Fondements de l'Apprentissage Machine (IFT 6390)
Final exam

Professor: Ioannis Mitliagkas

Thursday, December 12th 2019
Duration: 3h00

Allowed material: 4 handwritten sheets front and back (letter format 8" 1/2 x 11") for your course summary.

Name: AKSHAY SINGH RANA **Student ID:** 20152453

There are 100 points in this exam. Give brief but precise answers directly in the black boxes (in either English or French). **Good luck!**

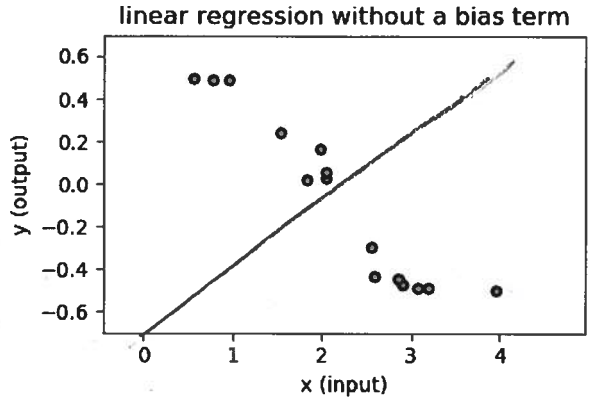
1. **True / False.** [15 points] For each of the questions below, answer by True or False by checking the corresponding box. Correct answers are worth 1 point each, incorrect answers cost -0.5 point each, giving no answer costs 0 points (if the total number of points is negative, you will get 0 points for this question).

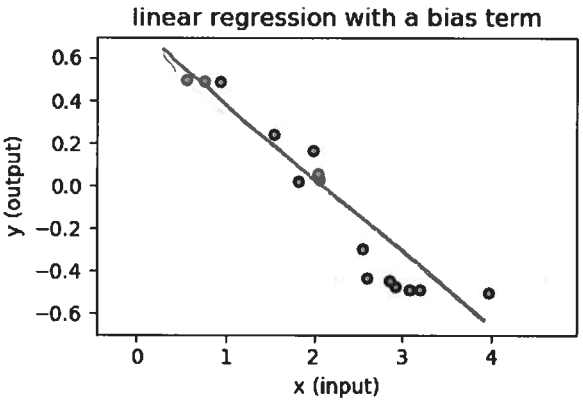
		True	False
(a)	Overfitting manifests itself by a low error on the validation set.	☐	☑
(b)	Underfitting manifests itself by a too big error both on the training and validation sets.	☑	☐
(c)	When you have the choice between several classifiers, you should choose the one that fits better the data it is trained on.	☐	☑
(d)	Linear support vector machines have more capacity than the 1-nearest neighbor algorithm.	☐	☑
(e)	An algorithm with a bigger capacity will tend to have more bias and less variance than an algorithm with smaller capacity.	☐	☑
(f)	The kernel trick allows us to implicitly use high dimensional representations without explicitly computing them.	☑	☐
(g)	The discriminant function computed by kernel methods are a linear function of its parameters, not necessarily a linear function of the inputs.	☑	☐
(h)	Convolutional neural networks are rotation invariant.	☐	☑
(i)	Making a decision tree deeper will surely lower the training error but may increase the test error.	☑	☐
(j)	K-means automatically adjusts the number of clusters.	☐	☑
(k)	Convolutional neural networks generally have fewer free parameters as compared to fully connected neural networks (for example, when we try to classify images of the same size).	☑	☐
(l)	We can get multiple local optimum solutions if we solve a linear regression problem by minimizing the sum of squared errors plus an L2 regularizer using gradient descent.	☑	☐
(m)	A neural network with multiple hidden layers and sigmoid activation functions can form non-linear decision boundaries.	☑	☐
(n)	Given a data matrix $X \in \mathbb{R}^{n \times d}$, where $d \leq n$, if we project our data onto a k dimensional subspace using PCA where k equals the rank of X, we recreate a perfect representation of our data with no loss.	☐	☑
(o)	Consider a neural network with no hidden layer and only one output neuron with a sigmoid activation. Then, gradient descent (with appropriate learning rate) used to minimize the cross-entropy loss function will always converge to a global minimum.	☐	☑

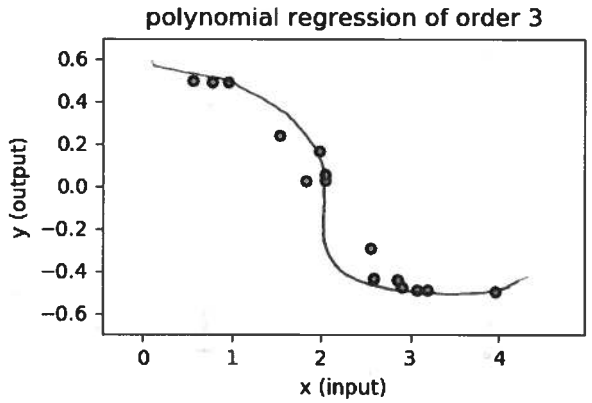
2. **Knowledge of machine learning models.** [30 points]

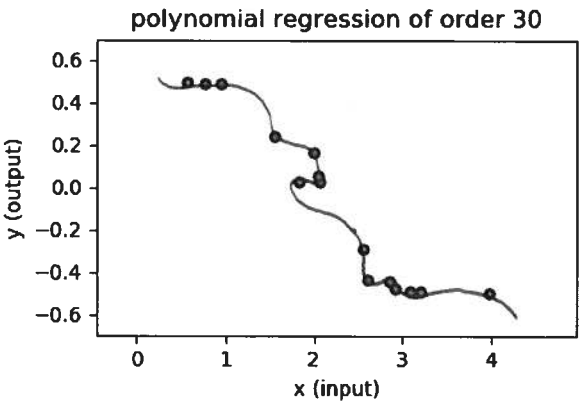
(a) **Regression.**

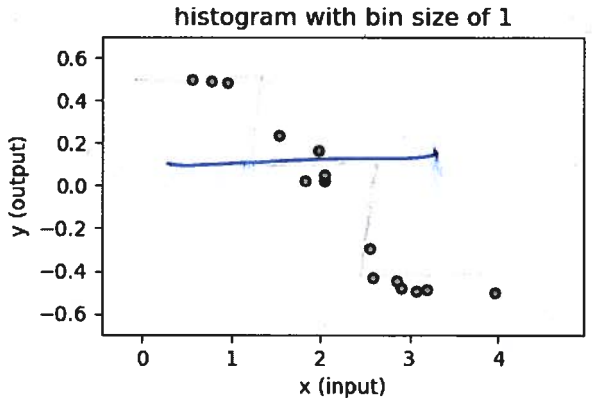
For each algorithm listed below, draw the approximate regression function learned on the dataset. Here, the input of the regression problem is a scalar (x-axis) and the output is a scalar (y-axis).

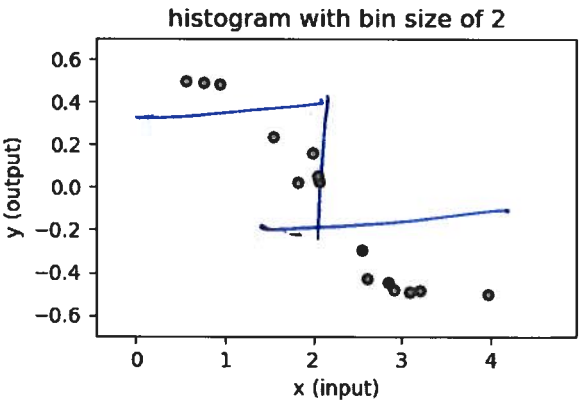












(b) **Linear classifiers.**

i. What is a *linear* classifier?

A linear classifier is the one which follows the below equation of model →

$y = \langle W X \rangle + b$

A linear classifier will give a linear decision boundary in 2D and a linear hyperplane in multi-dimension.

ii. Explain what it means for a set of points to be *linearly separable.*

Points are said to be linearly seperable if a linear classifier (linear line/hyperplane) is able to seperate them completely. It should have <u>zero</u> training error on those points.

iii. Give 3 examples of linear classifiers.

1. Logistic Regression
2. Linear SVM
3. Linear Regression (can also use for classification)

iv. Give an example of a learning algorithm that will return a linear classifier even if the data is not linearly separable.

Linear SVM → (without kernels)

Even if data is non-linear, the decision boundary will be linear.

$$\min \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{n} \max(0, 1 - y(\langle w, x \rangle + b))$$

Loss function ↗

(c) **Neural Networks.**

i. Explain how logistic regression, linear regression and the perceptron are all special cases of fully connected neural networks.

Linear Regression : $y = f_\theta(x) = w_1x_1 + w_2x_2 \cdots w_dx_d + b$

Logistic Regression : $y = f_\theta(x) =$ sigmoid $(\langle w, x\rangle + b)$

Perceptron : $y =$ sign $(\langle w, x\rangle + b)$

Fully connected NN : $y = g\langle w, h_{l-1}\rangle + b$, $g(x) \rightarrow$ activation function

As we can see that in all the above examples, the equation of the output comes down to $\underline{\langle w, x\rangle + b}$, and we can say that above algorithms are simplified special case of fully connected neural network.

ii. What are the two key differences between fully-connected neural networks and convolutional neural networks?

1. Output of Fully connected NN is obtained by multiplying the weight with inputs / output of prev layer, whereas in CNN, we convolve the Kernel weight matrix over input/output of prev layer.
2. The parameters in general are less in CNN as compared to fully connected layer.

iii. Give three examples of activation functions commonly used in neural networks.

1. RELU
2. Sigmoid
3. Tanh.
4. Softmax

iv. In RNN training, what is the main reason that we truncate back propagation through time?

Vanishing Gradient : Since we need to backpropogate to the beginning of the sentence. the gradients may start diminishing or exploding with time, hence the need to truncate them.
or
Exploding Gradient

3. **Principal component analysis** [15 points]

In this question you will do PCA on a simple dataset.

Consider a dataset of 3 data points, $D = \{x_1, x_2, x_3\}$ where

$$x_1 = \begin{bmatrix} 0 \\ 4 \end{bmatrix}, \quad x_2 = \begin{bmatrix} -\sqrt{3} \\ -2 \end{bmatrix}, \quad x_3 = \begin{bmatrix} \sqrt{3} \\ -2 \end{bmatrix}.$$

a) Calculate the 2 principal components and associated eigenvalues.

Hint: First make sure that your data is centered (i.e. zero mean). Then form the empirical covariance matrix. Its eigenvectors correspond to D's principal components.

Note: This is an easy example, but we are interested in seeing that you do all steps of the method, even the ones that are trivial. Please show all your steps.

$x_1 = \begin{bmatrix} 0 \\ 4 \end{bmatrix}$ $x_2 = \begin{bmatrix} -\sqrt{3} \\ -2 \end{bmatrix}$ $x_3 = \begin{bmatrix} \sqrt{3} \\ -2 \end{bmatrix}$

$\bar{x}_1 = 2$ $\bar{x}_2 = -1 - \frac{\sqrt{3}}{2}$ $\bar{x}_3 = -1 + \frac{\sqrt{3}}{2}$

$\Longrightarrow$ centering data

$X^T = \begin{bmatrix} x_1 & x_2 & x_3 \\ -2 & -\sqrt{3}+1+\frac{\sqrt{3}}{2} & \sqrt{3}+1-\frac{\sqrt{3}}{2} \\ 2 & \frac{\sqrt{3}}{2}-1 & -\frac{\sqrt{3}}{2}+1 \end{bmatrix}$ 2×3

$X = \begin{bmatrix} -2 & 2 \\ -\sqrt{3}+1+\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2}-1 \\ 2+1-\frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2}-1 \end{bmatrix}$ 3×2

$\Rightarrow XX^T = \begin{bmatrix} 8 & 2\sqrt{3}-4 & -2\sqrt{3}-4 \\ 2\sqrt{3}-4 & 5-\frac{\sqrt{3}}{2} & -\frac{1}{2} \\ -2\sqrt{3}-4 & -\frac{1}{2} & \frac{5}{2}+\sqrt{3} \end{bmatrix}$ 3×3

$XX^T = U \Sigma U^T$ [Eigen value decomposition of covariance matrix]

U: eigen vectors, Σ: eigen values (diagnol)

2 PCs = First 2 values of U

b) Project the dataset, D, onto its top principal component to reduce its dimension from 2 to 1.
Hint: The PCA method from our slides involves taking the inner product between the top principal component and the centered data point and then normalizing the result by the square root of the corresponding eigenvalue. The end result is a scalar representation $z_i \in \mathbb{R}$ corresponding to each original datapoint x_i.

$$PCA_x = \langle (U\Sigma), (x - \bar{x}) \rangle$$

$x.shape = (3, 2)$

$PCA_x = (3, 1)$

c) From the compressed representation, z_1, z_2, z_3, recover approximate versions of the original dataset, $\hat{x}_1$, $\hat{x}_2$, $\hat{x}_3$.
Hint: You will follow the reverse process to go from a normalized scalar representation to a 2-dimensional representation approximating the original data.

$\hat{X}_1 = Z_1 \times PC$

$\hat{X}_2 = Z_2 \times PC$

$\hat{X}_3 = Z_3 \times PC$

$$X_{3,2} = Z_{3\times 1} \; (U\Sigma)^T_{1\times 2}$$

4. **Maximum likelihood estimation.** [14 points]

(a) Suppose $x_1, x_2, \cdots, x_n \in \mathbb{R}$ are i.i.d. random variables with density function

$$f_\theta(x) = \frac{1}{2\theta} \exp\left(-\frac{|x|}{\theta}\right).$$

Find the maximum likelihood estimate of θ (justify your answer/show your derivations).

$$p(x|\theta) = \prod_{i=1}^{n} \left(\frac{1}{2\theta} \exp\left(\frac{-|x_i|}{\theta}\right)\right)$$

Taking log as it is a monotonic function and converting into summation ..

$$\log\left(p(x|\theta)\right) = \log \sum_{i=1}^{n} \log\left(\frac{1}{2\theta} \exp\left(\frac{-|x_i|}{\theta}\right)\right)$$

$$\theta = \arg\max_{\theta} \log p(x|\theta)$$

$$\therefore \frac{d \log p(x|\theta)}{d\theta} = 0 \,,$$

$$\log\left(\frac{dp(x)}{d\theta}\right) = \frac{-1}{2\theta^2} \exp\left(\frac{-|x|}{\theta}\right) + \frac{1}{2\theta} \frac{|x|}{\theta^2} \exp\left(\frac{-|x|}{\theta}\right)$$

$$= \underbrace{\frac{-1}{2\theta^2} \exp\left(\frac{-|x|}{\theta}\right)}_{-\left[\log\left(\exp\left(\frac{-|x|}{\theta}\right)\right) - \log(2\theta^2)\right]} \underbrace{\left[1 + \frac{|x|}{\theta}\right]}_{\theta = -x} = 0$$

$$\frac{+|x|}{\theta} + \log \theta^2 = 0$$

5. **Radial Basis Function neural networks** [26 points]

For Multilayer Perceptron (MLP) networks seen in class, a neuron N_k of the first hidden layer receives an entry x and has a weight vector $w_k \in \mathbb{R}^d$ and a bias $b_k \in \mathbb{R}$. It calculates its output h_k with the formula $h_k = \text{sigmoid}(\langle w_k, x\rangle + b_k)$, where $\langle w_k, x\rangle$ denotes the usual inner product.

This question focuses on a different type of neural network called RBF (Radial Basis Function) network. These single-layer networks are very similar to MLPs. The difference is that an RBF neuron N_k at the hidden layer, with weight vector w_k calculates its output h_k as follows:

$$\begin{aligned} h_k &= \exp(-\beta\|x - w_k\|^2) \\ &= \exp\left(-\beta \sum_{j=1}^{d} (x_j - w_{kj})^2\right) \end{aligned}$$

where exp stands for the exponential function and β is a hyper-parameter (the same for all neurons in the first hidden layer). Also note that there is no bias. A single hidden layer of m RBF neurons with outputs $(h_1, \ldots, h_m) = h$ is typically followed by a linear output layer with weights $(a_1, \ldots, a_m) = a$ to give an output $y = f_\theta(x) = \langle a, h\rangle = \sum_{k=1}^{m} a_k h_k$.

(a) What is the set, θ, of parameters (excluding hyperparameters) of such an RBF network? $\theta = \{\ldots\}$

$\theta = \{ w_1, w_2 \cdots w_m + a_1, a_2, \cdots a_m \}$ → $w_1 \to m$ ax vectors of dimension d

↑ weights between input layer and hidden layer ↑ weights b/w hidden layer and output layer

How many adjustable real numbers (how many scalars) does this correspond to?

$(d \times m) + (m \times 1)$ where d = no of dimensions
m = no. of neurons in hidden layer
1 = no. of output neurons.

(b) Consider an example x, for which the network predicts $f_\theta(x)$, while the real target is t. The loss for this prediction is given by a differentiable function $L(f_\theta(x), t)$. We're looking for parameter values that will minimize the average empirical cost over a training set $D_n = \{(x^{(1)}, t^{(1)}), \ldots, (x^{(n)}, t^{(n)})\}$. State this minimization problem.

Average empirical cost $\hat{R}(f_\theta, D_n) = \sum_{i=1}^{n} L(f_\theta(x_i), t_i)$

$\min \hat{R}(f_\theta, D_n)$

(c) Assume we organized θ as a vector containing all the parameters, and that we have a function to effectively calculate $\frac{\partial L}{\partial \theta}$. Write the procedure of stochastic gradient decency that would optimize the parameters:

for each example in training set:

$$\theta = \theta - \eta \frac{\partial L}{\partial \theta} \quad , \quad \eta = \text{training rate}$$

Since, it is stochastic, we will update the θ vector after finding the loss after each example.

(d) We want to use this network to predict a real t target. We are dealing with a regression problem. We will denote the network prediction by $y = f_\theta(x)$. Write down the specific loss function that we most commonly use for such a problem, known as the mean squared error: $L(y,t) = \ldots$

$$L(y,t) = (t-y)^2 = (t - f_\theta(x))^2$$

if all sample $L(y,t) = \frac{1}{n}\sum_{i=1}^{n} (t_i - f_\theta(x_i))^2$ (average over all samples)

(e) To be able to apply the gradient descent method, it is necessary to know how to calculate the gradient, i.e. the partial derivative of the loss L with respect to the parameters θ. To do this, we will use backpropagation.
Start by calculating $\frac{\partial L}{\partial y} = \ldots$

$$L = (t-y)^2$$

$$\frac{dL}{dy} = 2(t-y)\cdot(-1) = -2(t-y) \quad , \quad y = f_\theta(x) = \sum_{k=1}^{m} a_k h_k$$

(f) Express and calculate $\frac{\partial L}{\partial a_k} = \ldots$ et $\frac{\partial L}{\partial h_k} = \ldots$ (based on a_k, h_k, and $\frac{\partial L}{\partial y}$ without substituting them with their full expressions):

$$y = \sum_{i=1}^{m} a_k h_k$$

$$\frac{\partial L}{\partial a_k} = \frac{\partial L}{\partial y} \times \frac{\partial y}{\partial a_k}$$

$$= \frac{\partial L}{\partial y} \times h_k$$

$$\frac{\partial L}{\partial h_k} = \frac{\partial L}{\partial y} \times \frac{\partial y}{\partial h_k}$$

$$= \frac{\partial L}{\partial y} \times a_k$$

(g) Then express and calculate the gradient with respect to the parameters w (depending among other things on $\frac{\partial L}{\partial h_k}$ without substituting it with its full expression): $\frac{\partial L}{\partial w_{kj}} = \ldots$

$$h_k = \exp\left(-\beta \sum_{j=1}^{d} (x_j - w_{kj})^2\right)$$

$$\frac{\partial L}{\partial w_{kj}} = \frac{\partial L}{\partial h_k} \times \frac{\partial h_k}{\partial w_{kj}}$$

$$= \frac{\partial L}{\partial h_k} \times \exp\left(-\beta (x_j - w_{kj})^2\right) \left(-2\beta (x_j - w_{kj})\right) (-1)$$

$$= \frac{\partial L}{\partial h_k} \times \left(2\beta (x_j - w_{kj})\right) e^{-\beta (x_j - w_{kj})^2}$$

Remember the formula for differentiating an exponential: $\exp(u)' = u' \exp(u)$, *or even:* $\frac{\partial \exp(u)}{\partial \theta} = \frac{\partial u}{\partial \theta} \exp(u)$

(h) Using the quantities calculated above (but again, without substituting their expressions that you have already given), write the specific update of network parameters using a stochastic gradient step (with a learning rate η).

For each example in training set

$$w = w - \eta \frac{\partial L}{\partial w}$$

$$a = a - \eta \frac{\partial L}{\partial a}$$