Boston University
ENG EC 414 Introduction to Machine Learning
Exam 3 Solution
Released on Tuesday, 15 December, 2020 (120 minutes, 42 points + 2 bonus points), submit to Gradescope

---

- *There are 5 problems plus 1 bonus one.*
- *For each part, you must clearly outline the key steps and provide proper justification for your calculations in order to receive full credit.*
- *You can use any material from the class (slides, discussions, homework solutions, etc.), but you cannot look for solutions on the internet. Also, be aware of the limited time.*

Problem 3.1 [10pts] The empirical mean and the empirical covariance matrix of a set of feature vectors of some dataset are given by

$$\hat{\boldsymbol{\mu}}_x = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \text{ and } \hat{S}_x = \begin{pmatrix} 6 & 2 & 1 \\ 2 & 6 & 1 \\ 1 & 1 & 7 \end{pmatrix}.$$

(a) [4pts] Identify 3 orthonormal eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ of $\hat{S}_x$ from the following list of vectors and their corresponding eigenvalues $\lambda_1, \lambda_2, \lambda_3$ (ordered such that $\lambda_1 \geq \lambda_2 \geq \lambda_3$):

$$\mathbf{v}_1 = \frac{1}{\sqrt{3}}\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}, \mathbf{v}_2 = \frac{1}{\sqrt{6}}\begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}, \mathbf{v}_3 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}, \mathbf{v}_4 = \frac{1}{\sqrt{2}}\begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}, \mathbf{v}_5 = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \mathbf{v}_6 = \frac{1}{\sqrt{6}}\begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix}$$

(b) [3pts] Compute all principal components $\hat{y}_1, \hat{y}_2, \hat{y}_3$ of the feature vector $\mathbf{x}_{\text{test}} = \begin{bmatrix} 1 & -1 & 1 \end{bmatrix}^\top$ using the principal directions you found in part (a) and the empirical mean vector.

(c) [3pts] Reconstruct $\mathbf{x}_{\text{test}}$ using its first two principal components and $\hat{\boldsymbol{\mu}}_x$.

Solution:

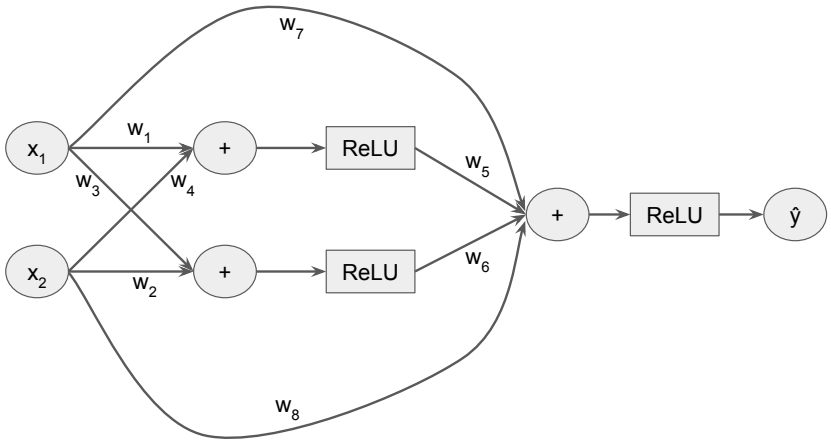
(a) [4pts] $\boxed{\mathbf{u}_1 = \mathbf{v}_5, \lambda_1 = 9, \mathbf{u}_2 = \mathbf{v}_6, \lambda_2 = 6, \mathbf{u}_3 = \mathbf{v}_3, \lambda_3 = 4}$ A vector $\mathbf{u}$ is an eigenvector of $\hat{S}_x$ with eigenvalue λ if, and only if, it is nonzero and $\hat{S}_x\mathbf{u} = \lambda\mathbf{u}$. It is orthonormal if it has unit length. Only $\mathbf{v}_3, \mathbf{v}_5, \mathbf{v}_6$ satisfy this definition with eigenvalues $4, 9, 6$ respectively.

(b) [3pts] $\boxed{\hat{y}_1 = -\frac{2}{\sqrt{3}}, \hat{y}_2 = -\frac{2}{\sqrt{6}}, \hat{y}_3 = \frac{2}{\sqrt{2}}}$ For $k = 1, 2, 3$, $\hat{y}_k = \mathbf{u}_k^\top(\mathbf{x}_{\text{test}} - \hat{\boldsymbol{\mu}}_x) = \mathbf{u}_k^\top \begin{pmatrix} 0 \\ -2 \\ 0 \end{pmatrix} = -2u_{2k}$.

(c) [3pts] $\boxed{(0,\ 0,\ 1)^\top}$ The reconstruction is given by

$$\hat{\boldsymbol{\mu}}_x + \hat{y}_1\mathbf{u}_1 + \hat{y}_2\mathbf{u}_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \frac{2}{\sqrt{3}} \times \frac{1}{\sqrt{3}}\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \frac{2}{\sqrt{6}} \times \frac{1}{\sqrt{6}}\begin{pmatrix} 1 \\ 1 \\ -2 \end{pmatrix} = \begin{pmatrix} 1 - 2/3 - 2/6 \\ 1 - 2/3 - 2/6 \\ 1 - 2/3 + 4/6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Problem 3.2 [13pts] In class, we said that usually neurons in a neural networks have only inputs from neurons of the previous layer. However, this is not a strict rule and we can consider dense networks, where each neuron is connected with all the neurons of all the previous layers. Here, we consider a very simple example of dense network:



Let $\ell(\hat{y}, y) = (\hat{y} - y)^2$ be the loss function, $\begin{bmatrix} 1 & -1 \end{bmatrix}^\top \in \mathbb{R}^2$ be a training sample with corresponding label y equal to 1, and initial weights $w_1 = -1, w_2 = 1/2, w_3 = 1, w_4 = 0, w_5 = -1, w_6 = 4, w_7 = 2, w_8 = 1$. Remember that $ReLU(x) = \max(x, 0)$ and use the value of 0 as "gradient" of the ReLU in 0.

(a) [3pts] Compute the values of $\hat{y}$ and $\ell(\hat{y}, y)$ in the first forward pass iteration of the backpropagation algorithm.

(b) [10pts] Compute the values of the partial derivatives of the loss with respect to $w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8$ in the first backward pass iteration of the backpropagation algorithm.

Solution:

(a) [3pts] Forward Pass First Iteration

Var.	Expression	Value
u_1	$w_1 x_1 + w_4 x_2$	-1
u_2	$w_3 x_1 + w_2 x_2$	0.5
s_1	$\max\{0, u_1\}$	0
s_2	$\max\{0, u_2\}$	0.5
u_3	$w_7 x_1 + w_8 x_2 + w_5 s_1 + w_6 s_2$	3
$\hat{y}$	$\max\{0, u_3\}$	3
ℓ	$(\hat{y} - y)^2$	4

(b) [10pts] Backward Pass First Iteration

Var.	Expression	Value
$\frac{\partial \ell}{\partial \hat{y}}$	$2(\hat{y} - y)$	4
$\frac{\partial \ell}{\partial u_3}$	$\sigma'(u_3) \cdot \frac{\partial \ell}{\partial \hat{y}}$	4
$\frac{\partial \ell}{\partial w_7}$	$x_1 \cdot \frac{\partial \ell}{\partial u_3}$	4
$\frac{\partial \ell}{\partial w_8}$	$x_2 \cdot \frac{\partial \ell}{\partial u_3}$	-4
$\frac{\partial \ell}{\partial s_1}$	$w_5 \cdot \frac{\partial \ell}{\partial u_3}$	-4
$\frac{\partial \ell}{\partial s_2}$	$w_6 \cdot \frac{\partial \ell}{\partial u_3}$	16
$\frac{\partial \ell}{\partial w_5}$	$s_1 \cdot \frac{\partial \ell}{\partial u_3}$	0
$\frac{\partial \ell}{\partial w_6}$	$s_2 \cdot \frac{\partial \ell}{\partial u_3}$	2
$\frac{\partial \ell}{\partial u_1}$	$\sigma'(u_1) \cdot \frac{\partial \ell}{\partial s_1}$	0
$\frac{\partial \ell}{\partial u_2}$	$\sigma'(u_2) \cdot \frac{\partial \ell}{\partial s_2}$	16
$\frac{\partial \ell}{\partial w_1}$	$x_1 \cdot \frac{\partial \ell}{\partial u_1}$	0
$\frac{\partial \ell}{\partial w_4}$	$x_2 \cdot \frac{\partial \ell}{\partial u_1}$	0
$\frac{\partial \ell}{\partial w_2}$	$x_2 \cdot \frac{\partial \ell}{\partial u_2}$	-16
$\frac{\partial \ell}{\partial w_3}$	$x_1 \cdot \frac{\partial \ell}{\partial u_2}$	16

Problem 3.3 [11pts] In class we have mentioned that it is possible to approximate sigmoidal functions with ReLUs, justifying the fact that we can expect neural networks with ReLUs to perform similarly to neural networks with sigmoids. So, let

$$h(x) = \begin{cases} -1 & x \leq -1 \\ x & -1 \leq x \leq 1 \\ 1 & \text{else} \end{cases}$$

Let $\sigma_{ReLU}(t) = \max\{0, t\}$ be the Rectifier Linear Unit activation function. Find values of $(\alpha_1, \beta_1, \gamma_1)$, $(\alpha_2, \beta_2, \gamma_2)$, and α_3 such that for all x,

$$h(x) = \underbrace{\alpha_1 \cdot \sigma_{ReLU}\left(\beta_1 + (\gamma_1 \cdot x)\right)}_{h_1(x)} + \underbrace{\alpha_2 \cdot \sigma_{ReLU}\left(\beta_2 + (\gamma_2 \cdot x)\right)}_{h_2(x)} + \alpha_3$$

Sketch the graphs of $h_1(x), h_2(x)$, and $h(x)$ and properly label axes and key points.

Solution: Multiple solutions are possible, one of them is: $\boxed{\alpha_1 = 1, \beta_1 = 1, \gamma_1 = 1, \alpha_2 = -1, \beta_2 = -1, \gamma_2 = 1, \alpha_3 = -1}$

Hence, $h(x) = ReLU(x + 1) - ReLU(x - 1) - 1 = \max(x + 1, 0) - \max(x - 1, 0) - 1$.

Problem 3.4 [4pts] For each statement, say if it is true or false. No justification is necessary here.

(a) The objective function of a neural network is always non-convex.

(b) Gradient descent converges to a local minimum for non-convex objective functions.

(c) A non-linear activation function in neural networks is needed to construct non-linear predictors

(d) The optimization through gradient descent of a neural network with ReLU activation functions can be initialized with all the weights to 0.

(e) The ReLU activation function in neural networks alleviates the problem of vanishing gradients.

(f) Backpropagation is an algorithm to minimize the objective function of neural networks.

(g) Recurrent neural networks are used to learn over sequences of fixed length.

(h) Convolutional neural network have a smaller variance compared to full-connected neural networks with the same number of neurons.

Solution:

(a) False, consider a network with 1 neuron, linear activation function, and square loss.

(b) True, we stated it without proof.

(c) True, because using linear activation function results in a linear function.

(d) False, if all the weights are zero the gradient is zero and we don't move. Note that it is not a minimum it is a saddle point.

(e) True, we stated it and gave some intuition.

(f) False, Backpropagation calculates the gradient and nothing else.

(g) False, RNN can be used on sequence of varying lengths.

(h) True, smaller number of weights due to the weight-sharing a.k.a. convolutions.

Problem 3.5 [4pts] Say if each statement is true or false and **explain your choices to get full credit**.

(a) The principal directions found by PCA are orthogonal.

(b) The interpretation of PCA as maximizing the variance is true even without centering the data.

(c) Suppose that $X \in \mathbb{R}^{m \times d}$ is the matrix of m samples in $\mathbb{R}^d$, then the number of zero eigenvalues in $XX^\top$ and $X^\top X$ is the same.

(d) Random projections use the labels of the samples too.

Solution:

(a) True, various reasons: eigenvectors are orthogonal or PCA formulation constraints them to be orthogonal.

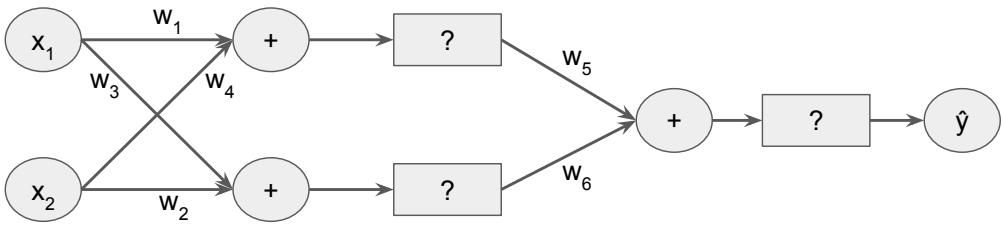
(b) False, to get the definition of variance you have to subtract the mean.

(c) False, the two matrices have the same non-zero eigenvalues but different dimensions.

(d) False, random projections is an unsupervised dimensionality reduction method like PCA.

Problem 3.6 [Bonus, 2pts] We said in class that neural network with ReLU activation function generates piecewise linear predictors and with enough neurons we can approximate any continuous function. Here, instead we want to find the activation functions that might work best for a particular problem

You have a neural network with 2 inputs, 2 neurons in the hidden layer and 1 output neuron. You have a training set $(\boldsymbol{x}_i, y_i)_{i=1}^m$ where $\boldsymbol{x}_i \in \mathbb{R}^2_{++}$ ($\mathbb{R}_+$+ are the positive real numbers). You expect the function to be learned to be roughly $y \approx 10x_1x_2$, where x_1 and x_2 are the inputs. Which activation functions would you use in the neurons in the hidden layer and the output neuron to hope to have a good predictor and why? You can use a different activation function in the hidden layer and in the last layer.



Solution: There are at least a couple of ways to do it. I think the simplest one is to use the activation function $\ln(x)$ in the hidden layer and $\exp(x)$ in the output layer. Given that $\exp(\ln(x_1) + \ln(x_2)) = x_1x_2$ such network would express multiplications. To get the coefficient 10 we would need the presence of the biases in the neurons.

Another solution is to use the activation function x^2 in the hidden layer and the identity function in the output layer. This works because in the hidden layer we can have for example $(x_1 + x_2)^2 = x_1^2 + x_2^2 + 2x_1x_2$ in one neuron and $(x_1 - x_2)^2 = x_1^2 + x_2^2 - 2x_1x_2$. Adding them in the output neuron removes the quadratic terms. I have to admit that I didn't think about this solution at first, but many of you proposed this route. That's a very clever one, I like it!