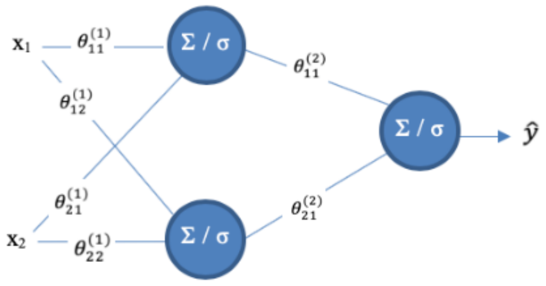
School of Computing and Information Systems
The University of Melbourne
COMP90049 Introduction to Machine Learning (Semester 1, 2021)

Week 8: Sample Solutions

1. Consider the following multilayer perceptron.



The network should implement the XOR function. Perform **one** epoch of backpropagation as introduced in the lecture on multilayer perceptron.

Notes:

- The activation function *f* for a perceptron is the *sigmoid function*:
$$f(x) = \frac{1}{1+e^{-x}}$$
- **The bias nodes are set to -1**. They are not shown in the network
- Use the following initial parameter values:

$\theta_{01}^{(1)} = 2$	$\theta_{02}^{(1)} = -1$	$\theta_{01}^{(2)} = -2$
$\theta_{11}^{(1)} = 6$	$\theta_{12}^{(1)} = 8$	$\theta_{11}^{(2)} = 6$
$\theta_{21}^{(1)} = -6$	$\theta_{22}^{(1)} = -8$	$\theta_{21}^{(2)} = -6$

- The learning rate is set to $\eta = 0.7$

i. Compute the activations of the hidden and output neurons.

Since our activation function here is the sigmoid (σ) function, in each node (neuron) we can calculate the output by applying the sigmoid function ($\sigma(x) = \frac{1}{1+e^{-x}}$) on the weighted sum (Σ) of its input (that we usually represent by $z_b^{(a)}$ where a shows the level of the neuron *(e.g., level 1, 2, 3)* and b is the index.

So, for the neuron *i* in level 1, we will have:

$$a_i^{(1)} = \sigma(z_i^{(1)}) = \sigma(\theta_i^{(1)}.X) = \sigma(\theta_{0i}^{(1)} \times x_0 + \theta_{1i}^{(1)} \times x_1 + \theta_{2i}^{(1)} \times x_2)$$

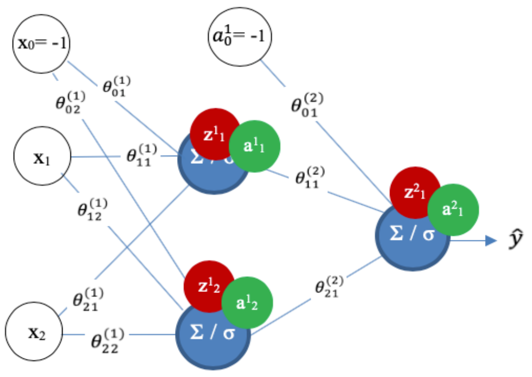
Therefore, we will have:

$$a_1^{(1)} = \sigma(\theta_{01}^{(1)} \times x_0 + \theta_{11}^{(1)} \times x_1 + \theta_{21}^{(1)} \times x_2)$$
$$= \sigma(2 \times (-1) + 6x_1 - 6x_2)$$
$$a_2^{(1)} = \sigma(\theta_{02}^{(1)} \times x_0 + \theta_{12}^{(1)} \times x_1 + \theta_{22}^{(1)} \times x_2)$$
$$= \sigma((-1) \times (-1) + 8x_1 - 8x_2)$$

Now the output of our network is simply applying the same rule on the inputs of our last neuron:

$$\hat{y} = a_1^{(2)} = \sigma\left(z_1^{(2)}\right) = \sigma\left(\theta^{(2)}.A^{(1)}\right) = \sigma\left(\theta_{01}^{(2)} \times a_0^{(1)} + \theta_{11}^{(2)} \times a_1^{(1)} + \theta_{21}^{(2)} \times a_2^{(1)}\right)$$
$$= \sigma\left((-2) \times (-1) + 6a_1^{(1)} - 6a_2^{(1)}\right)$$

You can find the schematic representation of these calculations in the following network.



ii. Compute the error of the network.

$$E = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(y - a_1^{(2)})^2$$

iii. *Backpropagate* the error to determine $\Delta\theta_{ij}$ for all weights θ_{ij} and updates the weight θ_{ij}.

In neural networks with backpropagation, we want to minimize the error of our network by finding the optimum weights (θ_{ij}) for our network. To do so we want to find the relation (dependency) between the error and the weights in each layer. Therefore, we use the derivatives of our error function.

$$\theta_{jk}^{(l)} \leftarrow \theta_{jk}^{(l)} + \Delta\theta_{jk}^{(l)}$$

$$where: \quad \Delta\theta_{jk}^{(l)} = -\eta\frac{\partial E}{\partial\theta_{jk}^{(l)}} = \eta\, \delta_k^{(l)} a_j^{(l-1)}$$

$$and \quad \delta_k^{(l)} = \underbrace{\left(1 - \sigma\left(z_k^{(l)}\right)\right)\sigma\left(z_k^{(l)}\right)}_{\sigma'\left(z_k^{(l)}\right)}\left(y - a_k^{(l)}\right) \qquad for\ the\ last\ layer$$

$$or \quad \delta_k^{(l)} = \sigma\left(z_k^{(l)}\right)\left(1 - \sigma\left(z_k^{(l)}\right)\right)\theta_{k1}^{(l+1)}\delta_1^{(l+1)} \qquad for\ the\ layer\ before$$

Now we use our first training data (1, 0) to train the model:

$$a_1^{(1)} = \sigma\left(z_1^{(1)}\right) = \sigma(6x_1 - 6x_2 - 2) = \sigma(6 \times 1 - 6 \times 0 - 2) = \sigma(4) \cong 0.982$$
$$a_2^{(1)} = \sigma\left(z_2^{(1)}\right) = \sigma(8x_1 - 8x_2 + 1) = \sigma(8 \times 1 - 8 \times 0 + 1) = \sigma(9) \cong 0.999$$
$$a_1^{(2)} = \sigma\left(z_1^{(2)}\right) = \sigma\left(2 + 6a_1^{(1)} - 6a_2^{(1)}\right) = \sigma(2 + 6 \times 0.982 - 6 \times 0.999) = \sigma(1.898)$$
$$\cong 0.8691$$

$$E = \frac{1}{2}(1 - 0.8691)^2 = 0.0086$$

x1	x2	$a_1^{(1)}$	$a_2^{(1)}$	$a_1^{(2)}$	Y (XOR)	$E(\theta)=\frac{1}{2}(y-\hat{y})^2$
1	0	$\sigma(4) = 0.982$	$\sigma(9) = 0.999$	0.8691	1	0.0086

Now we calculate the backpropagation error. Starting from the last layer, we will have:

$$\delta_1^{(2)} = \sigma\left(z_1^{(2)}\right)\left(1 - \sigma\left(z_1^{(2)}\right)\right)\left(y - a_1^{(2)}\right)$$
$$= \sigma(1.898)\left(1-\sigma(1.898)\right)(1-0.8691) = 0.8691(1-0.8691)(0.13) = 0.0149$$

$$\delta_1^{(1)} = \sigma\left(z_1^{(1)}\right)\left(1 - \sigma\left(z_1^{(1)}\right)\right)\theta_{11}^{(2)}\delta_1^{(2)}$$
$$= \sigma(4)\left(1-\sigma(4)\right) \times 6 \times 0.0147 = 0.982(1-0.982)0.0882 = 0.0016$$

$$\delta_2^{(1)} = \sigma\left(z_2^{(1)}\right)\left(1 - \sigma\left(z_2^{(1)}\right)\right)\theta_{21}^{(2)}\delta_1^{(2)}$$
$$= \sigma(9)\left(1-\sigma(9)\right) \times (-6) \times 0.0147 = 0.999(1-0.999)(-0.0882) = -0.000001103$$

Using the learning rate of η = 0.7 we can now calculate $\Delta\theta_{jk}^{(l)}$:

$$\Delta\theta_{01}^{(2)} = \eta\delta_1^{(2)}a_0^{(1)} = 0.7 \times 0.0149 \times (-1) = -0.0104$$
$$\Delta\theta_{11}^{(2)} = \eta\delta_1^{(2)}a_1^{(1)} = 0.7 \times 0.0149 \times 0.982 = 0.0102$$
$$\Delta\theta_{21}^{(2)} = \eta\delta_1^{(2)}a_2^{(1)} = 0.7 \times 0.0149 \times 0.999 = 0.0104$$

$$\Delta\theta_{01}^{(1)} = \eta\delta_1^{(1)}x_0 = 0.7 \times 0.0016 \times (-1) = -0.0011$$
$$\Delta\theta_{11}^{(1)} = \eta\delta_1^{(1)}x_1 = 0.7 \times 0.0016 \times 1 = 0.0011$$
$$\Delta\theta_{21}^{(1)} = \eta\delta_1^{(1)}x_2 = 0.7 \times 0.0016 \times 0 = 0$$

$$\Delta\theta_{02}^{(1)} = \eta\delta_2^{(1)}x_0 = 0.7 \times 0.000001103 \times (-1) \cong 0$$
$$\Delta\theta_{12}^{(1)} = \eta\delta_2^{(1)}x_1 = 0.7 \times 0.000001103 \times 1 \cong 0$$
$$\Delta\theta_{22}^{(1)} = \eta\delta_2^{(1)}x_2 = 0.7 \times 0.000001103 \times 0 = 0$$

Based on these results we can update the network weights:

$$\theta_{01}^{(2)} = \theta_{01}^{(2)} + \Delta\theta_{01}^{(2)} = -2 + (-0.0104) = -2.0104$$
$$\theta_{11}^{(2)} = \theta_{11}^{(2)} + \Delta\theta_{11}^{(2)} = 6 + 0.0102 = 6.0102$$
$$\theta_{21}^{(2)} = \theta_{21}^{(2)} + \Delta\theta_{21}^{(2)} = -6 + 0.0104 = -5.9896$$

$$\theta_{01}^{(1)} = \theta_{01}^{(1)} + \Delta\theta_{01}^{(1)} = 2 + (-0.0011) = 1.9989$$
$$\theta_{11}^{(1)} = \theta_{11}^{(1)} + \Delta\theta_{11}^{(1)} = 6 + 0.0011 = 6.0011$$

The rest of the weights do not change i.e.

$$\theta_{02}^{(1)} = -1,\ \ \theta_{21}^{(1)} = -6,\ \ \theta_{22}^{(1)} = -8,\ \ \Delta\theta_{12}^{(1)} = 8$$

Now we use our next training instance (0,1):

$$a_1^{(1)} = \sigma\left(z_1^{(1)}\right) = \sigma(1.9989 + 6.00112x_1 - 6x_2) = \sigma(6.00112 \times 0 - 6 \times 1 + 1.9989)$$
$$= \sigma(-7.9989) \cong 3.387e - 4$$

$$a_2^{(1)} = \sigma\left(z_2^{(1)}\right) = \sigma(-1 + 8x_1 - 8x_2) = \sigma(8 \times 0 - 8 \times 1 + 1) = \sigma(-7) \cong 9.11e - 4$$

$$a_1^{(2)} = \sigma\left(z_1^{(2)}\right) = \sigma\left(-2.01029 + 6.0101 \times \sigma(-7.9989) - 5.98972 \times \sigma(-7)\right) = \sigma(2.007)$$
$$= 0.8815$$

$$E = \frac{1}{2}(1 - 0.8815)^2 = 0.007$$

x1	x2	$a_1^{(1)}$	$a_2^{(1)}$	$a_1^{(2)}$	Y (XOR)	E(θ)= $\frac{1}{2}(y-\hat{y})^2$
1	0	$\sigma(4)$	$\sigma(9)$	0.8691	1	0.0086
0	1	$\sigma(-7.9989)$	$\sigma(-7)$	0.8815	1	0.007

To calculate the backpropagation error, we will have[1]:

$$\delta_1^{(2)} = \sigma(z_1^{(2)})\left(1-\sigma(z_1^{(2)})\right)(y-a_1^{(2)})$$
$$= \sigma(2.007)(1-\sigma(2.007))(1-0.8815) = 0.0124$$
$$\delta_1^{(1)} = \sigma(z_1^{(1)})\left(1-\sigma(z_1^{(1)})\right)\theta_{11}^{(2)}\delta_1^{(2)}$$
$$= \sigma(-7.9989)(1-\sigma(-7.9989)) \times 6.0102 \times 0.0124 \cong 0$$
$$\delta_2^{(1)} = \sigma(z_2^{(1)})\left(1-\sigma(z_2^{(1)})\right)\theta_{21}^{(2)}\delta_1^{(2)}$$
$$= \sigma(-7)(1-\sigma(-7)) \times (-5.9896) \times 0.0124 \cong 0$$

$$\Delta\theta_{01}^{(2)} = \eta\delta_1^{(2)}a_0^{(1)} = 0.7 \times 0.0124 \times (-1) = -0.0087$$
$$\Delta\theta_{11}^{(2)} = \eta\delta_1^{(2)}a_1^{(1)} = 0.7 \times 0.0124 \times 3.387e-4 \cong 0$$
$$\Delta\theta_{21}^{(2)} = \eta\delta_1^{(2)}a_2^{(1)} = 0.7 \times 0.0124 \times 9.11e-4 \cong 0$$
$$\Delta\theta_{01}^{(1)} = \eta\delta_1^{(1)}x_0 = 0.7 \times (2.496\,e-5) \times (-1) \cong 0$$
$$\Delta\theta_{11}^{(1)} = \eta\delta_1^{(1)}x_1 = 0.7 \times (2.496\,e-5) \times 0 = 0$$
$$\Delta\theta_{21}^{(1)} = \eta\delta_1^{(1)}x_2 = 0.7 \times (2.496\,e-5) \times 1 \cong 0$$

$$\Delta\theta_{02}^{(1)} = \eta\delta_2^{(1)}x_0 = 0.7 \times (-6.7454\,e-5) \times (-1) \cong 0$$
$$\Delta\theta_{12}^{(1)} = \eta\delta_2^{(1)}x_1 = 0.7 \times (-6.7454\,e-5) \times 0 = 0$$
$$\Delta\theta_{22}^{(1)} = \eta\delta_2^{(1)}x_2 = 0.7 \times (-6.7454\,e-5) \times 1 \cong 0$$

Based on these results we can update the network weights:

$$\theta_{01}^{(2)} = \theta_{01}^{(2)} + \Delta\theta_{01}^{(2)} = -2.01029 + (-0.0087) \cong -2.0191$$

Since the other values are very small, the rest of the weights do not change.

Now we use our next training instance (1, 1):

$$a_1^{(1)} = \sigma(z_1^{(1)}) = \sigma(6.0102x_1 - 5.9896x_2 + 2.0191)$$
$$= \sigma(6.0102 \times 1 - 5.9896 \times 1 + 2.0191) = \sigma(-1.9978)$$
$$a_2^{(1)} = \sigma(z_2^{(1)}) = \sigma(8x_1 - 8x_2 + 0.9999) = \sigma(8 \times 1 - 8 \times 1 + 0.9999) = \sigma(0.9999)$$
$$a_1^{(2)} = \sigma(z_1^{(2)}) = \sigma\left(2.0191 + 6.0102 \times \sigma(-1.9978) - 5.9896 \times \sigma(0.9999)\right)$$
$$= \sigma(-1.6416) = 0.1622$$

$$E = \frac{1}{2}(0 - 0.1622)^2 = 0.0132$$

[1] In this assignment, we make the simplifying assumption that any update < 1e-4 ≅ 0 to keep the computations feasible.

x1	x2	$a_1^{(1)}$	$a_2^{(1)}$	$a_1^{(2)}$	Y (XOR)	E(θ)=$\frac{1}{2}(y-\hat{y})^2$
1	0	$\sigma(4)$	$\sigma(9)$	0.8691	1	0.0086
0	1	$\sigma(-7.9989)$	$\sigma(-7)$	0.8815	1	0.007
1	1	$\sigma(-1.9978)$	$\sigma(0.9999)$	0.1622	0	0.0132

To calculate the backpropagation error, we will have:

$$\delta_1^{(2)} = \sigma(z_1^{(2)})\left(1 - \sigma(z_1^{(2)})\right)(y - a_1^{(2)}) = -0.0221$$
$$\delta_1^{(1)} = \sigma(z_1^{(1)})\left(1 - \sigma(z_1^{(1)})\right)\theta_{11}^{(2)}\delta_1^{(2)} = -0.0139$$
$$\delta_2^{(1)} = \sigma(z_2^{(1)})\left(1 - \sigma(z_2^{(1)})\right)\theta_{21}^{(2)}\delta_1^{(2)} = 0.0260$$

$$\Delta\theta_{01}^{(2)} = \eta\delta_1^{(2)}a_0^{(1)} = 0.0154$$
$$\Delta\theta_{11}^{(2)} = \eta\delta_1^{(2)}a_1^{(1)} = -0.0018$$
$$\Delta\theta_{21}^{(2)} = \eta\delta_1^{(2)}a_2^{(1)} = -0.0113$$

$$\Delta\theta_{01}^{(1)} = \eta\delta_1^{(1)}x_0 = -0.0098$$
$$\Delta\theta_{11}^{(1)} = \eta\delta_1^{(1)}x_1 = -0.0098$$
$$\Delta\theta_{21}^{(1)} = \eta\delta_1^{(1)}x_2 = 0.0182$$

$$\Delta\theta_{02}^{(1)} = \eta\delta_2^{(1)}x_0 = 0.0182$$
$$\Delta\theta_{12}^{(1)} = \eta\delta_2^{(1)}x_1 = -0.0098$$
$$\Delta\theta_{22}^{(1)} = \eta\delta_2^{(1)}x_2 = 0.0182$$

Based on these results we can update the network weights:

$$\theta_{01}^{(2)} = \theta_{01}^{(2)} + \Delta\theta_{01}^{(2)} = -2.0037$$
$$\theta_{11}^{(2)} = \theta_{11}^{(2)} + \Delta\theta_{11}^{(2)} = 6.0084$$
$$\theta_{21}^{(2)} = \theta_{21}^{(2)} + \Delta\theta_{21}^{(2)} = -6.0009$$

$$\theta_{01}^{(1)} = \theta_{01}^{(1)} + \Delta\theta_{01}^{(1)} = 2.0086$$
$$\theta_{11}^{(1)} = \theta_{11}^{(1)} + \Delta\theta_{11}^{(1)} = 5.9913$$
$$\theta_{21}^{(1)} = \theta_{21}^{(1)} + \Delta\theta_{21}^{(1)} = -6.0097$$

$$\theta_{02}^{(1)} = \theta_{02}^{(1)} + \Delta\theta_{02}^{(1)} = -1.0181$$
$$\theta_{12}^{(1)} = \theta_{12}^{(1)} + \Delta\theta_{12}^{(1)} = 8.0182$$
$$\theta_{22}^{(1)} = \theta_{22}^{(1)} + \Delta\theta_{22}^{(1)} = -7.9819$$

Now we use our next training instance (0, 0):

$$a_1^{(1)} = \sigma(z_1^{(1)}) = \sigma(5.9913x_1 - 6.00097x_2 + 2.0037)$$
$$= \sigma(5.9913 \times 0 - 6.00097 \times 0 + 2.0086) = \sigma(2.0086)$$
$$a_2^{(1)} = \sigma(z_2^{(1)}) = \sigma(8.0182x_1 - 7.9819x_2 + 1.0181) = \sigma(1.0181)$$
$$a_1^{(2)} = \sigma(z_1^{(2)}) = \sigma\left(2.0037 + 6.0084 \times \sigma(2.0086) - 6.0009 \times \sigma(1.0181)\right)$$
$$= \sigma(-1.6938) = 0.1553$$

$$E = \frac{1}{2}(\,00.1553)^2 = 0.0121$$

x1	x2	$a_1^{(1)}$	$a_2^{(1)}$	$a_1^{(2)}$	Y (XOR)	E(θ)=$\frac{1}{2}(y-\hat{y})^2$
1	0	$\sigma(4)$	$\sigma(9)$	0.8691	1	0.0086
0	1	$\sigma(-7.9989)$	$\sigma(-7)$	0.8815	1	0.007
1	1	$\sigma(-1.9978)$	$\sigma(0.9999)$	0.1622	0	0.0132
0	0	$\sigma(-2.0086)$	$\sigma(1.0181)$	0.1553	0	0.0121

To calculate the backpropagation error, we will have:

$$\delta_1^{(2)} = \sigma\left(z_1^{(2)}\right)\left(1-\sigma\left(z_1^{(2)}\right)\right)\left(y-a_1^{(2)}\right) = -0.0204$$
$$\delta_1^{(1)} = \sigma\left(z_1^{(1)}\right)\left(1-\sigma\left(z_1^{(1)}\right)\right)\theta_{11}^{(2)}\delta_1^{(2)} = -0.0128$$
$$\delta_2^{(1)} = \sigma\left(z_2^{(1)}\right)\left(1-\sigma\left(z_2^{(1)}\right)\right)\theta_{21}^{(2)}\delta_1^{(2)} = 0.0238$$

$$\Delta\theta_{01}^{(2)} = \eta\delta_1^{(2)}a_0^{(1)} = 0.0143$$
$$\Delta\theta_{11}^{(2)} = \eta\delta_1^{(2)}a_1^{(1)} = -0.0017$$
$$\Delta\theta_{21}^{(2)} = \eta\delta_1^{(2)}a_2^{(1)} = -0.0105$$

$$\Delta\theta_{01}^{(1)} = \eta\delta_1^{(1)}x_0 = -0.0098$$
$$\Delta\theta_{11}^{(1)} = \eta\delta_1^{(1)}x_1 = 0$$
$$\Delta\theta_{21}^{(1)} = \eta\delta_1^{(1)}x_2 = 0$$

$$\Delta\theta_{02}^{(1)} = \eta\delta_2^{(1)}x_0 = 0.0167$$
$$\Delta\theta_{12}^{(1)} = \eta\delta_2^{(1)}x_1 = 0$$
$$\Delta\theta_{22}^{(1)} = \eta\delta_2^{(1)}x_2 = 0$$

Based on these results we can update the network weights:

$$\theta_{01}^{(2)} = \theta_{01}^{(2)} + \Delta\theta_{01}^{(2)} = -2.0176$$
$$\theta_{11}^{(2)} = \theta_{11}^{(2)} + \Delta\theta_{11}^{(2)} = 6.0067$$
$$\theta_{21}^{(2)} = \theta_{21}^{(2)} + \Delta\theta_{21}^{(2)} = -6.0113$$

$$\theta_{01}^{(1)} = \theta_{01}^{(1)} + \Delta\theta_{01}^{(1)} = 2.0176$$
$$\theta_{11}^{(1)} = \theta_{11}^{(1)} + \Delta\theta_{11}^{(1)} = 5.9913$$
$$\theta_{21}^{(1)} = \theta_{21}^{(1)} + \Delta\theta_{21}^{(1)} = -6.0097$$

$$\theta_{02}^{(1)} = \theta_{02}^{(1)} + \Delta\theta_{02}^{(1)} = -1.0348$$
$$\theta_{12}^{(1)} = \theta_{12}^{(1)} + \Delta\theta_{12}^{(1)} = 8.0182$$
$$\theta_{22}^{(1)} = \theta_{22}^{(1)} + \Delta\theta_{22}^{(1)} = -7.9819$$

And this would be the end of epoch one! ☺

2. What is the difference between "model bias" and "model variance"?

Model Bias:

- Model bias is the propensity of a classifier to systematically produce the same errors; if it doesn't produce errors, it is unbiased; if it produces different kinds of errors on different instances, it is also unbiased. (An example of the latter: the instance is truly of class A, but sometimes the system calls it B and sometimes the system calls it C.)
- Note that this makes more sense in a regression context, where we can sensibly measure the difference between the prediction and the true value. In a classification context, these can only be "same" or "different".
- Consequently, a typical interpretation of bias in a classification context is whether the classifier labels the test data in such a way that the distribution of predicted classes systematically doesn't match the distribution of actual classes. For example, "bias towards the majority class", when the model predicts too many instances as the majority class.

Model variance is the propensity of a classifier to produce different classifications using different training sets (randomly sampled from the same population). It is a measure of the inconsistency of the classifier, from training set to training set.

(i). Why is a high bias, low variance classifier undesirable?

In short, because it's consistently wrong. Using the other interpretation: the distribution of labels predicted by the classifier is consistently different to the distribution of the true labels; this means that it must be making mistakes.

(ii). Why is a low bias, high variance classifier (usually) undesirable?

This is less obvious – it's low bias, so that it must be making a bunch of correct decisions. The fact that its high variance means that not all of the predictions can possibly be correct (or it would be low-variance!) — and the correct predictions will change, perhaps drastically, as we change the training data.

One obvious problem here is that it's difficult to be certain about the performance of the classifier at all: we might estimate its error rate to be low on one set of data, and high on another set of data.

The real issue becomes more obvious when we consider the alternative formulation: the low bias means that the distribution of predictions matches the distribution of true labels; however, the high variance means that which instances are getting assigned to which label must be changing every time.

This suggests the real problem — namely, that what we have is the second kind of unbiased classifier: one that makes different kinds of errors on different training sets, but always errors; and not the first kind: one that is usually correct.

3. Describe how validation set, and cross-validation can help reduce overfitting?

Machine learning models usually have one or more (hyper)parameters that control for model complexity, the ability of model to fit noise in the training set. In a practical application, we need to determine the values of such parameters, and the principal objective in doing so is usually to achieve the best predictive performance on new data. Furthermore, as well as finding the appropriate values for complexity parameters within a given model, we may wish to consider a range of different types of model in order to find the best one for our particular application.

We know that the performance on *training data* is not a good indicator of predictive performance on unseen data because of overfitting. If data is plentiful, then one approach is simply to use some of the available data to train a range of models, or a given model with a range of values for its complexity parameters, and then to compare them on independent data, sometimes called a *validation set*, and select the one having the best predictive performance. If the model design is iterated many times using a limited size data set, then some overfitting to the validation data can occur and so it may be necessary to keep aside a *third test set* on which the performance of the selected model is finally evaluated.

In many applications, however, the supply of data for training and testing will be limited, and in order to build good models, we wish to use as much of the available data as possible for training. However, if the validation set is small, it will give a relatively noisy estimate of predictive performance. One solution to this dilemma is to use *cross-validation*.