CpE 520: HW#13

West Virginia University

Ali Zafari

Question 1

Part (a): Inverse of $\mathbf{y_{n\times 1} = A_{n\times m} x_{m\times 1}}$ where n¡m and rank=n

In this case there are more unknowns than equations so the system is underdetermined. Because the matrix A is full rank with $rank = n$, AA^T will be an $n \times n$ non-singular square matrix. Therefore, there will be an inverse for it like below:

$$(AA^T)(AA^T)^{-1} = I \qquad (*)$$

As it is clear from the above equation, we may use $A^T(AA^T)^{-1}$ as a right inverse for matrix A. We first assume that the inverse we are seeking for has an equation like $\mathbf{x_{m\times 1} = R_{m\times n} y_{n\times 1}}$. Finding $R_{m\times n}$ will be the goal. We propose $R_{m\times n} = A^T(AA^T)^{-1}$ and through the equations below we will check if it will result in our main equation:

$$\boxed{\mathbf{x = A^T(AA^T)^{-1}y}}$$

$$\begin{aligned} &\mathbf{x = A^T(AA^T)^{-1}y} \\ \xrightarrow{\times A} \quad &\mathbf{Ax = AA^T(AA^T)^{-1}y} \\ \xrightarrow{\text{using } *} \quad &\mathbf{Ax = y} \end{aligned}$$

Part (b): Inverse of $\mathbf{y_{n\times 1} = A_{n\times m} x_{m\times 1}}$ where n¿m and rank=m

In this case there are more equations than unknowns, so the system is overdetermined. Because the matrix A is full rank with $rank = m$, A^TA will be an $m \times m$ non-singular square matrix. Therefore, there will be an inverse for it like below:

$$(A^TA)^{-1}(A^TA) = I \qquad (**)$$

As it is clear from the above equation, we may use $(A^TA)^{-1}A^T$ as a left inverse for matrix A.

$$\begin{aligned} &\mathbf{Ax = y} \\ \xrightarrow{\times A^T} \quad &\mathbf{A^TAx = A^Ty} \\ \xrightarrow[\text{using } *]{\times (A^TA)^{-1}} \quad &\mathbf{x = (A^TA)^{-1}A^Ty} \end{aligned}$$

$$\boxed{\mathbf{x = (A^TA)^{-1}A^Ty}}$$

Question 2

Part (a): Difference Between Ridge and LASSO Regressions

Ridge regression uses the the sum of squared values of unknown parameters; however, LASSO regression uses sum of the absolute values of unknown parameters. Therefore, LASSO is capable to make some of the unknown coefficients zero. On the other hand, Ridge regression is able to make the coefficient really small but not absolutely zero.

Part (b): Expressions for Ridge and LASSO Regressions

Ridge Regression:

$$\tilde{\mathbf{x}} = \min_{\mathbf{x}}\{(\mathbf{y} - \mathbf{A}\mathbf{x})^{\mathbf{2}} + \lambda \sum_{\mathbf{i=1}}^{\mathbf{m}} x_i^2\}$$

LASSO Regression:

$$\tilde{\mathbf{x}} = \min_{\mathbf{x}}\{(\mathbf{y} - \mathbf{A}\mathbf{x})^{\mathbf{2}} + \lambda \sum_{\mathbf{i=1}}^{\mathbf{m}} |x_i|\}$$

Question 3

Convolution of two vectors:

$$\begin{bmatrix}1\\2\\2\\1\end{bmatrix} * \begin{bmatrix}1 & 1 & 1\end{bmatrix} = \begin{bmatrix}1\\1+2\\1+2+2\\2+2+1\\2+1\\1\end{bmatrix} = \begin{bmatrix}1\\3\\5\\5\\3\\1\end{bmatrix}$$

Question 4

By using 1x1 convolution in DNNs we can change the depth of channels of a layer so there will be much less computations for the DNN to be trained. We can mention an example here to clarify its purpose:

Example 1: input (256 channels) -¿ 1x1 convolution (64 channels) -¿ 4x4 convolution (256 channels)
Example 2: input (256 channels) -¿ 4x4 convolution (256 channels)

Question 4 cont'd on next page...

If we accept that the computation cost is proportional to the number of weights, we can find it for both the examples above:
number of weights for example 1:

$$(256 \times 64 \times 1 \times 1) + (64 \times 256 \times 4 \times 4) = 278,528$$

$$256 \times 256 \times 4 \times 4 = 1,048,576$$

Question 5

- **Average Pooling**
  It computes the average of the values in its filter.
- **Global Average Pooling**
  This layer will compute the average over any channel of the input individually.
- **Max Pooling**
  It chooses a single maximum value in its filter.

Question 6

Softmax Regression is actually a generalization of logistic regression that could be used for multi-class classification, in contrast of what we have in logistic regression which classifies between two classes.

Question 7

Hinge loss expression:

$$L(d, y) = max(0, 1 - dy)$$

in which d is the desired response and y is the computed output by the SVM.

This loss function is mainly used in support vector machines (SVM).

Question 8

L1 norm:

$$||\mathbf{x}||_{\mathbf{1}} = \sum_{i=1} |x_i|$$

L2 norm:

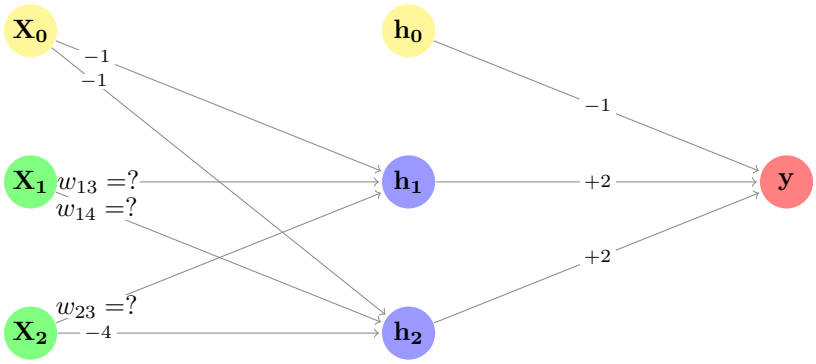
$$||\mathbf{x}||_{\mathbf{2}} = \sqrt{\sum_{i=1} |x_i|^2}$$

Question 9

Two main applications of autoencoders are **data denoising** and **dimensionality reduction** for data visualization. An advantage of autoencoders is that they do not require ay engineering to reduce the dimensionality. So, by only providing enough amount of data they could be trained automatically.

Question 10

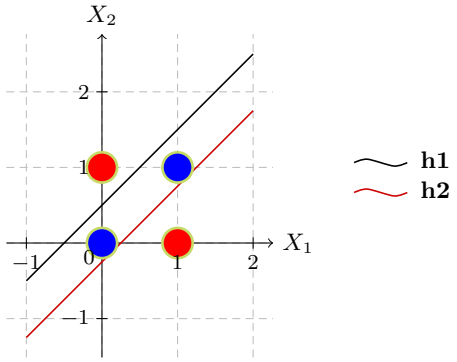
There are three unknown weights to be found on the below neural network, so it can perform XOR operation.



The equation for neurons of the network are written below (φ is the threshold function):

$$\mathbf{h_1} = \varphi(\mathbf{w_{13}X_1} + \mathbf{w_{23}X_2} - \mathbf{1})$$
$$\mathbf{h_2} = \varphi(\mathbf{w_{14}X_1} - \mathbf{4X_2} - \mathbf{1})$$
$$\mathbf{y} = \varphi(\mathbf{2h_1} + \mathbf{2h_2} - \mathbf{1})$$

With calculating the equations of the lines in the X_1-X_2 coordinate system, all of the weights will be found.



$$-\mathbf{2X_1} + \mathbf{2X_2} - \mathbf{1} = \mathbf{0} \rightarrow w_{13} = -2, w_{23} = 2$$
$$\mathbf{4X_1} - \mathbf{4X_2} - \mathbf{1} = \mathbf{0} \rightarrow w_{14} = 4$$

Question 11

Derivative of sigmoid function:

$$\begin{aligned}
\frac{\partial}{\partial x} f(x) &= \frac{\partial}{\partial x}(\frac{1}{1+e^{-\lambda x}}) \\
&= \frac{\lambda e^{-\lambda x}}{(1+e^{-\lambda x})^2} \\
&= \lambda(\frac{e^{-\lambda x}}{1+e^{-\lambda x}})(\frac{1}{1+e^{-\lambda x}}) \\
&= \lambda(\frac{1+e^{-\lambda x}-1}{1+e^{-\lambda x}})(\frac{1}{1+e^{-\lambda x}}) \\
&= \lambda(1-\frac{1}{1+e^{-\lambda x}})(\frac{1}{1+e^{-\lambda x}}) \\
&= \lambda f(x)(1-f(x))
\end{aligned}$$

Question 12

Kernel function is a function which with the help of dot product could do the same operation as mapping from input space to a high dimensional output space.

Kernel Trick is a way by which we can simply compute the inner products of the input samples (with the help of kernel function) in their own input feature space without any need to compute the mapping explicitly. This operation will be less computationally expensive.

Question 13

we start with $\phi(\mathbf{x_i})^T\phi(\mathbf{x_j})$:

$$\begin{aligned}
\phi(\mathbf{x_i})^T\phi(\mathbf{x_j}) &= \begin{bmatrix} x_{i_1}^2 \\ \sqrt{2}x_{i_1}x_{i_2} \\ x_{i_2}^2 \end{bmatrix}^T \begin{bmatrix} x_{j_1}^2 \\ \sqrt{2}x_{j_1}x_{j_2} \\ x_{j_2}^2 \end{bmatrix} \\
&= x_{i_1}^2x_{j_1}^2 + 2x_{i_1}x_{j_1}x_{i_2}x_{j_2} + x_{i_2}^2x_{j_2}^2 \\
&= (x_{i_1}x_{j_1} + x_{i_2}x_{j_2})^2 \\
&= (\mathbf{x_i^T x_j})^2 \\
&= k(\mathbf{x_i}, \mathbf{x_j})
\end{aligned}$$