Assignment 3: Neural Network & Image Classification

UVA CS 4501-03 :
Machine Learning (Spring 2018)

Out: Mar. 13 2018
Due: Mar. 31th, Sat midnight 11:59pm, 2018 @ Collab

a *The assignment should be submitted in the PDF format through Collob. If you prefer hand-writing QA parts of answers, please convert them (e.g., by scanning or using PhoneApps like officeLens) into PDF form.*

b *For questions and clarifications, please post on piazza.*

c *Policy on collaboration:*

Homework should be done individually: each student must hand in their own answers. It is acceptable, however, for students to collaborate in figuring out answers and helping each other solve the problems. We will be assuming that, with the honor code, you will be taking the responsibility to make sure you personally understand the solution to any work arising from such collaboration.

d *Policy on late homework: Homework is worth full credit at the midnight on the due date. Each student has three extension days to be used at his or her own discretion throughout the entire course. Your grades would be discounted by 15% per day when you use these 3 late days. You could use the 3 days in whatever combination you like. For example, all 3 days on 1 assignment (for a maximum grade of 55%) or 1 each day over 3 assignments (for a maximum grade of 85% on each). After you've used all 3 days, you cannot get credit for anything turned in late.*

1 Question: Neural Network Playground

Tensorflow Playground is an interactive tool for learning neural networks (more specifically, multi-layer-perceptron[1] networks). A customized version of Tensorflow Playground is available at `http://www.cs.virginia.edu/yanjun/HW/playground/`.

First, we will get familiar with the interface of Tensorflow Playground. Let's get to it!

You can choose from four different datasets on the left side of the page in the DATA panel: Circle, Exclusive Or, Gaussian, and Spiral.The actual data point locations are available on the right side under the OUTPUT label. Do not change the ratio of training to test data or click the REGENERATE button throughout this question. Batch size indicates how many samples are used in your mini-batch gradient descent. You may change this parameter if you want.

The neural network model is in the middle of DATA and OUTPUT. This model is a standard "feed-forward" neural network, where you can vary: (1) the input features (2) the number of hidden layers (3) the number of neurons at each layer. By default, it uses only the raw inputs X_1 and X_2 as features, and no hidden layers. You will need to change theses attributes later.

Several hyper-parameters are tunable at the top of the page, such as the learning rate, the activation (non-linearity) function of each neuron, the regularization norm as well as the regularization rate.

[1] A perceptron model typically uses a heaviside step function as its activation. Since we use a sigmoid (or other differentiable nonlinearity), our representation not exactly a perceptron model, but that is what we call it

There are three buttons at the top left for you to control the training of a neural network: Reset, Run/Pause and Step (which steps through each mini-batch of samples at a time).

1.1 Hand-crafted Feature Engineering

Now that we are familiar with the Playground, we will start to build models to classify samples.

First you are going to earn some experience in hand-crafted feature engineering with a simple perceptron model with no hidden layers, sigmoid activations, no regularization. In other words, don't change the model you're given by default when loading the page.

A perceptron (single-layer artificial neural network) with a sigmoid activation function is equivalent to logistic regression. As a linear model, it cannot fit some datasets like the Circle, Spiral, and Exclusive Or. To extend linear models to represent nonlinear functions of x, we can apply the linear model to a transformed input $\phi(x)$.

One option is to manually engineer ϕ. This can easily be done in the Playground since you are given 7 different features to choose from in the FEATURES panel.

Task: You are required to find the best perceptron models for the four datasets, **Circle, Exclusive Or, Gaussian and Spiral** by choosing different features. Try to select as few features as possible. For the best model of each dataset, you should report the selected features, iterations and test loss in a table. If you also change other hyper-parameters, *e.g.* the learning rate, you should include them in your report.

For each dataset (Circle, Exclusive Or, Gaussian and Spiral), please include a **web page screenshot** of the result in your report and explain why this configuration works.

(Note: Don't worry if you cannot find a good perceptron for the Spiral dataset. For the other three datasets, the test loss of a good model should be lower than 0.001.)

1.2 Regularization

Forget the best features you have found in last question. You should now select all the possible (*i.e.* all 7) features to test the regularization effect here.

You are required to work on the three datasets: **Circle, Exclusive Or and Gaussian**.

Task A: Try both L1 and L2 regularization with different (non-zero) regularization rates. In the report, you are required to compare the decision boundary and the test loss over the three models trained with similar number of iterations: no regularization, L1 regularized, L2 regularized.

For each dataset (Circle, Exclusive Or and Gaussian), please include a **web page screenshot** of the result in your report and explain why this configuration works.

Task B: We have learned that L1 regularization is good for feature selection. Take a look at the features with significantly higher weights. Are they the same as the ones you select in last question? Write down the results you observe in your report. (You can get the feature weights by moving the mouse pointer over the dash lines.)

1.3 Automated Feature Engineering with Neural Network

While we were able to find different parameters which were able to make good predictions, the previous two sections required a lot of hand-engineering and regularization tweaking :(We will now explore the power of a neural network's ability to *automatically* learn good features :) Let's try it out on two datasets: the **Circle** and **Exclusive Or**.

Here we should only select X_1 and X_2 as features (since we are trying to automatically learn all other features from the network). As we have previously seen, a simple perceptron is not going to learn the correct boundaries since both datasets are not linearly separable. However, a more complex neural network should be able to learn an approximation of the complex features that we have selected in the previous experiments.

You can click the + button in the middle to add some hidden layers for the model. There is a pair of + and - buttons at the top of each hidden layer for you to change the number of the hidden units. Note that each hidden unit, or neuron, is the same neuron from Lecture 17 (possibly varying the sigmoid activation function).

Task: Find a set of neural network model parameters which allow the model to find a boundary which correctly separates the testing samples. Report the test loss and iterations of the best model for each dataset. If you modify the other parameters (e.g. activation function), please report them too. Can it beat or approach the result of your hand-crafted feature engineering?

For each dataset (Circle, Exclusive Or), please include a **web page screenshot** of the result in your report and explain why the configuration would work.

1.4 Spiral Challenge (0.5 Extra credit)

Congratulations on your level up!

Task: Now try to find a model that achieves a test loss lower than 0.01 on the **Spiral** dataset. You're free to use other features in the input layer besides X_1 and X_2, but a simpler model architecture is preferred. Report the **input features, network architecture, hyper parameters, iterations and test loss** in a table. For simplicity, please represent your network architecture by the hidden layers **a-b-c-...**, where a, b, c are number of hidden units of each layer respectively.

Please include a **web page screenshot** of the result in your report and explain why this configuration works.

You are now a neural network expert (on the Playground)!

2 Question: Image Classification)

An online tutorial might be helpful to you for finishing this assignment,
`http://blog.yhathq.com/posts/image-classification-in-Python.html`

As part of Homework 3, you will be participating in a class-wide Kaggle-style competition. Given a dataset of images of handwritten digits, you will create a model to classify a given image as the digit it represents. You will be competing with your peers to create the model with the greatest classification accuracy.

The data set is available as zip.train and zip.test. Each example in the train and test data is a the number in question followed by 256 grayscale values representing a 16×16 image. Values have already been normalized. You can read a full description of the data set in the zip.info.txt file available in collab. Visualization of some samples are provided in the subdir "image_digits" in the attached data ZIP file.

Your objective is to develop a model with the highest possible classification accuracy for the images.

2.1 Evaluation

The main evaluation metric for this competition is Classification Accuracy (%-age of correctly classified images). Classification accuracy is given by:

$$Acc = \frac{\#correct}{\#total}$$

on the testing dataset.

Your code should generate predicted labels of the testing dataset. Each prediction label should be on its own line!

The results file should follow the following format:

```
1
9
5
6
2
3
8
4
6
0
```

2.2 Extra Credit

Top-performing models will receive up to 1 extra credit point on this homework: (The top 20 submissions will get extra credits. We will run your code to generate the predicted labels.)

2.3 Submission

Please submit your source code and PDF report containing your written answers via collab.

In collab, you will find starter code in number_recognition.py. This file will be run from the command line.

python number_recognition.py path/to/training_data path/to/testing_data

Feel free to add any methods or classes to your implementation, so long as your preserve the given command line behavior for your submission.

2.4 About Data

Normalized handwritten digits, automatically scanned from envelopes by the U.S. Postal Service. The original scanned digits are binary and of different sizes and orientations; the images here have been deslanted and size normalized, resulting in 16 x 16 grayscale images (LeCun et al., 1990).

The data are in two files. Each line in the training dataset consists of the digit id (0-9) followed by the 256 grayscale values. The testing dataset is identical, but with no digit id.

There are 7291 training observations and 2007 test observations, distributed as follows:

	0	1	2	3	4	5	6	7	8	9	Total
Train	1194	1005	731	658	652	556	664	645	542	644	7291
Test	359	264	198	166	200	160	170	147	166	177	2007

or as proportions:

	0	1	2	3	4	5	6	7	8	9
Train	0.16	0.14	0.1	0.09	0.09	0.08	0.09	0.09	0.07	0.09
Test	0.18	0.13	0.1	0.08	0.10	0.08	0.08	0.07	0.08	0.09

Alternatively, the training data are available as separate files per digit (and hence without the digit identifier in each row)

The test set is notoriously "difficult", and a 2.5% error rate is excellent. These data were kindly made available by the neural network group at AT&T research labs (thanks to Yann LeCunn).

Figure 1: Description of this data set (zip.info.txt file)

```
Normalized handwritten digits, automatically
scanned from envelopes by the U.S. Postal Service. The original
scanned digits are binary and of different sizes and orientations; the
images  here have been deslanted and size normalized, resulting
in 16 x 16 grayscale images (Le Cun et al., 1990).

The data are in two gzipped files, and each line consists of the digit
id (0-9) followed by the 256 grayscale values.

There are 7291 training observations and 2007 test observations,
distributed as follows:
         0    1   2   3   4   5   6   7   8   9 Total
Train 1194 1005 731 658 652 556 664 645 542 644 7291
 Test  359  264 198 166 200 160 170 147 166 177 2007

or as proportions:
         0    1   2    3    4    5    6    7    8    9
Train 0.16 0.14 0.1 0.09 0.09 0.08 0.09 0.09 0.07 0.09
 Test 0.18 0.13 0.1 0.08 0.10 0.08 0.08 0.07 0.08 0.09


Alternatively, the training data are available as separate files per
digit (and hence without the digit identifier in each row)

The test set is notoriously "difficult", and a 2.5% error rate is
excellent. These data were kindly made available by the neural network
group at AT&T research labs (thanks to Yann Le Cunn).
```

2.5 Discussion of PCA and Logistic Regression

Principal component analysis allows us to reduce the dimensions of our data while retaining as much variance as possible, thus retaining information to separate our data points. Scikit learn provides multiple ways to perform dimension reduction with PCA. For this problem, sklearn.decomposition.RandomizedPCA is recommended.

- Please select three different values of PCs you choose for PCA and present your prediction error results when using the Scikit logistic regression classifier. Please draw a figure showing how the prediction error changes when varying the number of PCs? Please also include the error discussions about the logistic regression without PCA in the discussion and in the figure.
- Since the labels of the zip.test are fake, you surely should not use results from zip.test to discuss the performance.
- - Q: can we use predefined functions like logistic regression, or cross validation from scikit-learn?
  - A: Yes.
- - Q: how many classifiers should be made, and how to deal with using the best one but including all of them?
  - A: You need to submit all the codes about the methods and model selection of the methods you tried. And you need to make sure the main function providing the interface for TA to run as the best one that you make!
  - A: please discuss the results of what you have tried in the written part of the submission.
- - Q: Must we implement all the models suggests by the skeleton code?
  - A: No. The template is just a recommendation what you can try.
  - A. The minima requirement for this coding problem requires your discussion of running the LogRegression(LoG) and PCA+LoG on the datasets.
- - Q: Is testing data being released separately and if so when? - Q: If not, will you simply run the scripts as stated in the homework instructions? (without the parameter for model selection, students just only include the one they want run).

- A. The feature part of the testing data has been included in the data.zip. We will release its labels when we release the result key of HW3.
- A: Yes. We will simply run your submitted code as stated in the instruction.

Congratulations ! You have implemented a state-of-the-art machine-learning toolset for an important OCR image labeling task !

3 Sample Exam Questions:

Each assignment covers a few sample exam questions to help you prepare for the midterm and the final. (Please do not bother by the information of points in some the exam questions.)

Question 1. Neural Nets (a) [5 points] Consider a single sigmoid threshold unit with three inputs x_1, x_2, and x_3.

$$y = g(w_0 + w_1x_1 + w_2x_2 + w_3x_3) \quad where \quad g(z) = \frac{1}{1 + exp(-z)}$$

We input values of either 0 or 1 for each of these inputs. Assign values to weights w_0, w_1, w_2, and w_3 so that the output of the sigmoid unit is greater than 0.5 if and only if $(x_1\ AND\ x_2)\ OR\ x_3$.
Answer: Answer: Multiple possible solutions. One solution: $w_0 = -0.75, w_1 = w_2 = 0.5, w_3 = 1$

(b) [10 points] Answer the following true or false (No explanation required).

- One can perform linear regression using either matrix algebra or using gradient descent. **Answer:** Answer: True
- The error surface followed by the gradient descent backpropagation algorithm changes if we change the training data. **Answer:** Answer: True
- Stochastic gradient descent is always a better idea than batch gradient descent. **Answer:** Answer: False
- Given a two-input sigmoid unit with weights w_0, w_1, and w_2, we can negate the value of the unit output by negating all three weights. **Answer:** Answer: Can be true or false based on interpretation
- The gradient descent update rule for a unit whose output is $w_0 + w_1(x_1 + 1) + w_2(x_2)^2$ is:

$$\Delta w_0 = \eta \sum_d (t_d - o_d)$$

$$\Delta w_1 = \eta \sum_d [(t_d - o_d)x_{d1} + (t_d - o_d)]$$

$$\Delta w_2 = \eta \sum_d [(t_d - o_d)2x_{d2}]$$

  Where:

  - t_d is the target output for the dth training example.
  - o_d is the unit output for the dth training example.
  - x_{d1} is the value of x_1 for the dth training example.
  - x_{d2} is the value of x_2 for the dth training example.

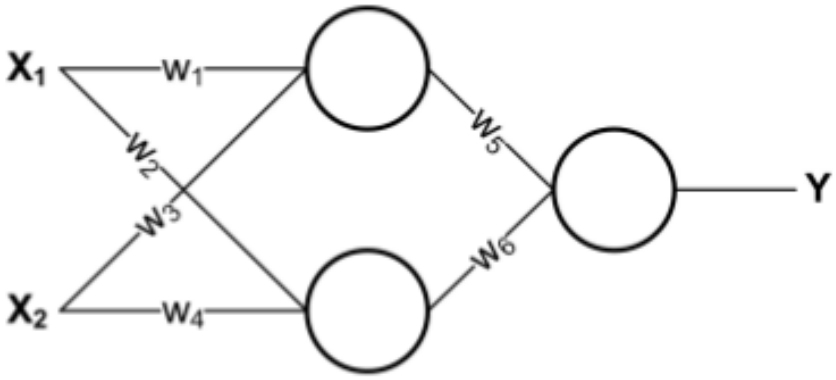
  Answer: Answer: False; $\Delta w_2 = \eta \sum_d [(t_d - o_d){x_{d2}}^2]$

Question 2. Neural Nets and Regression Consider a two-layer neural network to learn a function $f : X \rightarrow Y$ where $X = \langle X_1, X_2 \rangle$ consists of two attributes. The weights, $w_1, ..., w_6$, can be arbitrary. There are two possible choices for the function implemented by each unit in this network:

- **S:** signed sigmoid function $S(a) = sign[\sigma(a) - 0.5] = sign[\frac{1}{1+exp(-a)} - 0.5]$
- **L:** linear function $L(a) = ca$

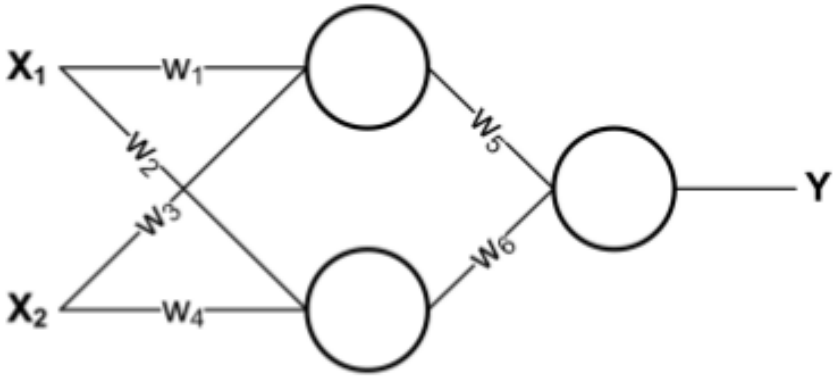
Where in both cases $a = \sum_i w_i X_i$.

(a) [4 points] Assign proper activation functions (**S** or **L**) to each unit in the following graph so this neural network simulates a linear regression: $Y = \beta_1 X_1 + \beta_2 X_2$.



Answer: Answer: Assign all L's

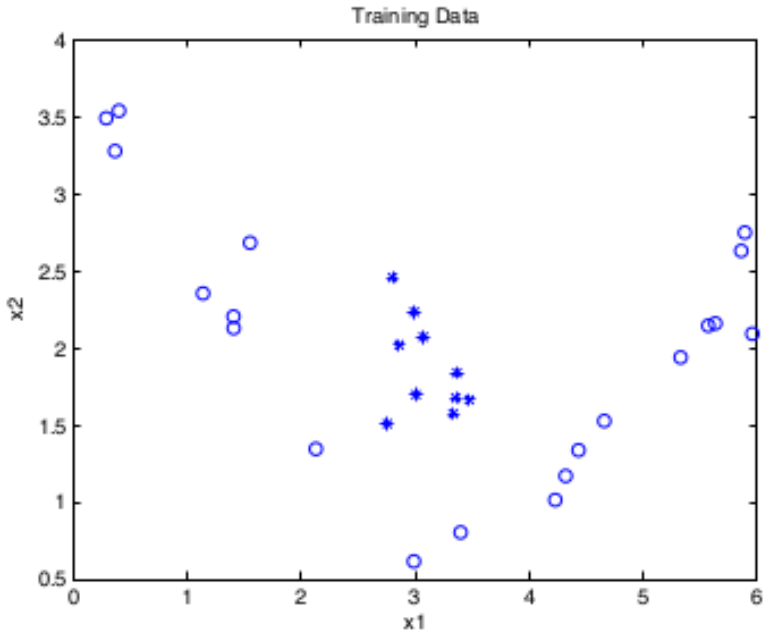
(b) [4 points] Assign proper activation functions (**S** or **L**) to each unit in the following graph so this neural network simulates a binary logistic regression classifier: $Y = argmax_y P(Y = y|X)$, where $P(Y = 1|X) = \frac{exp(\beta_1 X_1 + \beta_2 X_2)}{1+exp(\beta_1 X_1 + \beta_2 X_2)}, P(Y = -1|X) = \frac{1}{1+exp(\beta_1 X_1 + \beta_2 X_2)}$.

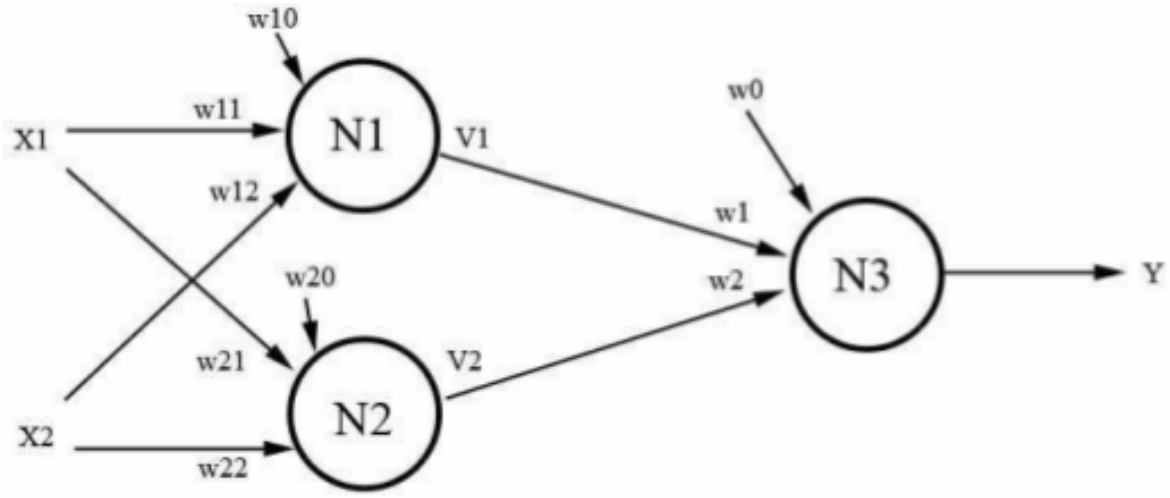


Answer: Answer: First two neurons are L's, output is S

(c) [3 points] Following (b), derive β_1 and β_2 in terms of $w_1, ..., w_6$.
Answer: Answer: $\beta_1 = c(w_1 w_5 + w_2 w_6), \beta_2 = c(w_3 w_5 + w_4 w_6)$

Question 3. Neural Nets Consider the following classification training data (where "*" = true or 1 and "O" = false or 0) and neural network model that uses the **sigmoid** response function ($g(t) = \frac{1}{1+e^(-t)}$).

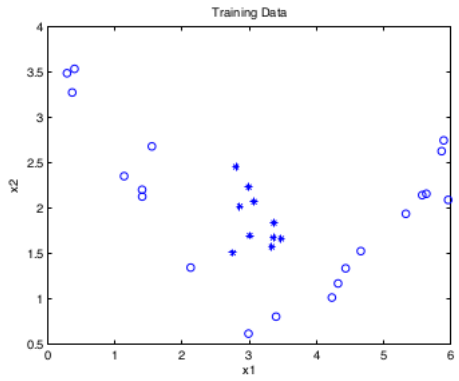




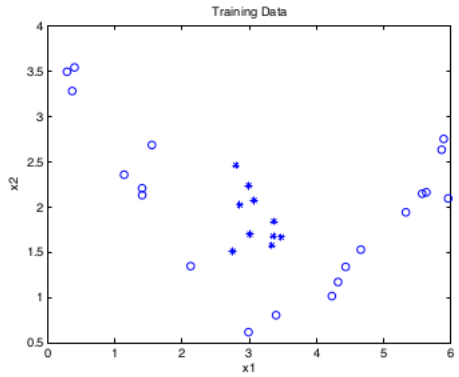
(a) [8 points] We would like to set the weights (w) of the neural network so that it is capable of correctly classifying this dataset. Please plot the decision boundaries for N_1 and N_2 (e.g., for neuron N_1, the line where $w_{10} + w_{11}X_1 + w_{12}X_2 = 0$) on the first two graphs. In the third graph, which has axes V_2 and V_1, plot $V_1(x_1, x_2), V_2(x_1, x_2)$ for a few of the training points and provide a decision boundary so that the neural net will correctly classify the training data.

All graphs are on the following page!

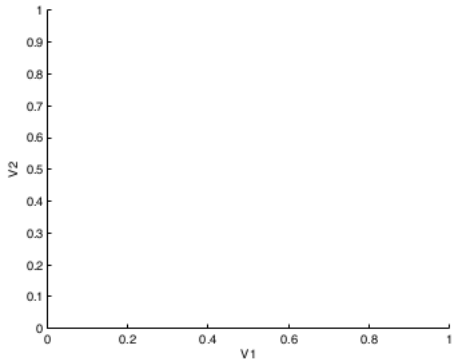
N1 (2 points)



N2 (2 points)

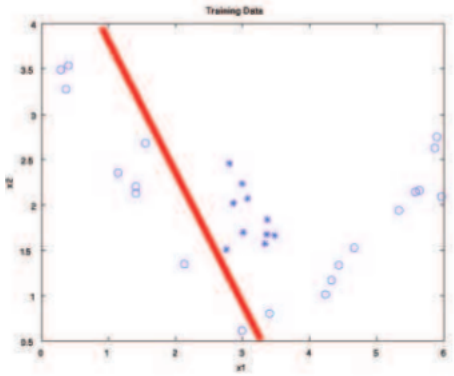


N3 (4 points)

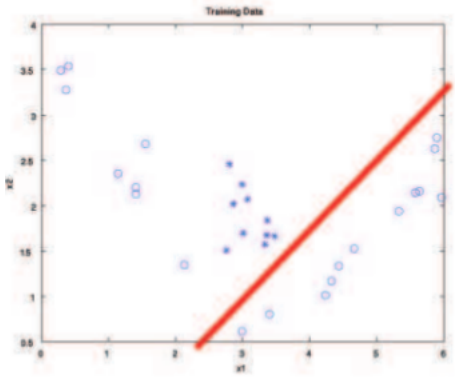


Answer: Answer:

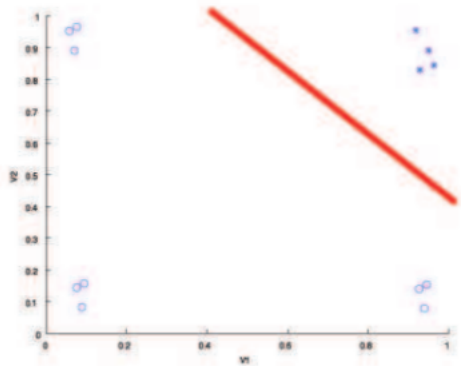
N1 (2 points)



N2 (2 points)



N3 (4 points)



Question 3. More Advanced Regression (Extra)

I have a dataset with R records in which the i^{th} record has one real-valued input attribute x_i and one real-valued output attribute y_i.

We have the following model with one unknown parameter w which we want to learn from data.

$$y_i \sim Normal(exp(wx_i), 1)$$

Note that the variance is known and equal to one.

1. (3 points) (no explanation required) Is the task of estimating w

   A. a linear regression problem?

   B. a non-linear regression problem?

   Answer: B

2. (6 points) (no explanation required) Suppose you decide to do a maximum likelihood estimation of w. You do the math and figure out that you need w to satisfy one of the following equations. Which one?

   (a) $\Sigma_i x_i exp(wx_i) = \Sigma_i x_i y_i exp(wx_i)$

   (b) $\Sigma_i x_i exp(2wx_i) = \Sigma_i x_i y_i exp(wx_i)$

   (c) $\Sigma_i x_i^2 exp(wx_i) = \Sigma_i x_i y_i exp(wx_i)$

   (d) $\Sigma_i x_i^2 exp(wx_i) = \Sigma_i x_i y_i exp(wx_i/2)$

   (e) $\Sigma_i exp(wx_i) = \Sigma_i y_i exp(wx_i)$

   Answer: B (this is totally extra and will not be covered in exams.)

Question 4. More Advanced Regression (Extra)

Now we use the following model to fit the data. The model has one unknown parameter w to be learned from data.

$$y_i \sim N(log(wx_i), 1)$$

Note that the variance is known and equal to one. (no explanation required) Suppose you decide to do a maximum likelihood estimation of w. You do the math and figure out that you need w to satisfy one of the following equations. Which one?

$A. \sum_i x_i log(wx_i) = \sum_i x_i y_i log(wx_i)$
$B. \sum_i x_i y_i = \sum_i x_i y_i log(wx_i)$
$C. \sum_i x_i y_i = \sum_i x_i log(wx_i)$
$D. \sum_i y_i = \sum_i log(wx_i)$

Answer: (this is totally extra and will not be covered in exams.) we perform Maximum Likelihood estimation.
$y_i \sim N(log(wx_i), 1)$
We could write the log likelihood as:

$$LL = log(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(y_i - log(wx_i))^2}{2\sigma^2})) = \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}} exp(-\frac{(y_i - log(wx_i))^2}{2}))$$

$$\frac{\partial LL}{\partial w} = 0 \rightarrow \frac{\partial \sum_{i=1}^{n} (y_i - log(wx_i))^2}{\partial w} = 0$$

Question 5: Logistic Regression + L2 norm (Extra)

We consider the following models of logistic regression for a binary classification with a sigmoid function $g(x) = \frac{1}{1+e^{-z}}$

Model 1 : $P(Y = 1|X, w_1, w_2) = g(w_1X_1 + w_2X_2)$

Model 2 : $P(Y = 1|X, w_1, w_2) = g(w_0 + w_1X_1 + w_2X_2)$

Suppose we train the logistic regression model (Model 2) based on the n training examples $x^{(1)}, ..., x^{(n)}$ and labels $y^{(1)}, ..., y^{(n)}$ by maximizing the penalized log-likelihood of the labels:

$$\sum_i logP(y^{(i)}|x^{(i)}, w) - \frac{\lambda}{2}||w||^2 = \sum_i logP(y^{(i)}w^Tx^{(i)}) - \frac{\lambda}{2}||w||^2$$

For large λ (strong regularization), the log-likelihood terms will behave as linear functions of w.

$$log \;\; g(y^{(i)}w^Tx^{(i)}) \approx \frac{1}{2}y^{(i)}w^Tx^{(i)}$$

Express the penalized log-likelihood using this approximation (with Model 1), and derive the expression for MLE $\hat{w}$ in terms of λ and training data $x^{(i)}, y^{(i)}$. Based on this, explain how w behaves as λ increases. (We assume each $x^{(i)} = (x_1^{(i)}, x_2^{(i)})^T$ and $y^{(i)}$ is either 1 or -1)

Answer:

(this is totally extra and will not be covered in exams.)

$$log\ l(w) \approx \sum_i \frac{1}{2}y^{(i)}w^Tx^{(i)} - \frac{\lambda}{2}||w||^2$$

$$\frac{\partial}{\partial w_1} log\ l(w) \approx \frac{1}{2}\sum_i y^{(i)}x_1^{(i)} - \lambda w_1 = 0$$

$$\frac{\partial}{\partial w_2} log\ l(w) \approx \frac{1}{2}\sum_i y^{(i)}x_2^{(i)} - \lambda w_2 = 0$$

$$\therefore w = \frac{1}{\lambda}\sum_i y^{(i)}x^{(i)}$$