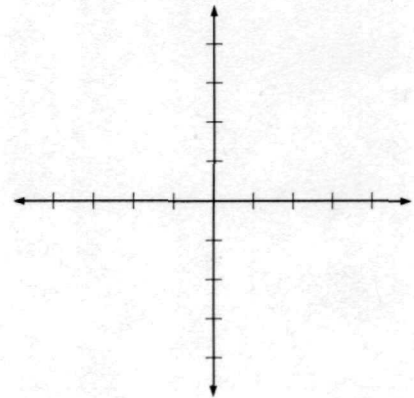
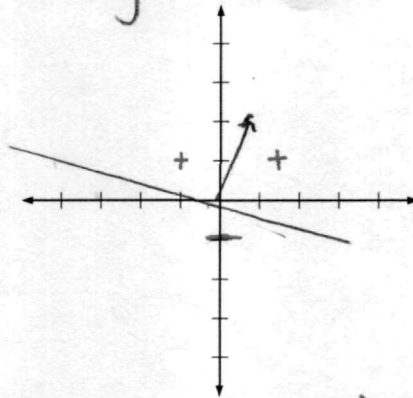
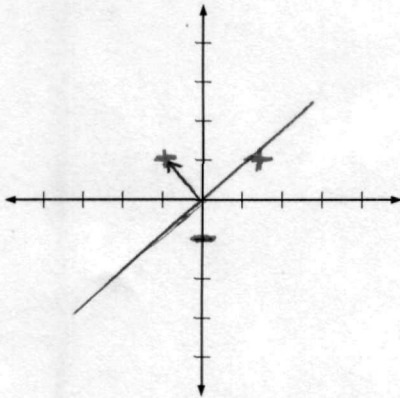
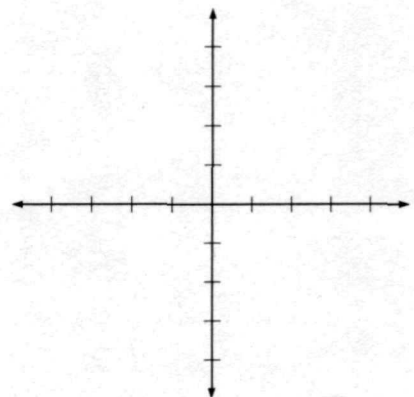
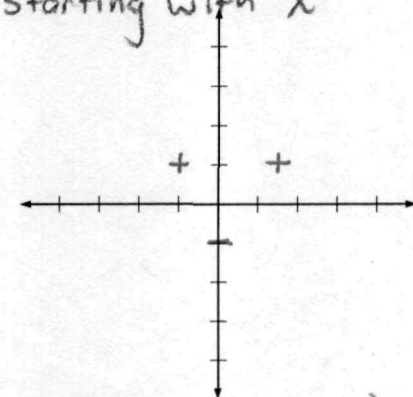
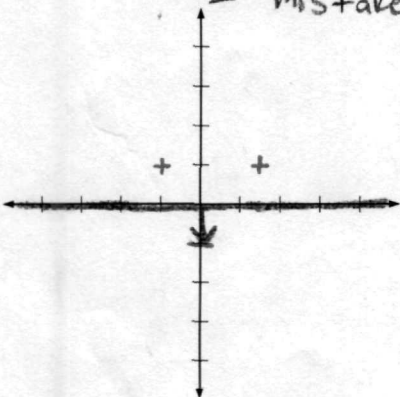


6) a)

2 mistakes starting with  $x^{(1)}$

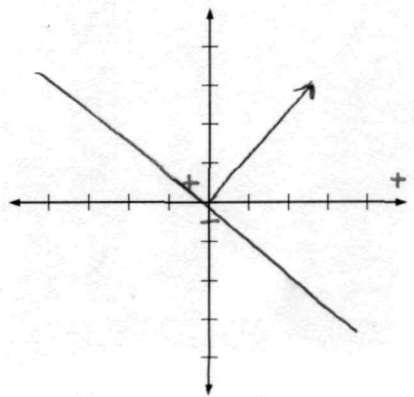
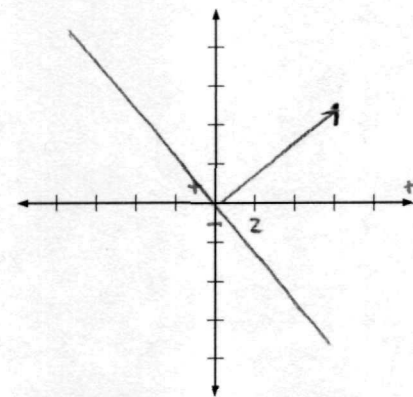
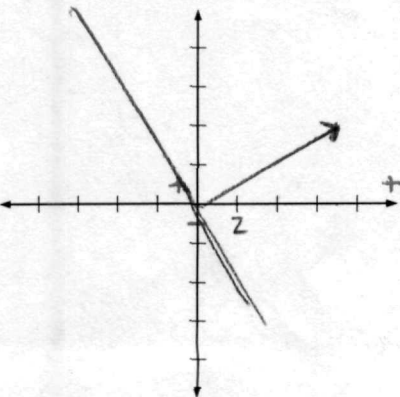
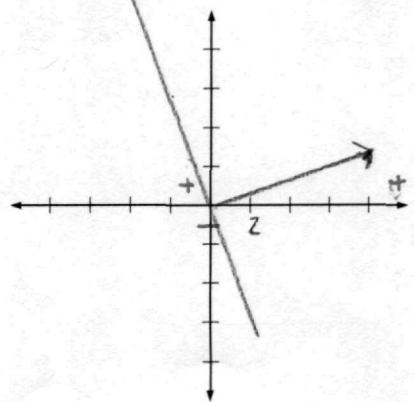
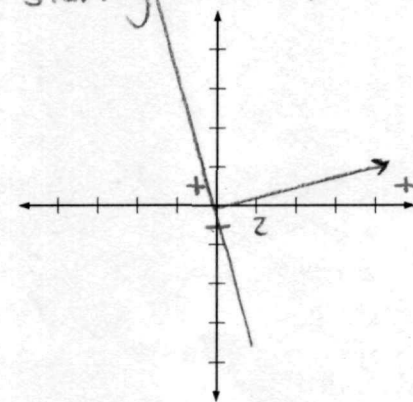
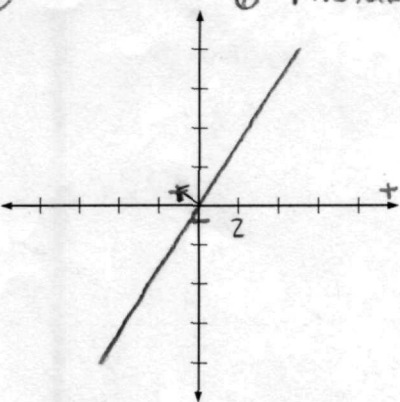


1 mistake starting with  $x^{(2)}$

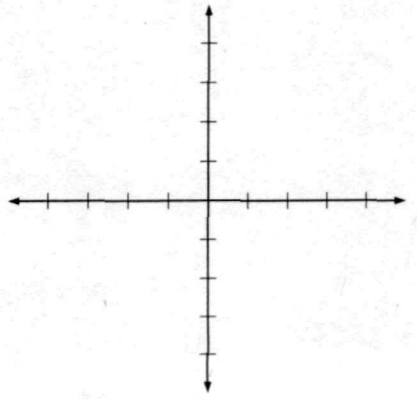
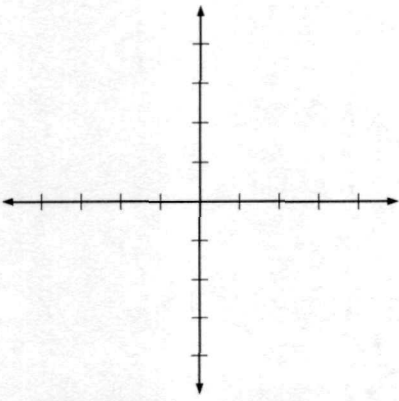
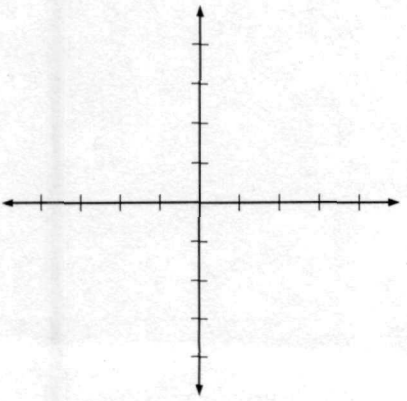
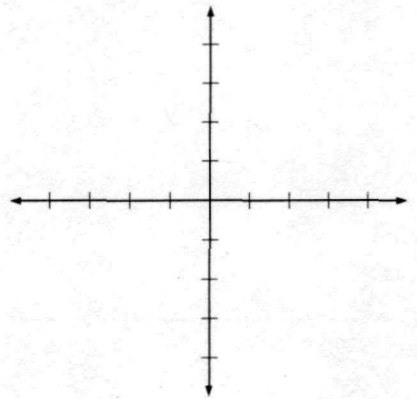
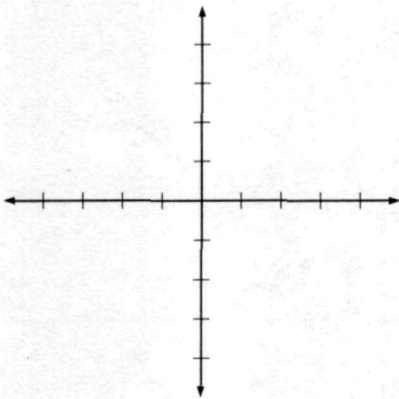
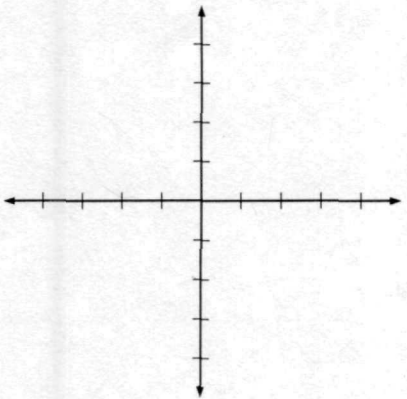
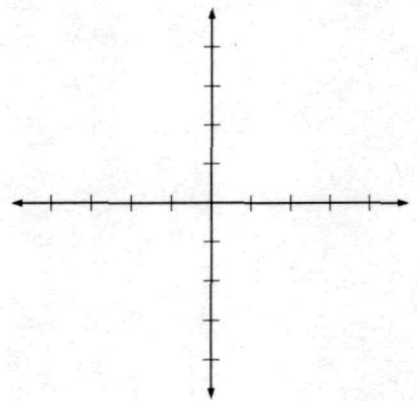
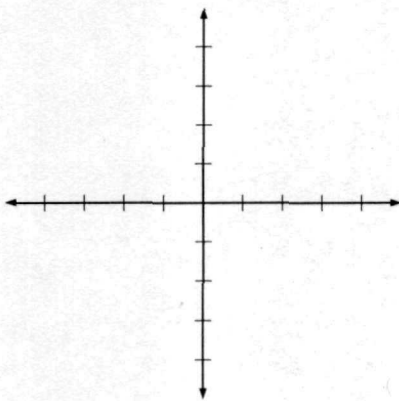
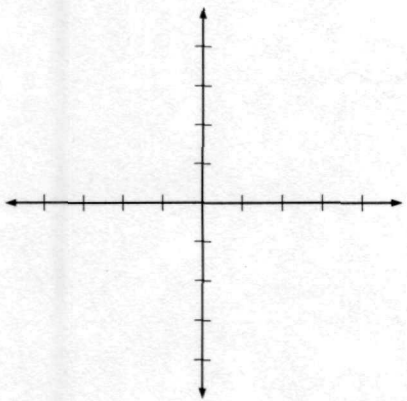
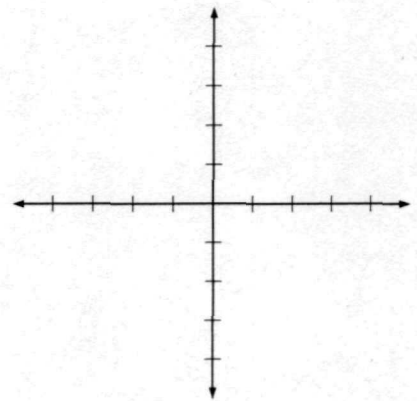
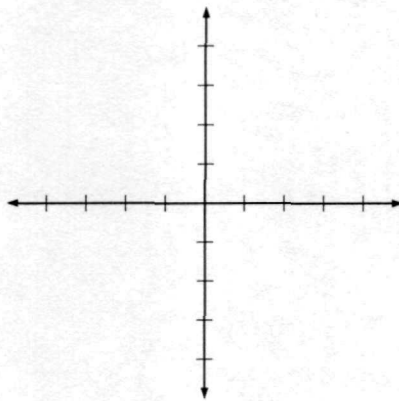
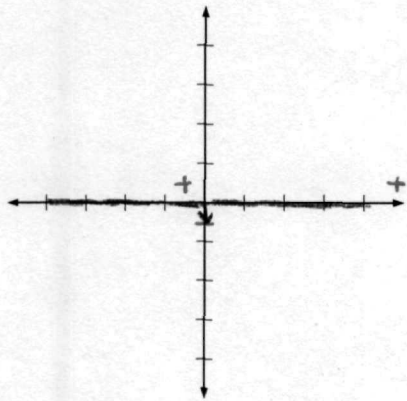


b)

6 mistakes starting with  $x^{(1)}$



1 mistake wrth  $x^{(2)}$



## 6.036: Machine Learning

Ryan Lacey <rlacey@mit.edu>

Collaborator(s): Jorge Perez, Amruth Venkatraman

February 14, 2014

6. (c) The range of possible separators is much smaller in part (b) than in part (a), which increases the likelihood of the perceptron algorithm requiring a high number of iterations. Starting with  $x^{(2)}$  we do not see this issue because the resulting decision boundary immediately and perfectly separates the data points. Starting with  $x^{(1)}$ , on the other hand, we see how the smaller changes in the slope of the decision boundary with  $x^{(3)}$  at  $[10, 1]$  has a large impact on the performance of the algorithm, taking three times as many iterations as with  $x^{(3)}$  at  $[1.5, 1]$ .
- (d) An adversary maximizes potential errors with the following strategy (maximizing  $\frac{R^2}{\gamma^2}$ ):
- Spread the points out over a large area, thereby increasing  $R$ .
  - Assign labels to points in a manner that minimizes the margin between differently labeled points, thereby reducing  $\gamma$ .
  - Traverse the points in order from nearest to the origin to furthest. Smaller magnitude vectors will require more mistakes to be made in order to converge to a decision boundary because of their small incremental progress.
7. (a) No, the training procedures do not necessarily converge to the same  $\theta$  and  $\theta_0$ . If training points are linearly separable, there are many choices for  $\theta$  and  $\theta_0$ .

As an example take the training points  $x^{(1)} = [2, 1]$ ,  $x^{(2)} = [-2, -1]$ , and  $x^{(3)} = [-1, 5]$  with corresponding labels  $y^{(1)} = (+1)$ ,  $y^{(2)} = (-1)$ , and  $y^{(3)} = (+1)$ .

We define two different decision boundaries with the following  $\theta$  and  $\theta_0$ :

$$\theta^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{ and } \theta_0^{(1)} = 0$$

$$\theta^{(2)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \text{ and } \theta_0^{(2)} = -1$$

These decision boundaries linearly separate the data so when the perceptron algorithm is applied to the training data with these parameters the boundaries incur no updates. Thus we have differing results from the perceptron algorithm on the same data.

- (b) The decision boundaries will not necessarily have the same performance on the test data. Let us assume we have some test data that is linearly separable along the  $x_1$  axis. In this case the first decision boundary from part (a) would perfectly classify the data, while the second boundary is apt to incorrectly predict several points.
- (c)  $\theta = \begin{bmatrix} -3 \\ 2 \end{bmatrix}$  and  $\theta_0 = -2$
8. (a) i. No such  $\theta$  could exist because the dot product with  $x$  would always be zero, thus making the second constraint unsatisfiable.  
ii. Yes, the addition of the offset parameter would allow us to satisfy both constraints.

- (b)
  - i. No such family member exists because three of the points are equidistant from the origin and only two are of the same label. Thus the classifier would have to predict these points to all be of the same value, which is incorrect.
  - ii. The points can be correctly classified with a circle of radius 2 centered at  $[-1, -1]$ .
  - iii. Looking at only the negatively labeled points we can conclude that no such family member exists. The points would either lie directly on the decision boundary (thus both being misclassified) or one would lie above and the other below the boundary (thus misclassifying one of the two points).
  - iv. A line with normal  $[1,1]$  and offset of  $-0.5$  would satisfy.
- (c) Families (iii) and (iv) are linear classifiers.

9. (a)  $A = \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} \end{bmatrix}$

- (b) Yes, given a good  $\theta_z$  one can find an equivalent  $\theta_x$ .

Classifier in  $z$ -space  $y^{(i)} (\theta_z z^{(i)}) > 0$  becomes  $y^{(i)} (\theta_z A x^{(i)}) > 0$ .

We desire the  $\theta_x$ , which must equal  $\theta_z A$  from the above equation, which gives the familiar  $y^{(i)} (\theta_x x^{(i)}) > 0$ .

- (c) No, there will not necessarily always be a  $\theta_z$  that will identically classify all the  $z$ 's. It will only hold when  $A$  is invertible.

Classifier  $y^{(i)} (\theta_x x^{(i)}) > 0$  becomes  $y^{(i)} (\theta_z A^{-1} z^{(i)}) > 0$  which requires  $A$  to be invertible.

- (d) Yes, the algorithm can converge more quickly in  $z$ -space. Take the situation in which one feature is enough to classify the data. Then the goal is only to converge over this feature rather than a large dimensional vector, which should require less iterations of the algorithm.
- (e) No, one cannot find a more accurate classifier in  $z$ -space on the training data because you have at best the same amount of information to train on as you would in  $x$ -space. You could have overtraining on the training data, however, in which case the classifier in  $z$ -space could outperform the classifier in  $x$ -space on test data.

10. (a) The training set might not be representative of the general population of data. If so, then the classifier would be a poor predictor of the correct labels that should be applied to the population data.

Only measuring performance by the training set also leads to a tendency of overfitting to the training data. Because of this, even with training data that is fairly representative of population data, the predictions may not generalize well.

- (b) The perceptron algorithm would be ran on  $n - 1$  points  $n$  number of times, each time leaving out one of the training points to use as the test. The goal of cross validation is to test how well the classifier generalizes and about how often it is expected to misclassify data from the test set. If the training data is drawn from and is representative of the population being studied, then cross validation should show errors about as often as test set validation. The advantage, however, is that cross validation requires less data to be collected since data points are reused from training for testing.
- (c) A large number of cross validation errors with a low number of training set errors suggests that the boundary decision resulting from the learning algorithm overfit to the training set. One possible solution would be to take the best decision boundary that resulted from the data points subsets of the cross validation runs.