CS446: Machine Learning **Fall 2016**

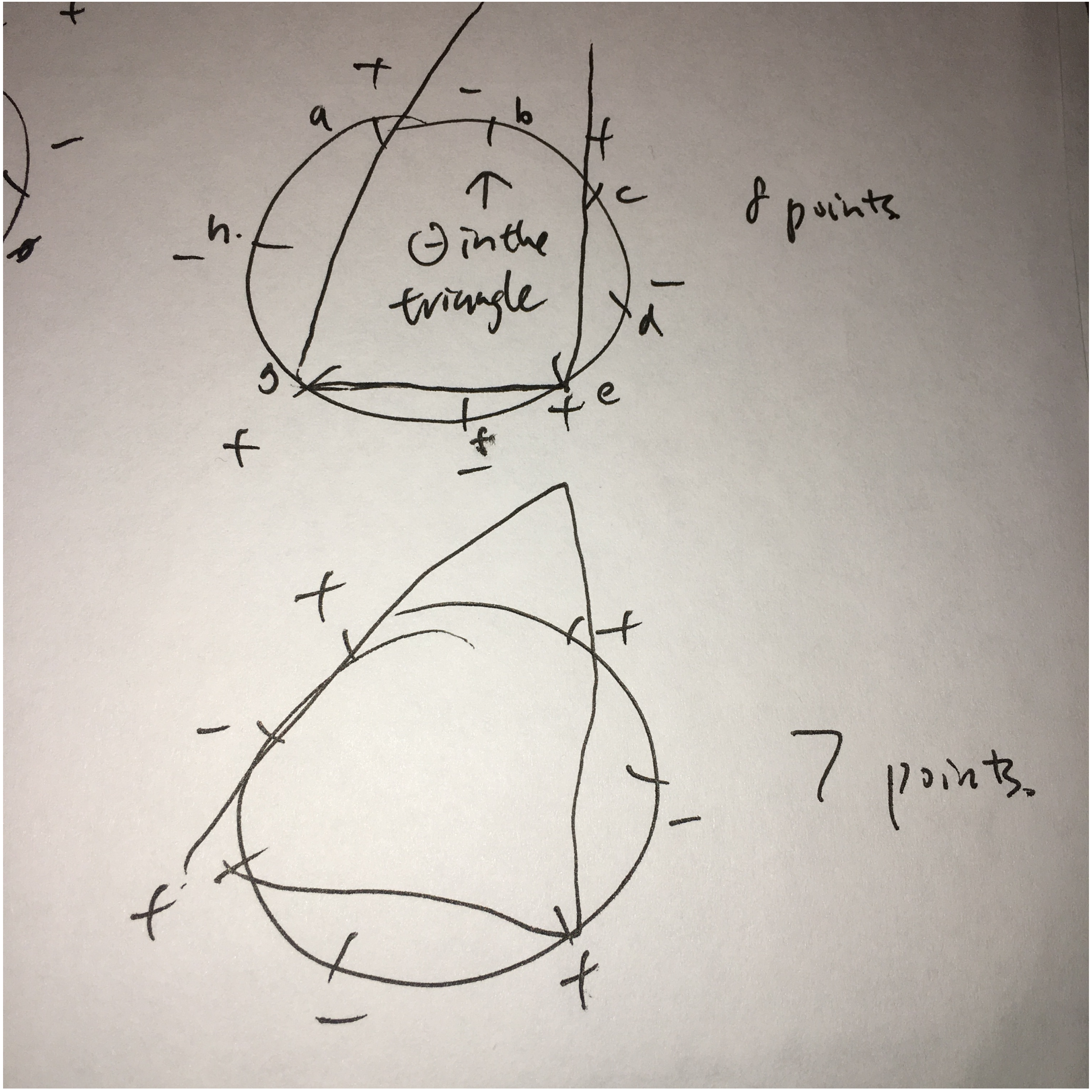
Problem Set 4

Yihao Zhang yzhng127 *Handed In: October 21, 2016*

1. Answer to problem 1

   a.

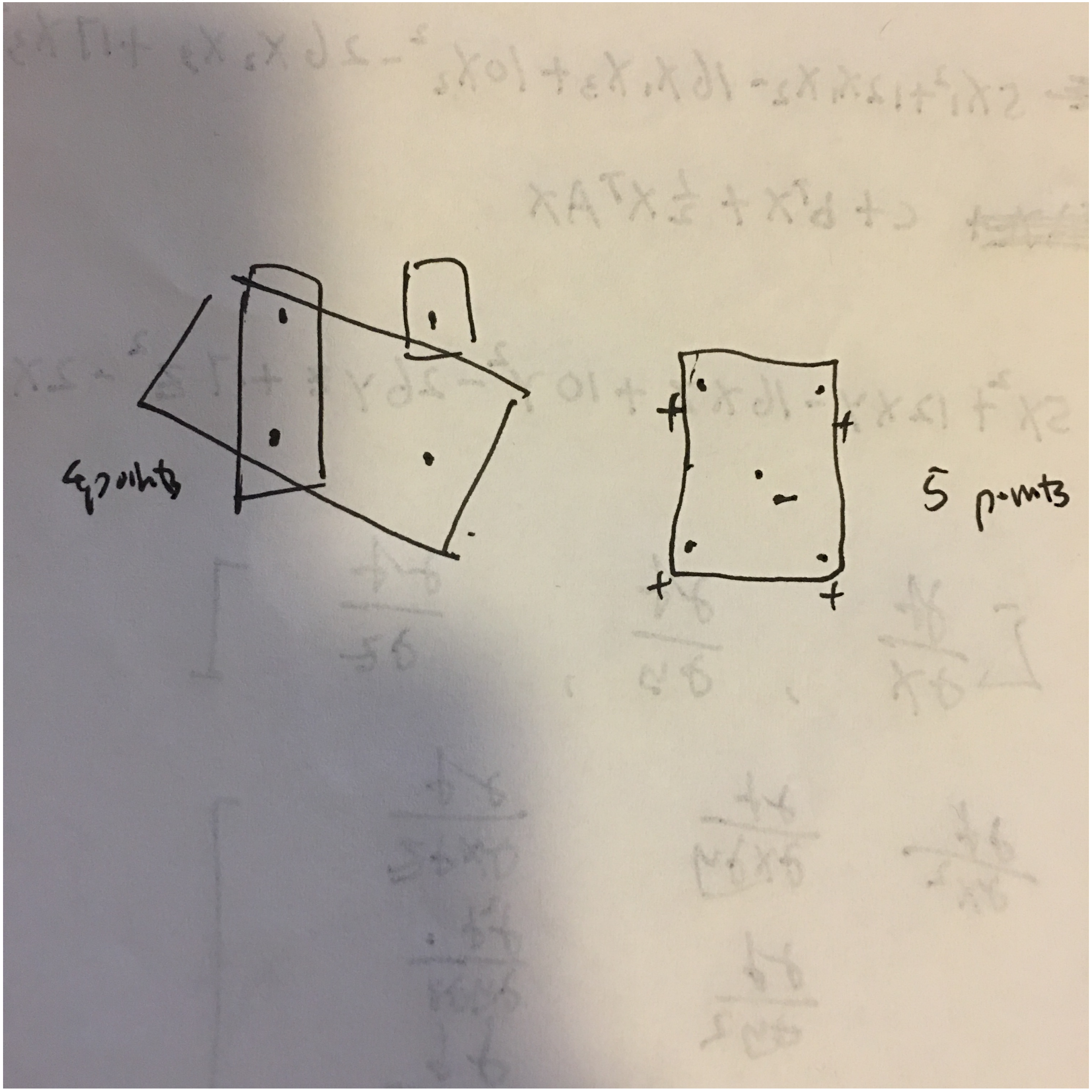
   The VC dimension of $\mathcal{H}$ is 7. By drawing on circles we can easily find that 7 points can be shattered by a triangle and any more points cannot. The graph below is an illustration of failed case using 8 points. And 7 points will be perfectly separated by a triangle.



   Let's say we have a set of 8 points(a,b,c,d,e,f,g,h), just like on the graph. And we mark them one by one, alternating positive and negative, like shown on the graph. We need, by definition of convex hull, include a,v,g,e in the triangle and exclude all others. However in this case b is also in the triangle. As we can see, if we want a convex hull by 3 lines, it is not possible to separate positive and negative. We will need one more line to go through a and c in order to take out the negative in the positive convex hull.

b.

If d = 2, we can have VC dimension of 4.Since if there's 5 points we cannot use rectangle to separate them. As shown in the picture below, there's a negative point in the convex hull of positive points.



Basically if there's d dimension, we are going to have d max values and d min values for each coordinate, which will give us 2d points. And that is the upper bound of points we can put. Thus the VC dimension is 2d.

2. Answer to problem 2

(a) Input: $S = ((\vec{x_1}, y_1), ..., (\vec{x_n}, y_n)), \vec{x_n}, x \in R^n, y \in 1, -1$
Instead of using $f(x) = sgn(w \cdot x + \theta)$, we should use a dual representation using α.
w will be $\sum_{1,m} r\alpha_i y_i x$

```
Set $\alpha$ to $\vec{0}$ of length $n$, where $n$ is the length of input examples.
For i from 1 to n
For each example in the training set $(x, y)$:
if $y((\sum_{i=1}^{n} \alpha_i y_i (x_i \cdot x)) + \theta) < 0$:
$\alpha_i = w\alpha_i + 1$ (i is instance index)
$\theta = \theta + y$
```

(b) In these terms, $K_1(x, z) = x^T z$ is kernel, because the feature map $\phi_1(x)$ is just the same feature map, so that $K_1 = \langle \phi_1(x), \phi_1(z) \rangle$.
Following theorem adopted from:`http://l2r.cs.illinois.edu/~danr/Teaching/CS446-16/Lectures/04-LecOnline-P2.pdf`
Theorem: Let K_1 and K_2 be valid Kernels over $X * X$, $X \in R^N$, $\alpha 0$, $0 \lambda 1$, f a real-valued function on X, M:X $\rightarrow$ R^m ¡m with a kernel K_3 over $R^m \times R^m$, and K a symmetric positive semi-definite matrix. Then the following functions are valid Kernels:
1.$K(x, z) = \lambda K1(x, z) + (1 - \lambda) K2(x, z)$
2.$K(x, z) = \alpha K1(x, z)$
3.$K(x, z) = K_1(x, z) K_2(x, z)$

Before proving that $K(x, z)$ is a valid kernel, I'll prove one more property of kernels.
Using theorem 3, $(\vec{x}^T \vec{z})^2 = K(x, z) * K(x, z)$, so it is a valid Kernel
$(\vec{x}^T \vec{z})^3 = (\vec{x}^T \vec{z})^2 * (\vec{x}^T \vec{z}) = K_1(x, z) * K_2(x, z)$, so it is a valid Kernel
Using theorem 2, $400(\vec{x}^T \vec{z})^2$ and $100x^T z$ is a valid Kernel.
Using theorem 1, the sum is a valid Kernel. So $1(x^T z)^3 + 400(x^T z)^2 + 100x^T z$ is a valid Kernel as whole.

(c) C here is monotone conjunctions. c(x)c(z) = 1 can implies both c(x) and c(z) are true. We can count the same elements in x and z that are 1, assuming that we have result j¡k, because the upper bound is k. Use the same function derived from lecture. `http://l2r.cs.illinois.edu/~danr/Teaching/CS446-16/Lectures/04-LecOnline-P2.pdf`
As a result we are having $K = \sum_{j=0}^{k} \binom{same(x,z)}{j}$

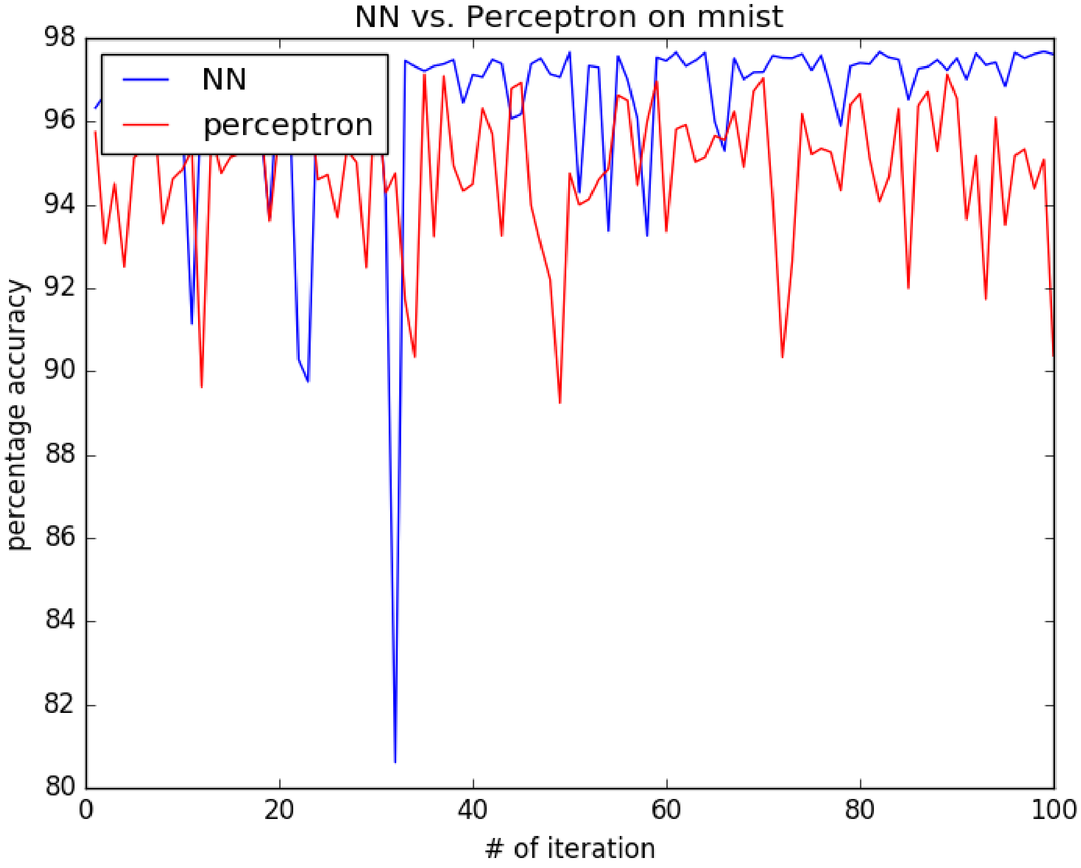
(d) In class we have proved that the mistake bound is $\dfrac{R^2}{\gamma^2}$

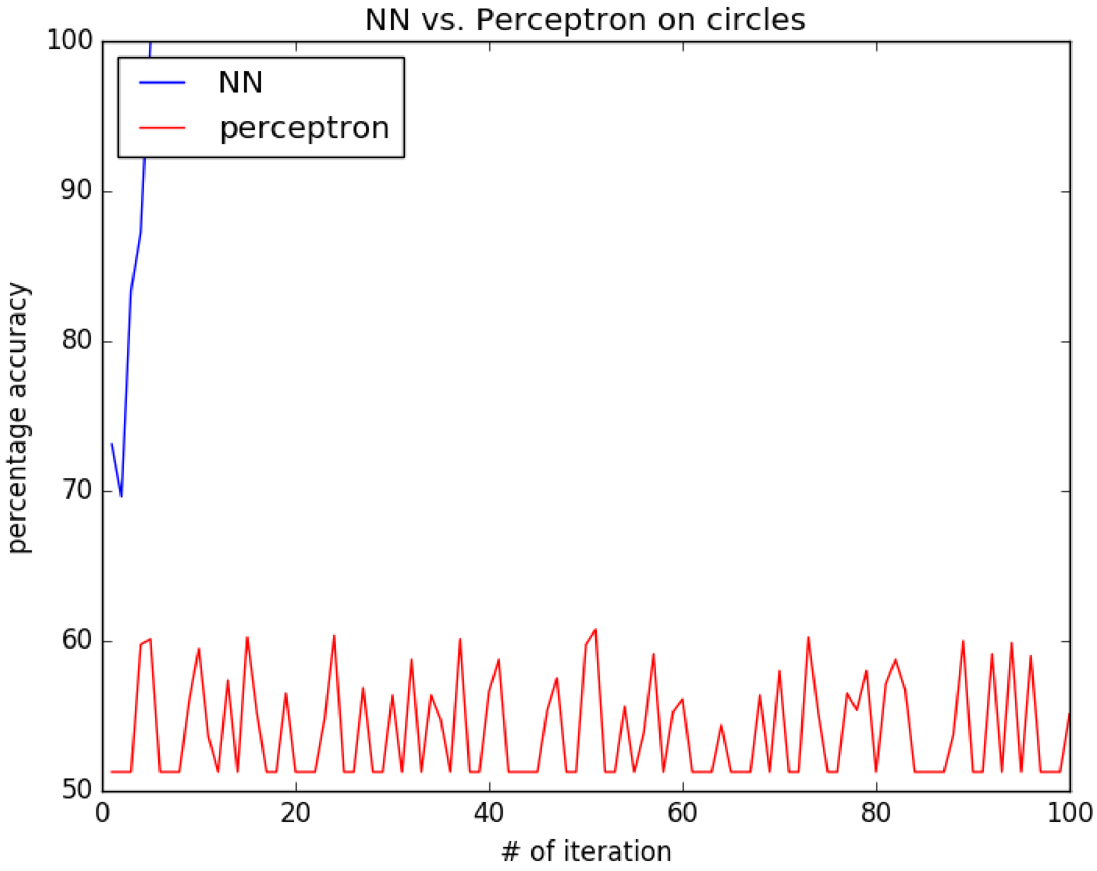
R will be the new feature space, which is $\sqrt{\binom{\sum_{j=0}^{k} n}{j+1}}$,where n is the size of x, where +1 comes from θ.
For γ, we want to use gamma to constrain the weight vector x', so that x' will be optimal.

3. Answer to problem 3

(a) The error will be: $\dfrac{1}{2}\sum(t_k - o_k)^2$
Activation function being: f(x) = max(0,x)
R is the learning rate
$\frac{\partial E}{\partial o} = (t - o)$
$\frac{\partial f(x)}{\partial x} = \; 1 \;\; if x > 0 \; and \; 0 \; if x < 0$
For hidden units:$\delta_j = \sum_{l \in L} \delta_l * w_{jl}, \;\; if \;\; f(x) > 1, otherwise 0$
For output unites:$\delta_j = t_j - o_j, \;\; if \;\; f(x) > 1, otherwise 0$
For hidden units: $\Delta w_{ij} = R(\sum -\delta_k * w_{jk}) * x_{ij}, \;\; if \;\; f(x) > 1, otherwise 0$
For output unites: $\Delta w_{ij} = R(t_j - o_j), \;\; if \;\; f(x) > 1, otherwise 0$

(b) For this problem NN does well for circles. The reason is perceptron algorithm cannot separate circle data. So the accuracy will always be oscillating in the middle. It will never converge for perceptron. This is as expected. For mnist the accuracy is always high after a few iterations. This is also expected. After parameter tuning the speed of convergence is pretty quick for neural net. Overall neural net has better performance than linear separator. I have attached graph and tuning result in this report.


NN vs. Perceptron on mnist
NN
perceptron
percentage accuracy
# of iteration
98
96
94
92
90
88
86
84
82
80
0
20
40
60
80
100


NN vs. Perceptron on circles
NN
perceptron
percentage accuracy
100
90
80
70
60
50
0
20
40
60
80
100
# of iteration