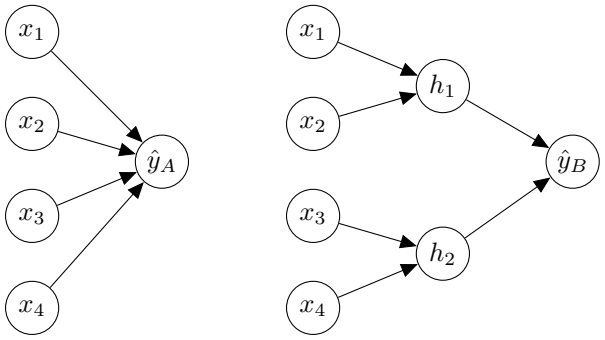
$$\hat{y_A} \leftarrow f_{0/1}(\mathbf{w}^\top \mathbf{x} + w_0)$$

with weight vector $\mathbf{w} \in \mathbb{R}^4$ and bias scalar $w_0 \in \mathbb{R}$.

Let $\mathbf{x}^L = [x_1, x_2]$ and $\mathbf{x}^R = [x_3, x_4]$. Define neural network B as follows:

$$\begin{aligned} h_1 &\leftarrow f_{0/1}(\mathbf{t}^\top \mathbf{x}^L + a) \\ h_2 &\leftarrow f_{0/1}(\mathbf{u}^\top \mathbf{x}^R + b) \\ \mathbf{h} &\leftarrow [h_1, h_2] \\ \hat{y_B} &\leftarrow f_{0/1}(\mathbf{v}^\top \mathbf{h} + c) \end{aligned}$$

with weight vectors $\mathbf{t}, \mathbf{u}, \mathbf{v} \in \mathbb{R}^2$ and bias scalars $a, b, c \in \mathbb{R}$. Basically, B can only look at the two halves of the input separately and has an extra layer to merge the transformations on the two halves of the input with another transformation:



a. i. Describe a logical formula on inputs that can be expressed by A but not by B and provide weights for $\mathbf{w}$ and w_0 that implement the formula in A (hint: think about things you may want to do with binary vectors, e.g. `AND`s, `OR`s)

ii. Provide an argument for why B cannot express this formula (we don't expect a rigorous proof, but try to give a complete and convincing argument).

iii. How might you change the architecture of B to fix this issue? What downside might this have?

b. What is the concern about training the networks as currently defined? What change would you make to the network to alleviate this concern?

c. State **two** ways in which a *validation set* can be used when training neural networks (one sentence for each is fine).

Solution

a. i. One that works is to detect if three or more dimensions are 1:

$$\sum_{j=1}^{4} x_j \geq 3$$

Easy to see that this is solvable with network A:

$$\sum_{j=1}^{4} w_j x_j - 3 \geq 0$$

with $w_j = 1$ for all j.

ii. This can't work for network B though. Argument is that there are 5 cases with at least three 1's:

$$1111, 0111, , 1011, 1101, 1110$$

However, to detect any of the latter four patterns, either side of B's middle layer, h_1 and h_2, needs to be able to pick up the pattern 01 or 10. But that means that the both h_1 and h_2 will fire 1's for:

$$0101, 1010, 0110, 1001$$

iii. The easiest answer is to add more connections. Alternative answer is to add more basis functions. Either way, the downside is that you are adding more parameters to train to the model.

b. As defined, the networks use the 0/1 activation for its basis functions. Similarly to using 0/1 in final layers, this means that the gradients cannot pass back to the weight parameters and they cannot be learned. The easiest way to alleviate this is to switch to the sigmoid σ activation, which is a smooth approximation of 0/1.

c. i. validation can be used to set the regularization parameters of the network.

ii. validation can be used to structure the architecture of the network, by helping to select size and connection properties of layers.

End Solution