# 50.021 Artificial Intelligence
## Theory Bonus Homework

**Materials from week 1 - 6**

1. Assume we have trained several classifiers to learn the label $Y$ from input feature vector $X \in \mathbb{R}^3$. Do the following classifiers contain sufficient information to calculate $P(X_1, X_2, X_3, Y)$? State yes/no and give a brief reason.

   (a) Logistic Regression
   **Solution:**
   No, logistic regression computes $P(Y|X_1, X_2, X_3)$

   (b) Linear Regresion
   **Solution:**
   No, similarly, linear regression also computes $P(Y|X_1, X_2, X_3)$

2. We trained a few samples from two different populations, each with the same logistic regression model:

$$P(y = 1|x, w) = \sigma(w_0 + w_1 x)$$

where $\sigma$ is a sigmoid function. We end up with two logistic regression models shown in figure 1. Answer the following questions.

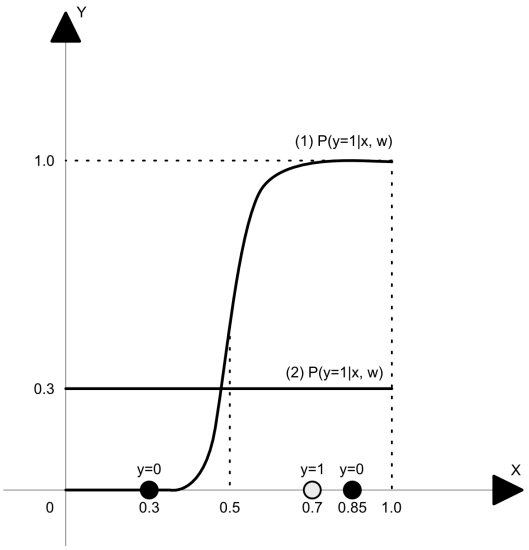


Figure 1: Plots of two classification models and three training samples

(a) How many classification errors are made by model (1)? Which point(s) is/are wrongly classified?
**Solution:**
Model 1 makes 1 classification error at $x = 0.85$.

(b) How many classification errors are made by model (2)? Which point(s) is/are wrongly classified?
**Solution:**
Model 2 makes 1 classification error at $x = 0.7$.

(c) Which model, (1) or (2) is most likely to be the result of training using the samples shown in figure 1? i.e. it maximises the joint probability $P(y = 0|x = 0.3, w) \cdot P(y = 1|x = 0.7, w) \cdot P(y = 0|x = 0.85, w)$.
**Solution:**
Model 2. Because for model 1, $P(y = 0|x = 0.85, w)$ is near zero.

3. In class, we learn that the logistic regression model takes the form of logistic function,

$$h(x) = \frac{1}{1 + e^{-\boldsymbol{w}\cdot\boldsymbol{x}+b}}$$

where $h(x)$ can be interpreted for 2 classes : $P(Y = 1|X = \boldsymbol{x})$ or $P(Y = 0|X = \boldsymbol{x})$.

Consider a bivariate Gaussian distribution, with the parameters $\mu_1, \mu_2, \sigma_1, \sigma_2, \rho$. Its probability density is,

$$P(X_1, X_2|Y = k) = \frac{1}{2\pi\sigma_1\sigma 2\sqrt{1-\rho^2}} e^{-\frac{\sigma_2^2(X_1-\mu_{1k})^2+\sigma_1^2(X_2-\mu_{2k})^2-2\rho\sigma_1\sigma_2(X1-\mu_{1k})(X_2-\mu_{2k})}{2(1-\rho^2)\sigma_1^2\sigma_2^2}},$$

where $K \in 0, 1$, and

$$P(Y = 1) = \pi, P(Y = 0) = 1 - \pi$$

Note that only $\mu_{1k}, \mu_{2k}$ depend on the value $Y$, and $\sigma_1, \sigma_2$ and correlation $\rho$ do not depend on the value $Y$. $X_1$ and $X_2$ are **not** conditionally independent given $Y$.

With the above density $P(X_1, X_2|Y = k)$, can we still use logistic regression method to find the parameters that reflect the maximum likelihood function?
*Hints*:

(a) Show that it is possible to write $P(X_1, X_2|Y = k)$ in the form of

$$h(x) = \frac{1}{1 + e^{-\boldsymbol{w}\cdot\boldsymbol{x}+b}}$$

(b) In oder to do that, we first show that $P(Y = K|X_1, X_2)$ can be written by Bayes rule in terms of $P(Y = K)$ and $P(X_1, X_2|Y = K)$.

(c) It will be a complicated equation so just write them in terms of probability first

(d) Rewrite the quotient term into the shape of $\frac{1}{1+T}$ where T is a term that depends on $P(Y = K)$ and $P(X_1, X_2|Y)$

(e) Rewrite $\frac{1}{1+T}$ as $\frac{1}{1+e^{-U}}$

(f) Now plug in the definition of the probabilities into $U$ and show that $U$ is affine in $X_1, X_2$,

**Solution:**

$$\begin{aligned}
P(Y = 1|X_1, X_2) &= \frac{P(Y = 1)P(X_1, X_2|Y = 1)}{P(Y = 1)P(X_1, X_2|Y = 1) + P(Y = 0)P(X_1, X_2|Y = 0)} \\
&= \frac{1}{1 + \frac{P(Y=0)P(X_1,X_2|Y=0)}{P(Y=1)P(X_1,X_2|Y=1)}} \\
&= \frac{1}{1 + \exp\left(\ln \frac{P(Y=0)P(X_1,X_2|Y=0)}{P(Y=1)P(X_1,X_2|Y=1)}\right)} \\
&= \frac{1}{1 + \exp\left(\ln \frac{1-\pi}{\pi} + \ln \frac{P(X_1,X_2|Y=0)}{P(X_1,X_2|Y=1)}\right)}
\end{aligned}$$

Since only $\mu_1, \mu_2$ depend on $Y$ and we can cancel out the independent terms (the first term before the exp in the bivariate Gaussian distribution above), we can arrive at the simplified expression,

$$\ln \frac{P(X_1, X_2|Y=0)}{P(X_1, X_2|Y=1)} = \frac{\sigma_2^2(X_1-\mu_{11})^2 + \sigma_1^2(X_2-\mu_{21})^2 - 2\rho\sigma_1\sigma_2(X1-\mu_{11})(X_2-\mu_{21})}{2(1-\rho^2)\sigma_1^2\sigma_2^2} - \frac{\sigma_2^2(X_1-\mu_{10})^2 + \sigma_1^2(X_2-\mu_{20})^2 - 2\rho\sigma_1\sigma_2(X1-\mu_{10})(X_2-\mu_{20})}{2(1-\rho^2)\sigma_1^2\sigma_2^2}$$

And then by expanding the quadratic terms, we can find all of them cancel out, leaving only $X_1, X_2$ terms,

$$\begin{aligned}\ln \frac{P(X_1, X_2|Y=0)}{P(X_1, X_2|Y=1)} &= \frac{2\sigma_2^2(\mu_{10}-\mu_{11}) + 2\rho\sigma_1\sigma_2(\mu_{21}-\mu_{20})}{2(1-\rho^2)\sigma_1^2\sigma_2^2}X_1 \\ &+ \frac{2\sigma_1^2(\mu_{20}-\mu_{21}) + 2\rho\sigma_1\sigma_2(\mu_{11}-\mu_{10})}{2(1-\rho^2)\sigma_1^2\sigma_2^2}X_2 \\ &+ \frac{\sigma_2^2(\mu_{11}^2-\mu_{10}^2) + \sigma_1^2(\mu_{21}^2-\mu_{20}^2) + +2\rho\sigma_1\sigma_2(\mu_{10}\mu_{20}-\mu_{11}\mu_{21})}{2(1-\rho^2)\sigma_1^2\sigma_2^2}\end{aligned}$$

So, the answer is **yes**, substituting the expression above back to $P(Y=1|X_1, X_2)$ gives the form represented by logistic regression $h(x)$.

4. In this question, we are extending the binary logistic regression to handle multi-class classification. Lets assume we have $K$ classes. The posterior probability for class $k$ is given by:

$$P(Y = k|X = \boldsymbol{x}) = \frac{e^{\boldsymbol{w}_k^T \boldsymbol{x}}}{1 + \sum_{t=1}^{K-1} e^{\boldsymbol{w}_t^T \boldsymbol{x}}} \text{ for } k = 1, ..., K-1$$

$$P(Y = K|X = \boldsymbol{x}) = \frac{1}{1 + \sum_{t=1}^{K-1} e^{\boldsymbol{w}_t^T \boldsymbol{x}}}$$

where $\boldsymbol{x}, \boldsymbol{w}_t \in \mathbb{R}^n$. To simplify the expression, assume there's no bias involved. Answer the following questions.

(a) Assume we are given $N$ training samples, each with $n$ dimension. Derive the expression for the log likelihood, $L(\boldsymbol{w}_1, ..., \boldsymbol{w}_K)$.
*Hint: you can use an indicator function $I_{ik}$, where $I_{ik} = 1$ if $Y^i$ true label is $k$ and $I_{ik} = 0$ if otherwise, for all $i = 1, ..., N$.* Show your steps.
Recall that in the binary case,

$$\begin{aligned} P(Y = y|X = x) &= P(Y = +1|X = x)^y \cdot P(Y = 0|X = x)^{1-y} \\ &= h(x)^y (1 - h(x))^{1-y} \\ E(\boldsymbol{w}) &= \prod_{i=1}^{N} P(Y^i = 1|\boldsymbol{X} = \boldsymbol{x}^i; \boldsymbol{w})^{I_{i1}} P(Y^i = 0|\boldsymbol{X} = \boldsymbol{x}^i; \boldsymbol{w})^{I_{i0}} \\ &= \prod_{i=1}^{N} h(x_i)^{y_i} (1 - h(x_i))^{1-y_i} \end{aligned}$$

Notice that $h(x_i)$ is zero when $y_i = 0$, and otherwise. Extend this to 3 classes first, and then generalize it to $K$ classes.
**Solution:**
Using the indicator function and extending it to multiclass,

$$\begin{aligned} E(\boldsymbol{w}_1, ..., \boldsymbol{w}_{K-1}) &= \prod_{i=1}^{N} \prod_{k=1}^{K} P(Y^i = k|\boldsymbol{X} = \boldsymbol{x}^i; \boldsymbol{w}_1, ..., \boldsymbol{w}_{K-1})^{I_{ik}} \\ &= \prod_{i=1}^{N} \prod_{k=1}^{K} \left( \frac{e^{\boldsymbol{w}_k^T \boldsymbol{x}^i}}{1 + \sum_{t=1}^{K-1} e^{\boldsymbol{w}_k^T \boldsymbol{x}^i}} \right)^{I_{ik}} \end{aligned}$$

Taking logarithm,

$$\log E(\boldsymbol{w}_1, ..., \boldsymbol{w}_{K-1}) = \sum_{i=1}^{N} \sum_{k=1}^{K} \left( I_{ik} \left[ \boldsymbol{w}_k^T \boldsymbol{x}^i - \log \left( 1 + \sum_{t=1}^{K-1} e^{\boldsymbol{w}_t^T \boldsymbol{x}_i} \right) \right] \right)$$

(b) Since the maximization of the expression in part (a) doesn't have a closed-form solution, derive the expression for the partial derivative of $\log E(\boldsymbol{w}_1, ..., \boldsymbol{w}_K)$ with respect to $w_k$, $k = 1, ..., K-1$.
**Solution:**

$$\frac{\partial \log E(\boldsymbol{w}_1, ...\boldsymbol{w}_{K-1})}{\partial \boldsymbol{w}_k} = \sum_{i=1}^{N} \left[ I_{ik} - \frac{e^{\boldsymbol{w}_k^T \boldsymbol{x}_i}}{1 + \sum_{t=1}^{K-1} e^{w_t^T \boldsymbol{x}_i}} \right] \boldsymbol{x}_i$$

(c) Now we want to solve the maximization problem above with gradient descent method. Assume that the weights at time step $T = 0$ is 0 for all classes. Write the update rule for $w_k$, using $\eta$ as a symbol for stepsize.

**Solution:** Since we use gradient descent, we need to minimize the *negative* log likelihood of of $E(\boldsymbol{w}_1, ...\boldsymbol{w}_{K-1})$.
$\boldsymbol{w}_k^{T+1} \leftarrow \boldsymbol{w}_k^T - \eta \left( \frac{\partial - \log E(\boldsymbol{w}_1, ...\boldsymbol{w}_{K-1})}{\partial \boldsymbol{w}_k} \right)$ where the expression for the partial derivative is as per part (b).
The final answer is,

$$\boldsymbol{w}_k^{T+1} \leftarrow \boldsymbol{w}_k^T + \eta \sum_{i=1}^{N} \left[ I_{ik} - \frac{e^{\boldsymbol{w}_k^T \boldsymbol{x}_i}}{1 + \sum_{t=1}^{K-1} e^{w_t^T \boldsymbol{x}_i}} \right] \boldsymbol{x}_i$$

This can be further simplified to be,

$$\boldsymbol{w}_k^{T+1} \leftarrow \boldsymbol{w}_k^T + \eta \sum_{i=1}^{N} \left[ I_{ik} - P(y^i = k | \boldsymbol{x}^i) \right] \boldsymbol{x}_i$$

(d) Will the solution converge to a global minimum?
**Solution:** Yes since it is a negative concave (convex) function.

5. Consider a neural network that has only these two types of activation function,
   (1) Identity function, $g_I(x) = x$
   (2) Step function, $g_s(x) = 1$, if $x >= 1$, 0 otherwise.

   (a) Draw a neural network structure with 1 input $x$, 1 hidden layer with 3 neurons, and 1 output $y$, that has a response of $y = c$ if $x < a$, 0 otherwise. Specify the activation function for each neuron, and the weights /bias values.
   **Solution:**
   This is a network with equation: $g_I(c - \sum_{i=1}^{n} \frac{1}{3} c \cdot g_s^i(x - a))$

   (b) Now draw another neural network with 1 input $x$, 1 hidden layer with 4 neurons, and 1 output $y$ with a response of $y = c$ if $x \in [a, b)$, 0 otherwise. Specify the activation function for each neuron, and the weights / bias values.
   **Solution:**
   This is a network with equation: $g_I[\sum_{i=1}^{2} \frac{1}{2} c \cdot g_s^i(x - a) - \sum_{i=1}^{2} \frac{1}{2} c \cdot g_s^i(x - b)]$

6. Recall that linear regression can be made more powerful with the help of some basis functions $\phi(\boldsymbol{x})$. Assume that $\boldsymbol{x} \in \mathbb{R}^3$, hence $\phi(\boldsymbol{x}) = \{\phi_1(x_1), \phi_2(x_2), \phi_3(x_3)\}$ This is equivalent to fitting a feed-forward neural network with one hidden layer, as shown in Figure 2, where a set of weights is **fixed**. In not more than 5 sentences, briefly explain how linear regression with basis function can be represented with the neural network in Figure 2. E.g: what the first layer of weights, hidden layer, and the second layer of weights correspond to in a linear regression with basis function? Which sets of weights are held fixed?

**Solution:**
The first layer of weights correspond to the parameters used in computing the feature vectors to another basis. This weight is kept fixed.The hidden layer is the feature vector $\phi(\boldsymbol{x})$. The second layer of weights are the regression weights.

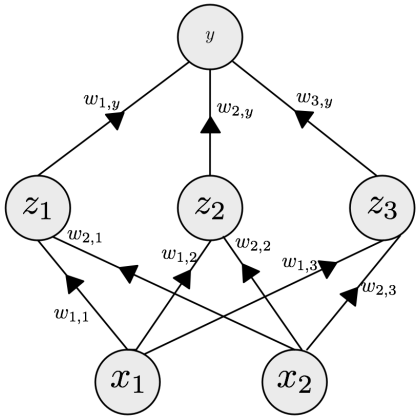


Figure 2: Feed-forward Neural Net

7. Consider the RNN in figure 3, where

$$y^t = w_3 z^t$$

$$z^t = \sigma(w_2 z^{t-1} + w_1 x^t)$$

and $\sigma(x)$ is the logistic function $\frac{1}{1+e^{-x}}$. Suppose we want to train this network using gradient

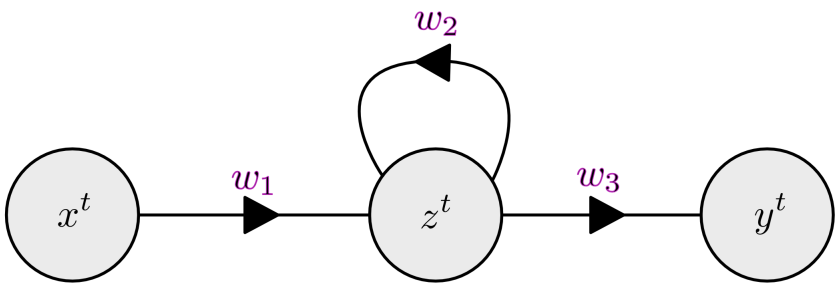


Figure 3: A Simple RNN

descent to fit the input/output time series of length 3: $(x^1, y^1), (x^2, y^2), (x^3, y^3)$. Obtain an expression for the change in weights $w_3$ and $w_2$ (you need not do $w_1$) with respect to the error function, $E = \sum_{t=1}^{3}(y^t - \hat{y}^t)^2$, where $\hat{y}^t$ is the true output at time $t$. Assume that $z_0 = 0$.

*Hint: $z^3$ depends on $z^2$ depends on $z^1$. So any derivative with respect to $z^t$ cannot be treated as a constant and we have to apply chain rule. To find the total gradient, we **sum** up the contributions at **each** time step, because the weights are shared across the the network.*

**Solution:**

$$\begin{aligned}
\frac{\partial E}{\partial y^t} &= 2(y^t - \hat{y}^t), \forall t = 1, 2, 3 \\
\frac{\partial y^t}{\partial z^t} &= w_3, \forall t = 1, 2, 3 \\
\frac{\partial z^t}{\partial z^{t-1}} &= \sigma(w_2 z^{t-1} + w_1 x^t)(1 - \sigma(w_2 z^{t-1} + w_1 x^t))w_2, \forall t = 1, 2, 3 \\
\frac{\partial z^t}{\partial w_2} &= z^t(1 - z^t)z^{t-1} \\
\frac{\partial E}{\partial w_3} &= \sum_{t=1}^{3} \frac{\partial E}{\partial y^t}\frac{\partial y^t}{\partial w_3} \\
&= \sum_{t=1}^{3} 2(y^t - \hat{y}^t)z^t \\
\frac{\partial E}{\partial w_2} &= \frac{\partial E}{\partial y^3}\frac{\partial y^3}{\partial z^3}\frac{\partial z^3}{\partial w_2} + \frac{\partial E}{\partial y^3}\frac{\partial y^3}{\partial z^3}\frac{\partial z^3}{\partial z^2}\frac{\partial z^2}{\partial w_2} + \frac{\partial E}{\partial y^3}\frac{\partial y^3}{\partial z^3}\frac{\partial z^3}{\partial z^2}\frac{\partial z^2}{\partial z^1}\frac{\partial z^1}{\partial w_2} \\
&\quad + \frac{\partial E}{\partial y^2}\frac{\partial y^2}{\partial z^2}\frac{\partial z^2}{\partial w_2} + \frac{\partial E}{\partial y^2}\frac{\partial y^2}{\partial z^2}\frac{\partial z^2}{\partial z^1}\frac{\partial z^1}{\partial w_2} \\
&\quad + \frac{\partial E}{\partial y^1}\frac{\partial y^1}{\partial z^1}\frac{\partial z^1}{\partial w_2}
\end{aligned}$$

8. Consider a layer in CNN that takes in a single channel input of $100 \times 100$, and has 100 filters. In each of the following cases, compute the number of parameters that are learned in this layer. We assume that bias is present for each weight.

   (a) A convolution layer with filters of same size as the input.
   **Solution:** There are $100 \times 100$ weights per filter, and 100 filters since the output is 100 neurons. There are also 100 bias. Total parameters is $100^3 + 100$.

   (b) A convolution layer with $10 \times 10$ filters with stride of 5
   **Solution:** There are $(10 \times 10 \times 100 + 100)$ parameters.

   (c) A convolution layer with $1 \times 1$ filter and a stride of 1
   **Solution:** There are $(1 \times 100 + 100)$ parameters

9. Indicate whether the following statements about CNN is true or false.

   (a) There are no parameters learned in the ReLU layer [T/F]
   **Solution:** True

   (b) If we use a convolution filter with larger size (more parameters), it leads to a more complex model which is more difficult to train [T/F]
   **Solution:** True

   (c) Too many parameters cause overfitting [T/F]
   **Solution:** True

   (d) Overfitting is more likely if we have less training data [T/F]
   **Solution:** True

   (e) The best objective function for *classification* problem is sum-squared error [T/F]
   **Solution:** False

   (f) We augment the training data to allow the neural nets to see multiple views of the same sample, hence making it more robust to noise [T/F]
   **Solution:** True

   (g) Weights are shared in the convolution layer, but biases are not. [T/F]
   **Solution:** False