# MIDTERM: CS 6375
# INSTRUCTOR: VIBHAV GOGATE
# October, 27 2014

**The exam is closed book. You are allowed a one-page cheat sheet. Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, use an additional sheet (available from the instructor) and staple it to your exam.**

- **NAME:** ***Solution***

- **UTD-ID if known** _____

| **Section Number** | **Points** |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| Total | |

# 1 SHORT ANSWERS [11 points]

1. (3 points) For linearly separable data, can a small slack penalty hurt the training accuracy when using a linear SVM (no kernel)? If so, explain how. If not, why not?

**Solution:** Yes. Consider the dual formulation (see class slides). We know that all $\alpha$'s should be bounded by $C$. If the optimal values of $\alpha$'s (in the dual formulation) are greater than $C$, we may end up with a sub-optimal decision boundary with respect to the training examples.

2. (3 points) Let us assume that the data ($y$) was generated from the following distribution:

$$\Pr(y) = \frac{\theta^y e^{-5\theta}}{y!}$$

Let us assume that you are given $n$ data points $y_1, \ldots, y_n$ independently drawn from this distribution. Write down the expression for the maximum likelihood estimate for $\theta$.

**Solution:** Consider the expression for the log-likelihood of the data:

$$LL = \sum_{i=1}^{n} \log(\Pr(y_i)) = \sum_{i=1}^{n} \log\left(\frac{\theta^{y_i} e^{-5\theta}}{y_i!}\right)$$

Simplifying the term inside the bracket, we get

$$LL = \sum_{i=1}^{n} y_i \log\theta - 5\theta - \log(y_i!)$$

The maximum likelihood estimate of $\theta$ is given by:

$$\theta_{MLE} = \arg\max_{\theta}\left(\sum_{i=1}^{n} y_i \log\theta - 5\theta - \log(y_i!)\right)$$

We can solve this by computing the derivative of the log-likelihood w.r.t. $\theta$ and setting it to zero.

The derivative is given by

$$\sum_{i=1}^{n} \frac{y_i}{\theta} - 5$$

Setting it to zero and solving for $\theta$, we get

$$\theta_{MLE} = \frac{\sum_{i=1}^{n} y_i}{5n}$$

3. (2 points) Gaussian Naive Bayes is a linear classifier. True or False. Explain.

**Solution:** True. Only under a certain circumstances. As we saw in class, if we assume that the variance is independent of the class then it is.

**Leave-one-out cross validation** is a special case of $k$-fold cross validation in which $k$ equals the number of examples $n$. Thus, at each iteration $i$, where $1 \leq i \leq n$, we use the $i$-th example for testing and the remaining $n-1$ examples for training. The leave-one-out cross validation error equals the average error over the $n$ iterations.

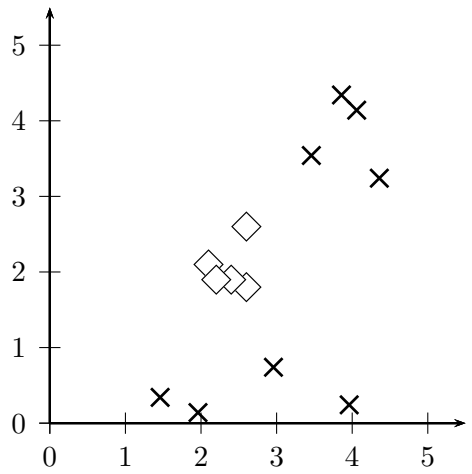
4. (3 points) The Leave-one-out cross validation error of 1-nearest neighbor classifier is always zero (assume that you are using the Euclidean distance as your distance measure). True or False. Explain your answer.

**Solution:** False. Consider a 1D dataset: $+-$ arranged on a line (the precise co-ordinates do not matter). Clearly, the leave one out cross validation error is not zero. The nearest point to the first point is a '$-$', while its class label is a '$+$'.

## 2 Classification [9 points]

For each of the following datasets, write alongside each classifier whether it will have zero training error on the dataset. Also, explain why in one sentence or by drawing a decision boundary. No credit if the explanation is incorrect.

1. (3 points)




Logistic Regression:

**Solution:** No. Not linearly separable.

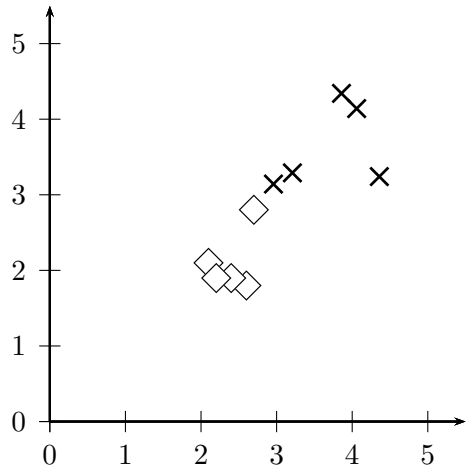
3-nearest neighbors:

**Solution:** Yes. At least two nearest points to each data point are of the same class as the data point.

SVMs with a quadratic kernel:

**Solution:** Yes. The Kernel will induce a circle around the squares.

2. (3 points)




Logistic Regression:

**Solution:** Yes. Linearly separable.

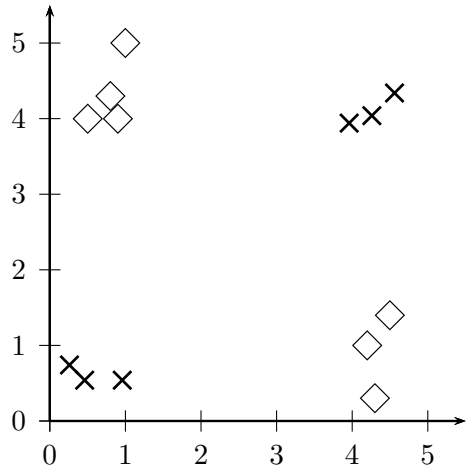
3-nearest neighbors:

**Solution:** No. The square at (3,3) has two crosses close to it.

SVMs with a quadratic kernel:

**Solution:** Yes. The kernel is not required; a linear SVM would have done the job.

3. (3 points)




Logistic Regression:

**Solution:** No. Not linearly separable.

3-nearest neighbors:

**Solution:** Yes. At least two nearest points to each data point are of the same class as the data point.

SVMs with a quadratic kernel:

**Solution:** Yes. An ellipse surrounding the squares or the crosses.

# 3 Linear Regression [8 points]

Consider a linear regression problem $y = w_1 x + w_0$, with a training set having $m$ examples $(x_1, y_1), \ldots, (x_m, y_m)$. Suppose that we wish to minimize the mean $fifth$ degree error (loss function) given by:

$$Loss = \frac{1}{m} \sum_{i=1}^{m} (y_i - w_1 x_i - w_0)^5$$

1. (4 points) Calculate the gradient with respect to the parameters $w_1$ and $w_0$. Hint: $\frac{dx^k}{dx} = kx^{k-1}$.

**Solution:** The gradient w.r.t. $w_1$ is:

$$\frac{1}{m} \sum_{i=1}^{m} 5(y_i - w_1 x_i - w_0)^4 (-x_i)$$

The gradient w.r.t. $w_0$ is:

$$\frac{1}{m} \sum_{i=1}^{m} 5(y_i - w_1 x_i - w_0)^4 (-1)$$

2. (4 points) Write down pseudo-code for online gradient descent for this problem.

**Solution:**

Initialize $w_0$ and $w_1$. Assume a value for $\eta$
Repeat Until Convergence
    For each example $(x_i, y_i)$ in the training set do
        $\Delta = 5(y_i - w_1 x_i - w_0)^4$
        $w_0 = w_0 + \eta \times \Delta$
        $w_1 = w_1 + \eta \times \Delta \times x_i$
    End For
End Repeat

# 4 Short Questions (12 points)

1. (4 points) Consider a d-bit parity problem in which the input is a $d$-dimensional Boolean vector $\mathbf{x} = (x_1, \ldots, x_d)$ and the output is 1 or $true$ if the number of 1's in $\mathbf{x}$ is odd and 0 or $false$ otherwise. Is the problem linearly separable for $d > 1$? Explain.

   **Solution:** The problem is not linearly separable for $d > 1$. It yields the XOR problem in the 2-d case.

2. (4 points) Estimate the number of nodes that a decision tree will require to model the d-bit parity function perfectly. Assume that your training data contains all possible, $2^d$ combinations and each split is on a value of a SINGLE Boolean variable. (**Hint**: Construct trees for $d = 3$ and $d = 4$ and compare the number of nodes in them to the maximum number of nodes that a decision tree can have).

   **Solution:** The size of the decision tree will be $O(2^d)$. Students should try constructing decision trees for $d = 2$, $d = 3$ and $d = 4$ before reaching this conclusion.

3. (4 points) Consider a non-uniform prior which assigns positive probability mass to each possible hypothesis. As the number of data points grows to infinity, the MLE estimate of a parameter approaches the MAP estimate to arbitrary precision. True of False. Explain.

   **Solution:** TRUE. When the number of data points grows to infinity, the data distribution dominates the prior distribution and the MLE equals MAP.

# 5 Support Vector Machines [10 points]

Consider the training data given below ($Y$ is the class variable)

| X | Y |
|---|---|
| -2 | 1 |
| -1 | -1 |
| 1 | -1 |
| 2 | 1 |

1. (3 points) Assume that you are using a linear SVM. Let $\alpha_1, \alpha_2, \alpha_3$ and $\alpha_4$ be the lagrangian mutlipliers for the four data points. Write the precise expression for the lagrangian dual optimization problem that needs to be solved in order to compute the values of $\alpha_1, \ldots, \alpha_4$ for the dataset given above.

**Solution:** The dual formulation is:

$$maximize \sum_{i=1}^{4} \alpha_i - \frac{1}{2} \sum_{i=1}^{4} \sum_{j=1}^{4} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

subject to the constraints: $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \geq 0$ and $\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 = 0$. The quantity $y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ for different values of $i$ and $j$ is given by the cells of the following matrix

$$\begin{pmatrix} 4 & -2 & 2 & -4 \\ -2 & 1 & -1 & 2 \\ 2 & -1 & 1 & -2 \\ -4 & 2 & -2 & 4 \end{pmatrix}$$

2. (2 points) Do you think, you will get zero training error on this dataset if you use linear SVMs (Yes or No)? Explain your answer.

**Solution:** No, because the data is not linearly separable. Plot it on a line and see for yourself.

The dataset is replicated here for convenience.

| X | Y |
|---|---|
| -2 | 1 |
| -1 | -1 |
| 1 | -1 |
| 2 | 1 |

3. (3 points) Now assume that you are using a quadratic kernel: $(1+x_i^T x_j)^2$. Again, let $\alpha_1, \alpha_2, \alpha_3$ and $\alpha_4$ be the lagrangian mutlipliers for the four data points. Write the precise expression for the lagrangian dual optimization problem that needs to be solved in order to compute the values of $\alpha_1, \ldots, \alpha_4$ for the dataset and the quadratic kernel given above.

**Solution:**

$$maximize \sum_{i=1}^{4} \alpha_i - \frac{1}{2} \sum_{i=1}^{4} \sum_{j=1}^{4} \alpha_i \alpha_j y_i y_j (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$$

subject to the constraints: $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \geq 0$ and $\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 = 0$. The quantity $y_i y_j (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$ for different values of $i$ and $j$ is given by the cells of the following matrix

$$\begin{pmatrix} 25 & -9 & -1 & 9 \\ -9 & 4 & 0 & -1 \\ -1 & 0 & 4 & -9 \\ 9 & -1 & -9 & 25 \end{pmatrix}$$

4. (2 points) Do you think, you will get zero training error on this dataset if you use the quadratic kernel (Yes or No)? Explain your answer.

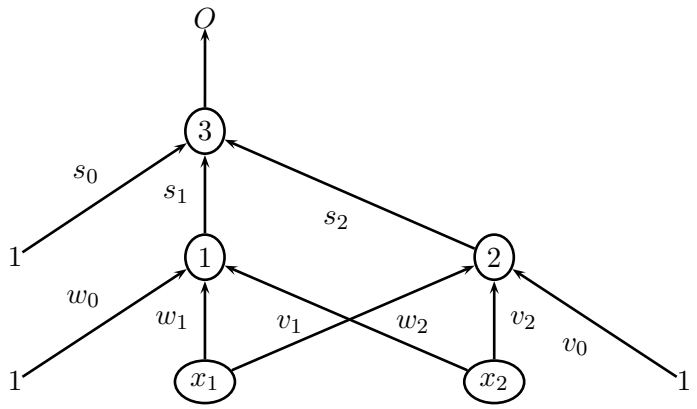
**Solution:** Yes, the quadratic kernel will correctly classify this data because it will add the feature $x^2$ to the dataset. You can plot the points in 2-D with this added dimension and see for yourself.

# 6 Neural networks [25 points]

1. (2 points) Is it always possible to express a neural network made up of only linear units without a hidden layer? Namely, given a neural network with several layers made up of only linear units, can you find a perceptron that is equivalent to the neural network? Explain your answer.

**Solution:** This is true because each linear unit will just attach a different co-efficient to each input co-ordinate and the output will still be a linear function of the input no matter how many hidden nodes/layers we use (assuming all hidden nodes are linear units).

For example, consider the neural network given below:




Note that the units numbered by 1, 2 and 3 are linear units. The output of the neural network is

$$O = s_0 + s_1(w_0 + w_1x_1 + w_2x_2) + s_2(v_0 + v_1x_1 + v_2x_2)$$

Rearranging the terms we get:

$$O = (s_0 + s_1w_0 + s_2v_0) + (s_1w_1 + s_2v_1)x_1 + (s_1w_2 + s_2v_2)x_2$$

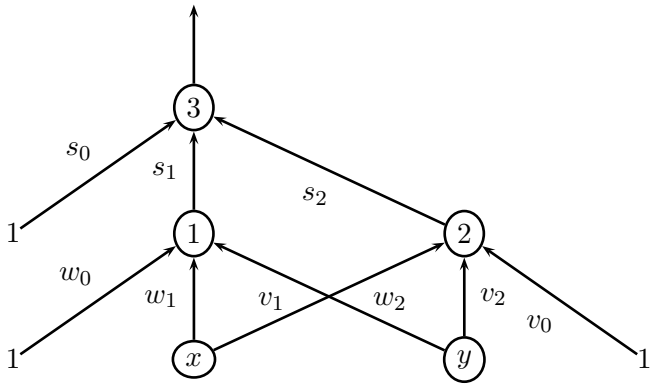
$O$ is a linear function in $x_1$ and $x_2$ and therefore it can be expressed using a Perceptron.

2. (8 points) Draw a neural network that represents the function $f(x, y)$ defined below:

| $x$ | $y$ | $f(x,y)$ |
|---|---|---|
| 0 | 0 | 10 |
| 0 | 1 | -5 |
| 1 | 0 | -5 |
| 1 | 1 | 10 |

Note that to get full credit, you have to write down the precise numeric weights (e.g., $-1$, $-0.5$, $+1$, etc.) as well as the precise units that you will use at each node (e.g., sigmoid, linear, simple threshold, tanh etc.).

**Solution:**




Nodes labeled by 1 and 2 are simple threshold units while the node labeled by 3 is a linear unit.

A possible setting of the weights is given below. Recall that the simple threshold unit is given by:

$$out = \begin{cases} +1 & \text{if } \sum_i w_i x_i > 0 \\ -1 & \text{otherwise} \end{cases}$$

$o_1$, which is the output of node labeled by 1 implements the following function

$$o_1 = \begin{cases} +1 & \text{if } \neg x \wedge \neg y \text{ is true} \\ -1 & \text{otherwise} \end{cases}$$

To achieve this, we can use $w_0 = 1$ and $w_1 = w_2 = -2$.

$o_2$, which is the output of node labeled by 2 implements the following function

$$o_2 = \begin{cases} +1 & \text{if } x \wedge y \text{ is true} \\ -1 & \text{otherwise} \end{cases}$$

To achieve this, we can use $v_0 = -2$ and $v_1 = v_2 = 1.5$.

$o_3$ implements the following function

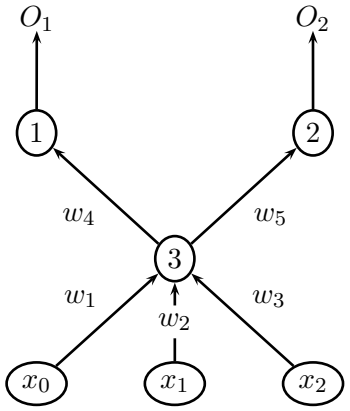
$$o_3 = \begin{cases} +10 & \text{if } o_1 = +1 \text{ or } o_2 = +1 \\ -5 & \text{otherwise} \end{cases}$$

Note that since 3 is a linear unit, we need it to obey the following constraints:
$s_0 + s_1 - s_2 = 10$ (if $o_1 = +1$ and $o_2 = -1$)
$s_0 - s_1 + s_2 = 10$ (if $o_1 = -1$ and $o_2 = +1$)
$s_0 - s_1 - s_2 = -5$ (if $o_1 = -1$ and $o_2 = -1$)

Notice that the case $o_1 = +1$ and $o_2 = +1$ can never happen.

A solution to the three equations is $s_0 = 10$ and $s_1 = s_2 = 7.5$.

Consider the neural network given below:




Assume that all internal nodes compute the sigmoid function. Write an explicit expression that shows how back propagation (applied to minimize the least squares error function) changes the values of $w_1$, $w_2$, $w_3$, $w_4$ and $w_5$ when the algorithm is given the example $x_1 = 0$, $x_2 = 1$, with the desired response $y_1 = 0$ and $y_2 = 1$ ($x_0 = 1$ is the bias term). Assume that the learning rate is $\alpha$ and that the current values of the weights are: $w_1 = 3$, $w_2 = 2$, $w_3 = 2$, $w_4 = 3$ a1nd $w_5 = 2$. Let $O_1$ and $O_2$ be the output of the output units 1 (which models $y_1$) and 2 (which models $y_2$) respectively . Let $O_3$ be the output of the hidden unit 3.

3. (5 points) Forward propagation. Write equations for $O_1$, $O_2$ and $O_3$ in terms of the given weights and example.

**Solution:** $O_3 = \sigma(w_1x_0 + w_2x_1 + w_3x_2) = \sigma(3 * 1 + 2 * 0 + 2 * 1) = \sigma(5)$
$O_2 = \sigma(w_5o_3) = \sigma(2\sigma(5))$
$O_1 = \sigma(w_4o_3) = \sigma(3\sigma(5))$

4. (5 points) Backward propagation. Write equations for $\delta_1$, $\delta_2$ and $\delta_3$ in terms of the given weights and example where $\delta_1$, $\delta_2$ and $\delta_3$ are the values propagated backwards by the units denoted by 1 and 2 and 3 respectively in the neural network.

**Solution:** $\delta_1 = (y_1 - o_1)o_1(1 - o_1) = o_1^3 - o_1^2$
$\delta_2 = (y_2 - o_2)o_2(1 - o_2) = o_2(1 - o_2)^2$
$\delta_3 = o_3(1 - o_3)(w_4\delta_1 + w_5\delta_2) = o_3(1 - o_3)(3\delta_1 + 2\delta_2)$

5. (5 points) Give an explicit expression for the new (updated) weights $w_1$, $w_2$, $w_3$, $w_4$ and $w_5$ after backward propagation.

**Solution:** Let $\eta$ denote the learning rate
$w_1 = w_1 + \eta\delta_3 x_0 = 3 + \eta\delta_3 \times 1 = 3 + \eta\delta_3$
$w_2 = w_2 + \eta\delta_3 x_1 = 2 + \eta\delta_3 \times 0 = 2$
$w_3 = w_3 + \eta\delta_3 x_2 = 2 + \eta\delta_3 \times 1 = 2 + \eta\delta_3$
$w_4 = w_4 + \eta\delta_1 o_3 = 3 + \eta\delta_1 o_3$
$w_5 = w_5 + \eta\delta_2 o_3 = 2 + \eta\delta_2 o_3$