CSE 512: Homework 3 Due Wednesday October 16

1. **(2 pts)** *Logistic regression for Binary MNIST*

   (a) What is the gradient of the logistic loss function

$$f(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\log(\sigma(y_i x_i^T\theta)), \quad \sigma(s) = \frac{1}{1+e^{-s}}$$

   where $y_i \in \{-1, 1\}$? (Hint: Check out problem 1 again.)

   Ans. To reduce notation, we write $z_i = y_i x_i$, and the matrix $Z = [z_1, z_2, ...z_m]^T$ has each vector z_i as a row. We can work out the derivatives of the sigmoid function, and show that

$$\sigma'(s) = \sigma(s)(1-\sigma(s)).$$

   After that, standard calculus rules gets us to

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = -\frac{1}{m}\sum_{i=1}^{m}\frac{\sigma(z_i^T\theta)(1-\sigma(z_i^T\theta))}{\sigma(z_i^T\theta)} z_i.$$

   This yields

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{1}{m}\sum_{i=1}^{m}(\sigma(z_i^T\theta)-1)z_i.$$

   which we can write succinctly as

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{1}{m}Z^T(d - \mathbf{1})$$

   for $d_i = \sigma(z_i^T\theta)$, $i = 1, ..., m$.

   (b) **Coding gradient descent.** Do **not** use `scikit-learn` or other built in tools for this exercise. Please only use the packages that are already imported in the notebook. [1]

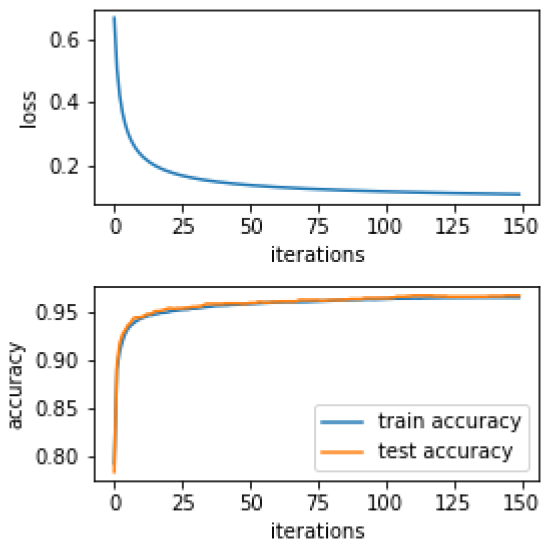
   - Open `mnist_logreg.ipynb`. We will use logistic regression to diffrentiate 4's from 9's, a notoriously tricky problem. Run the first box to see what the data looks like.
   - In the second box I have set up the problem for you by pulling out the train and test set, selecting out only the data samples related to 4's and 9's. I have not altered the data in any other way. While other normalization tricks will help you improve the accuracy, for the purposes of this exercise we will forgo them, so that it's easy to compare everyone's solutions.
   - Fill in the next box by providing code that will return the loss function value and the gradient. Make sure that everything is normalized, e.g. don't forget the $1/m$ term in the front of our loss function. Run the script. If done correctly, you should see

```
45.192, 12343.177
```

   - Write a script that returns the classification accuracy given θ.
   - Use gradient descent to minimize the logistic loss for this classification problem. Use a step size of 10^{-6}.
   - **(1.5 pts)** Run for 1500 iterations. In your report, give the plot of the train loss and train/test misclassification rate, plotted as a function of *iterations*. Report the final train and test accuracy values.

     Ans. Train accuracy: 96.58%, test accuracy: 96.83%. Plot:

[1] This is to test your understanding of the basic machine learning concepts, like calculating accuracy and logistic loss; in the future you can use whatever tools you'd like.



(c) **Coding stochastic gradient descent.** Do **not** use `scikit-learn` or other built in tools for this exercise. Please only use the packages that are already imported in the notebook. Now, fill in the next box a function that takes in θ and a minibatch $\mathcal{B}$ as either a list of numbers or as an np.array, and returns the *minibatch gradient*

$$\nabla_{\mathcal{B}} f(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla f_i(\theta)$$

where $f_i(\theta)$ is the contribution to the gradient from datapoint i:

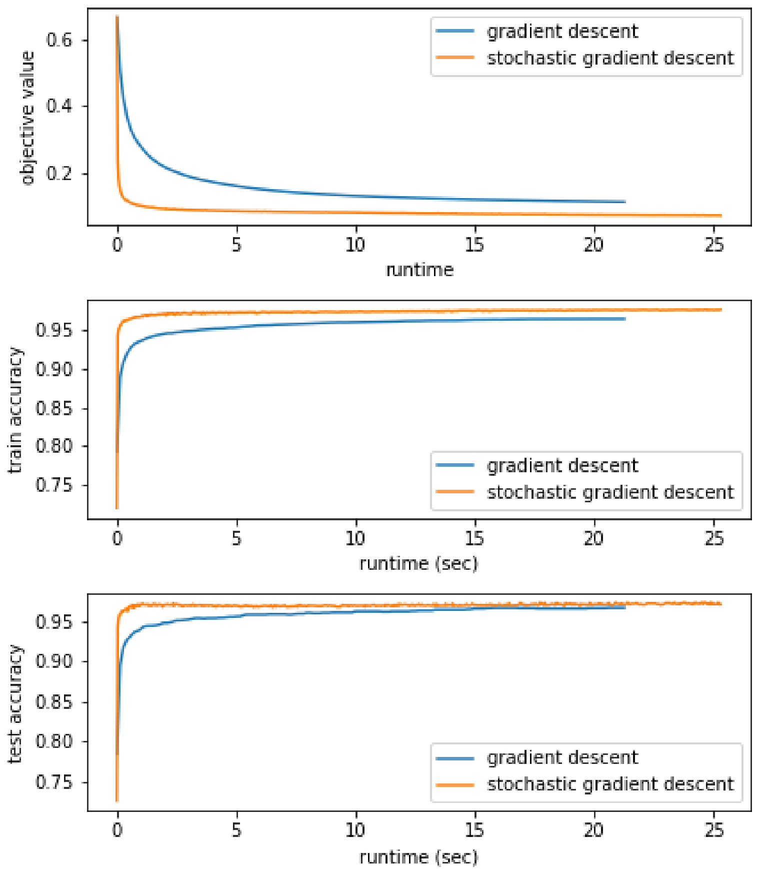
$$f_i = -\log(\sigma(y_i x_i^T \theta)).$$

Run the script. If done correctly, you should see the number 5803.5 printed out.

(d) Write a script to run stochastic gradient descent over logistic regression. When coding up the minibatching, make sure you cycle through an entire training set once before moving on to the next epoch. Additionally, use `time()` to record the runtime, and compare the performance of gradient descent and stochastic gradient descent, using a minibatch size of 50 data samples and running for 50000 iterations. Return a plot that compares the objective loss, train accuracy, and test accuracy between the two optimization methods, as a function of *runtime*. Comment on the pros and cons of the two methods.

Important Remember that calculating the loss function and train/test accuracy requires making *full passes* through the data. If you do this at each iteration, you will not see any runtime benefit between stochastic gradient descent and gradient descent. Therefore I recommend you log these values every 10 iterations for gradient descent, and every 100 iterations for stochastic gradient descent.

Ans.



The computational runtime benefit of using minibatched gradients is clear here, even though our original full batch gradient descent is not that computationally burdensome. A possible con of using the SGD vs the full batch GD approach is that picking the minibatch size might not be obvious. Also, although it is not obvious in these plots, running SGD with a constant step size cannot actually get to the exact minimum loss, but rather wanders in a small region of error. However, in this case, the small region of error is so tiny that it is imperceptible, and certainly does not affect the test accuracy.

2. **(2 pts) Logistic regression and the Fisher Information Matrix.** Recall that the training of logistic regression models is equivalent to minimizing the following convex optimization problem

$$\underset{\theta\in\mathbb{R}^n}{\text{minimize}} \quad f(\theta) := \frac{1}{m}\sum_{i=1}^{m}\log(1+\exp(-y_i x_i^T\theta)) \qquad \text{(LogReg)}$$

for data features $x_1, ..., x_m \in \mathbb{R}^n$ and binary labels $y_1, ..., y_m \in \{-1, 1\}$.

(a) **(.5 pt)** Derive the gradient of f with respect to θ. Hint: It may help to use the sigmoid function $\sigma(s) = (1+\exp(-s))^{-1}$.
Hint 2: It may help also to use linearity of gradients, and first just work with subproblems

$$f_i(\theta) = \log(1+\exp(-y_i x_i^T\theta)).$$

Ans. Using chain rule,

$$\nabla_\theta f_i(\theta) = \nabla_\theta(-\log(\sigma(y_i x_i^T\theta)) = \left(\frac{\partial}{\partial s}(-\log(\sigma(s))\right)|_{s=y_i x_i^T\theta} \cdot y_i x_i$$

Using the trick (you can derive this yourself, and would be helpful just to memorize it) $\sigma'(s) = (1-\sigma(s))\sigma(s) = \sigma(-s)\sigma(s)$, we can simplify to

$$\nabla_\theta f_i(\theta) = (\sigma(y_i x_i^T \theta) - 1) y_i x_i.$$

Overall, the gradient of f can be written as

$$\nabla_\theta f(\theta) = \frac{1}{m} \sum_{i=1}^m (\sigma(y_i x_i^T \theta) - 1) y_i x_i$$

(b) **(0.5 pt)** Derive the Hessian of f w.r.t. θ. Hint: Don't attempt this without first solving part (a) successfully.
Ans. Note that in the gradient of f_i, only the first scalar term depends on θ. Thus, if we write $d_i(\theta) = \sigma(y_i x_i^T \theta)$, then the Hessian is simply

$$\nabla_\theta^2 f_i(\theta) = y_i \nabla d_i(\theta) x_i^T.$$

Furthermore,

$$\nabla d_i(\theta) = \sigma(y_i x_i^T \theta)(1 - \sigma(y_i x_i^T \theta) y_i x_i$$

and recall that $y_i \in \{-1, 1\}$, so $y_i^2 = 1$. Finally, we may write

$$\nabla_\theta^2 f_i(\theta) = \sigma(y_i x_i^T \theta)(1 - \sigma(y_i x_i^T \theta) x_i x_i^T$$

and in aggregate,

$$\nabla_\theta^2 f(\theta) = \frac{1}{m} \sum_{i=1}^m \sigma(y_i x_i^T \theta)(1 - \sigma(y_i x_i^T \theta) x_i x_i^T$$

(c) **(1 pt)** The Fisher information matrix is often used to describe the amount of information about a paramether θ that is carried by observables (x_i, y_i). Formally, it is written as

$$I(\theta) = \mathbb{E}_{x,y} \left[(\nabla_\theta \log(p_\theta(x,y))) (\nabla_\theta \log(p_\theta(x,y)))^T | \theta \right]$$

where $p_\theta(x,y) = \mathbf{Pr}(y|x,\theta)$. This is the logistic model. Show that the Fisher information matrix is equivalent to the Hessian of f in (LogReg), if x_i, y_i were drawn i.i.d. from this distribution, in the limit of infinite data ($m \to +\infty$).
Note that this implies that, for logistic models, *we can identify the Hessian using only gradient information,* which may be a significant computational benefit.
Hint: It helps to recall that $1 - \sigma(s) = \sigma(-s)$.
Hint 2: Recall the *exact* formula for an expectation of a continuous distribution. Don't try to guess this part.
Ans. Using this function of x and y, the term inside the gradient $\log(p_\theta(x,y))$ should look very familar, and we may write

$$\nabla_\theta \log(p_\theta(x,y)) = (1 - \sigma(yx^T\theta))yx = \sigma(-yx^T\theta)yx$$

and therefore

$$(\nabla_\theta \log(p_\theta(x,y)))(\nabla_\theta \log(p_\theta(x,y)))^T = \sigma(-yx^T\theta)^2 xx^T.$$

Now, we apply the expectations one at a time. Using $p(y|x) = \sigma(yx^T\theta)$,

$$\mathbb{E}_{y|x}[\sigma(-yx^T\theta)^2 xx^T] = \sum_{y \in \{-1,1\}} \sigma(yx^T\theta)\sigma(-yx^T\theta)^2 xx^T$$

Note that only the scalar terms depend on y. Specifically, we may simplify

$$\sum_{y \in \{-1,1\}} \sigma(yx^T\theta)\sigma(-yx^T\theta)^2 = \sigma(x^T\theta)\sigma(-x^T\theta) \underbrace{(\sigma(x^T\theta) + \sigma(-x^T\theta))}_{=1}$$

leaving

$$\mathbb{E}_{y|x}[\sigma(-yx^T\theta)^2 xx^T] = \sigma(yx^T\theta)\sigma(-yx^T\theta) xx^T.$$

Finally, since $\mathbb{E}_{x,y}[f(x,y)] = \mathbb{E}_x[\mathbb{E}_{y|x}[f(x,y)]]$, we have

$$\mathbb{E}_{x,x}[\sigma(-yx^T\theta)^2 xx^T] = \int_x p(x)\sigma(yx^T\theta)\sigma(-yx^T\theta) xx^T.$$

which is the limit $m \to +\infty$ of the empirical averaging in (LogReg).

3. **Generalization and linear regression. (2pts)** Consider a linear model

$$y_i = x_i^T \theta + z_i, \qquad z_i \sim \mathcal{N}(0,1)$$

Here, $x_i \in \mathbb{R}^n$ where n is very large. After accruing m training pairs (x_i, y_i) for $i = 1, ..., m$, I form a data matrix and target vector

$$X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_m^T \end{bmatrix}, \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}.$$

Now I will compose a *truncated* linear model by minimizing the following regression loss

$$\hat{\theta}_t = \underset{\theta}{\text{argmin}} \; \|X_t\theta - y\|_2^2, \qquad t \ll n$$

where X_t contains only the first t columns of X, and θ_t is a vector with length t. Now I have a new point, $\tilde{x}$, and my new linear model will then perform inference on future data points as

$$\hat{y} = \tilde{x}_t^T \hat{\theta}_t$$

where $\hat{\theta}_t$ are the learned truncated parameters, and $\tilde{x}_t$ contains only the first t elements of x.

Assume that the number of datapoints $m \gg n$ and each truncated datasample is linearly independent, even if t is very small. Also, assume $X_t^T X_t$ is invertible.

(a) What is the bias of a future prediction $\tilde{x}_t^T \hat{\theta}_t$?
Ans.
The solution to this linear model is

$$\hat{\theta}_t = (X_t^T X_t)^{-1} X_t^T y = (X_t^T X_t)^{-1} X_t^T (X_t \theta_t + X_r \theta_r + z)$$

where the subscript r indicates all the indices removed (e.g. $r = \{i : t < i \leq n\}$). Then, simplifying,

$$\hat{\theta}_t = \theta_t + (X_t^T X_t)^{-1} X_t^T X_r \theta_{\bar{n}'} + (X_t^T X_t)^{-1} X_t^T z$$

and since z is the only random vector,

$$\mathbb{E}[\hat{\theta}_t] = \theta_t + (X_t^T X_t)^{-1} X_t^T X_r \theta_r.$$

Thus the bias is

$$\mathbb{E}[\tilde{x}_t^T \hat{\theta}_t] - \tilde{x}_t^T \theta_t = \tilde{x}_t^T (X_t^T X_t)^{-1} X_t^T X_r \theta_r - \tilde{x}_t^T \theta_r.$$

(b) In terms of big-O notation, what is the bias' dependence on t? (linear? quadratic? exponential?) Assume that every value in X has about the same magnitude.
Ans. First, note that if $t = n$, then the bias is 0. If $t = 1$, then $\|\hat{\theta}_r\|_2 = O(n)$, and more generally, $\|\hat{\theta}_r\|_2 = O(n - t)$. Since

$$\|x_t^T (X_t^T X_t)^{-1} X_t^T X_r \hat{\theta}_r\|_2 \leq \|x_t\|_2 \|(X_t^T X_t)^{-1} X_t^T\|_2 \|X_r \hat{\theta}_r\|_2$$

where since $(X_t^T X_t)^{-1} X_t^T$ is a projection matrix, then its norm is ≤ 1. So, overall,

$$\|\mathbb{E}[\tilde{x}_t^T \hat{\theta}_t] - \tilde{x}_t^T \theta_t\|_2 = O(t(n-t) + (n-t)) = O(n-t).$$

That is, the more the truncation, the higher the bias.

(c) What is the variance of $\tilde{x}_t^T \hat{\theta}_t$?
Ans. Since $\theta_t + (X_t^T X_t)^{-1} X_t^T X_r \theta_r$ is a constant term, we only need to worry about the last term, e.g.

$$\textbf{var}(x_t^T \hat{\theta}_t) = \textbf{var}(\tilde{x}_t^T (X_t^T X_t)^{-1} X_t^T z_t) = \|X_t (X_t^T X_t)^{-1} \tilde{x}_t\|_2.$$

That is, the less the truncation, the higher the variance.

(d) In terms of big-O notation, what is the variance's dependence on t? (linear? quadratic? exponential?)
Ans. $\|X_t (X_t^T X_t)^{-1} \tilde{x}_t\|_2 = O(\|\tilde{x}_t\|_2) = O(t)$.

4. **Logistic regression fundamentals. (2pts)** Recall that in logistic regression, our goal is to minimize the following loss function

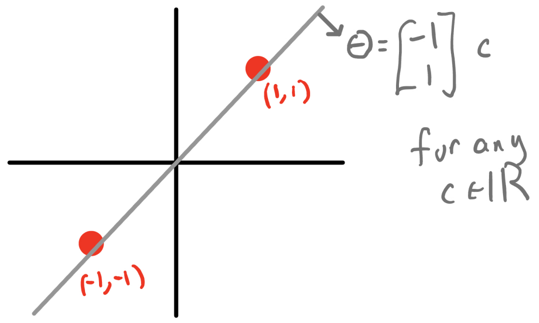
$$\underset{\theta}{\text{minimize}} \quad -\sum_{i=1}^{m} \log(\sigma(y_i x_i^T \theta)) \tag{LogLoss}$$

In the following cases, there exists an intuitive answer for what the best θ. In each case, find what that θ is using whatever method you choose (hint, draw a picture). Then, argue why it is optimal by showing that it minimizes (LogLoss) (or that it does in a limit).

(a)

i	x	y
1	$(1,1)$	1
2	$(-1,-1)$	1

Ans.



$\theta = \begin{bmatrix}-1\\1\end{bmatrix} c$
for any $c \in \mathbb{R}$
(1,1)
(-1,-1)

Intuitively, we can see that this is an "impossible situation" where every line will get one point right, and one point wrong, except $\theta = c \cdot (-1, 1)$ which labels both points as "uncertain". In this case, $\sigma(y_i x_i^T \theta) = 0$ for $i = 1$ and $i = 2$, and the gradient of the loss becomes

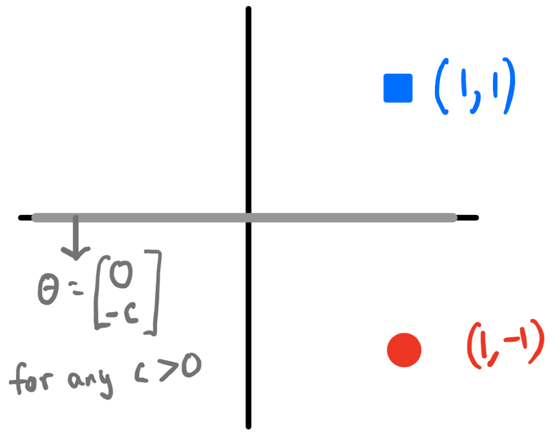
$$\nabla f(\theta) = (y_1 x_1 + y_2 x_2)(0 - 1) = 0.$$

So, while this classifier doesn't do a great job, it is indeed optimal.

(b)

i	x	y
1	$(1,-1)$	1
2	$(1,1)$	-1

Ans.



Here, we can perfectly classify these two points with a line $\theta = (0, -c)$ for any $c > 0$. In this case, $\sigma(y_i x_i^T \theta) = \sigma(c)$ for $i = 1$ and $i = 2$, and the gradient of the loss becomes

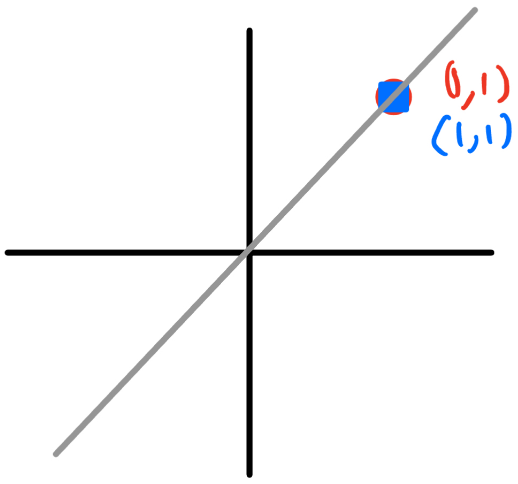
$$\nabla f(\theta) = (y_1 x_1 + y_2 x_2)(0 - \sigma(c)) = \sigma(c) \begin{bmatrix}0\\1\end{bmatrix}.$$

However, here we can see that the gradient *isn't 0* and instead will only be 0 if $c \to \infty$, which forces $\sigma(c) \to 1$ and $\nabla f(\theta) \to 0$.

(c)

i	x	y
1	$(1,1)$	1
2	$(1,1)$	-1

Ans.

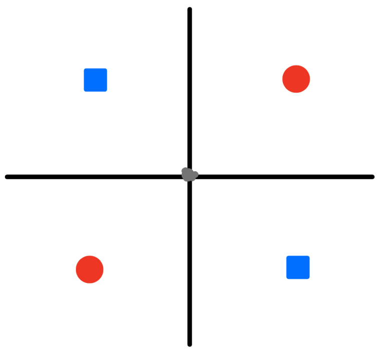


Again, this problem is impossible to get right on both points, so we try to compromise by drawing a point right through both points, with $\theta = [1, -1] \cdot c$ again. In this case, $\sigma(y_i x_i^T \theta) = 0$ for $i = 1, 2$ and we again see that $\nabla f(\theta) = (0 - 1) \cdot \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0$.

(d)

i	x	y
1	$(1, 1)$	1
2	$(1, -1)$	-1
3	$(-1, 1)$	-1
4	$(-1, -1)$	1

Ans.

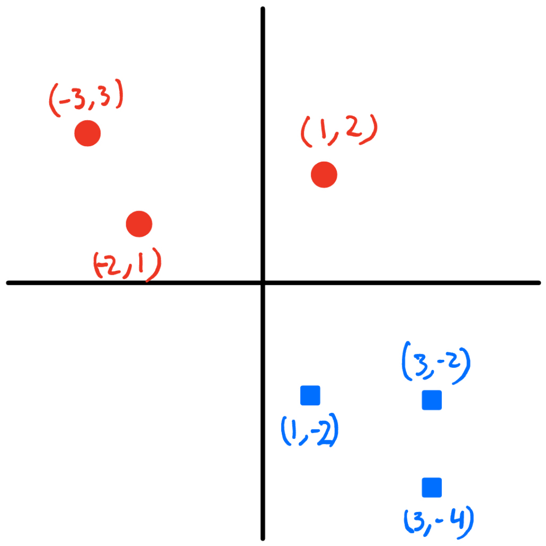


$\theta = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

This problem is actually impossible to make any progress. It turns out that here, the best solution is to pick $\theta = (0, 0)$ and then

$$\nabla f(\theta) = ((1, 1) + (1, -1) + (-1, 1) + (-1, -1)) \cdot \sigma(0) = 0$$

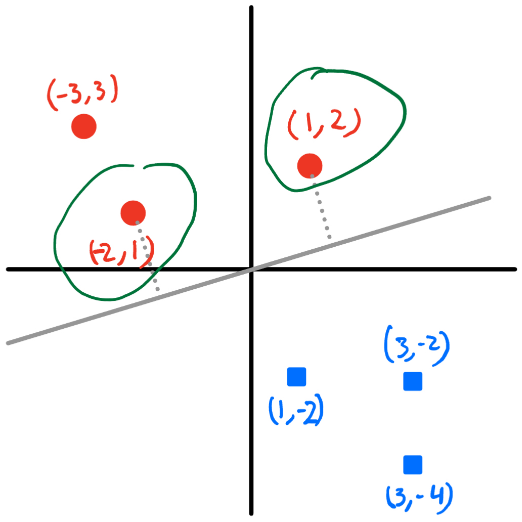
5. **Margins and SVM (2pts)**



Consider the problem of separating the red circles and blue squares from above. We will use a support vector machine with no bias, e.g. the discriminator must go through the origin.

(a) Draw a discriminator that goes through the origin, perfectly separates the two sets, and visually maximizes the distance between the discriminator and any datapoint. Circle the two points closest to the discriminator, which form the support vectors.

Ans.



(b) Deduce the exact values in θ, such that the red circles get labels $+1$ and the blue squares get labels -1

Ans. Based on the two points I circled, the line that is equally close to both points must have the same slope, so the slope should be $1/3$. That is, either $\theta = \begin{bmatrix} 1 \\ -3 \end{bmatrix}$ or $\theta = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$. To ensure that the red dots get positive labels, then it must be that $\theta = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$.

(c) Compute the margin distance of each point to the discriminator you proposed. You can leave your answer in radicals.

Ans. Using this choice of θ, I can now write out each distance as $|x_i^T\theta|/\|\theta\|_2 = |x_i^T\theta|/\sqrt{10}$:

point	margin distance
(1,2)	$5/\sqrt{10}$
(-3,3)	$12/\sqrt{10}$
(-2,1)	$5/\sqrt{10}$
(3,-2)	$9/\sqrt{10}$
(1,-2)	$7/\sqrt{10}$
(3,-4)	$15/\sqrt{10}$