MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.036—Introduction to Machine Learning
Spring Semester 2017

Assignment 1 & 2

Issued: Tuesday, February 14th
Due: Friday, February 24th, 9:00 AM

Classification and the Perceptron Algorithm - Assignment 1

1. **Perceptron Mistakes**

   Consider applying the perceptron algorithm (through the origin) based on a small training set containing three points: $x^{(1)} = [-1, -1]$, $x^{(2)} = [1, 0]$, and $x^{(3)} = [-1, 1.5]$ with labels $y^{(1)} = 1$, $y^{(2)} = -1$, and $y^{(3)} = 1$. Given that the algorithm starts with $\theta^{(0)} = 0$, the first point that the algorithm sees is always considered a mistake. The algorithm starts with some data point and then cycles through the data until it makes no further mistakes.

   (a) How many mistakes does the algorithm make until convergence if the algorithm starts with data point $x^{(1)}$? How many mistakes does the algorithm make if it starts with data point $x^{(2)}$? Draw the progression of the separating plane as the algorithm cycles.

   (b) Now assume that $x^{(3)} = [-1, 10]$. How many mistakes does the algorithm make until convergence if cycling starts with data point $x^{(1)}$? How many if it starts with data point $x^{(2)}$? Draw the progression of the separating plane as the algorithm cycles.

   (c) Explain why there is a difference between the number of mistakes made by the algorithm in part a) and part b).

   (d) Briefly describe an adversarial procedure for selecting the order and value of labeled data points so as to maximize the number of mistakes the perceptron algorithm makes before converging. Assume that the data is indeed linearly separable, by a hyperplane that passes through the origin. You don't need to give a formal proof of your algorithm.

2. **Perceptron Performance**

   In class we initialized the perceptron algorithm with $\theta = 0$. In this problem we will also explore other initialization choices.

   (a) The following table shows a data set and the number of times each point is misclassified during a run of the perceptron algorithm (with offset). θ is initialized to zero.

i	$x^{(i)}$	y	times misclassified
1	[-4, 2]	+1	1
2	[-2, 1]	+1	0
3	[-1, -1]	-1	2
4	[2, 2]	-1	1
5	[1, -2]	-1	0

   Write down the post-training θ.

(b) Provide one example of a different initialization of θ such that the perceptron algorithm with this initialization could not produce the exact same mistakes as in part a).

(c) Given a set of training examples that are linearly separable through the origin, show that the initialization of θ does not impact the perceptron algorithm's ability to eventually converge.

(d) Since the convergence of the perceptron algorithm doesn't depend on the initialization, the end performance on the training set must be the same. Are the resulting θ's the same regardless of the initialization? Does this necessarily imply that the performance on a test set is the same?

3. **Decision Boundaries**

(a) Consider the function defined over three binary variables: $f(x_1, x_2, x_3) = (\neg x_1 \wedge \neg x_2 \wedge \neg x_3)$. We aim to find a θ such that, for any $x = [x_1, x_2, x_3]$, where $x_i \in \{0, 1\}$:

$$\theta \cdot x + \theta_0 > 0 \text{ when } f(x_1, x_2, x_3) = 1, \text{ and}$$

$$\theta \cdot x + \theta_o < 0 \text{ when } f(x_1, x_2, x_3) = 0.$$

i. If $\theta_0 = 0$ (no offset), would it be possible to learn such a θ?

ii. Would it be possible to learn the pair θ and θ_o?

(b) You are given the following labeled data points:

- Positive examples: $[-1, 1]$ and $[1, -1]$,
- Negative examples: $[1, 1]$ and $[2, 2]$.

For each of the following parameterized families of classifiers, find the parameters of a family member that can correctly classify the above data, or explain (with a clear diagram and/or words) why no such family member exists.

i. Inside or outside of an origin-centered circle with radius r.

ii. Inside or outside of an $[x, y]$-centered circle with radius r.

iii. Strictly above or below a line through the origin with normal θ.

iv. Strictly above or below a line with normal θ and offset θ_0.

(c) Which of the above are families of linear classifiers?

4. **Feature Vectors**

Consider a sequence of n-dimensional data points, $x^{(1)}, x^{(2)}, ...$, and a sequence of m-dimensional feature vectors, $z^{(1)}, z^{(2)}, ...$, extracted from the x's by a linear transformation, $z^{(i)} = Ax^{(i)}$. If m is much smaller than n, you might expect that it would be easier to learn in the lower dimensional feature space than in the original data space.

(a) Suppose $n = 6$, $m = 2$, z_1 is the average of the elements of x, and z_2 is the average of the first three elements of x minus the average of fourth through sixth elements of x. Determine A.

(b) Suppose $h(z) = sign(\theta_z \cdot z)$ is a classifier for the feature vectors, and $h(x) = sign(\theta_x \cdot x)$ is a classifier for the original data vectors. Given a θ_z that produces good classifications of the feature vectors, is there a θ_x that will identically classify the associated x's.

(c) Given the same classifiers as in (b), if there is a θ_x that produces good classifications of the data vectors, will there *always* be a θ_z that will identically classify the associated z's? An explanation on intuition is sufficient.

(d) **Optional**: Under what conditions on A will such a θ_z always exist?

(e) If $m < n$, can we find a more accurate classifier by training in z-space, as measured on the training data? How about on unseen data?

Perceptron Convergence Rates - Assignment 2

5. **Mistake Bounds**

Consider a set of n labeled training data points $\{(x^{(t)}, y^{(t)}), t = 1, \ldots, n\}$, where each $y^{(t)} \in \{-1, +1\}$ is the label for the point $x^{(t)}$, a d-dimensional vector defined as follows:

$$x_i^{(t)} = \begin{cases} \cos(\pi t) & i = t \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Recall the no-offset perceptron algorithm, and assume that $\theta \cdot x = 0$ is treated as a mistake, regardless of label. Assume in this problem that $n = d$ and that we initialize $\theta = 0$ as usual.

(a) Consider the case $d = 2$. For each assignment of labels $y^{(1)}, y^{(2)}$, sketch the θ that results by running the perceptron algorithm until convergence. Explain briefly why the algorithm will make 2 updates for any ordering (and labeling) of the feature vectors.

(b) Now consider the general case. Show that the no-offset perceptron algorithm will make exactly d updates to θ, regardless of the order in which we present the feature vectors ($x^{(t)}$), and regardless of how these vectors are labeled, i.e., no matter how we choose $y^{(t)}$, $t = 1, \ldots, t$.

(c) What is the vector θ that the perceptron algorithm converges to based on this d-dimensional training set? Does θ depend on the ordering? How about the labeling?

(d) Is the number of perceptron algorithm mistakes made, d, a violation of the $\frac{R^2}{\gamma^2}$ bound on mistakes we proved in class? Why or why not (Hint: $\|x^{(t)}\| = 1$ for all t, what is $\|\theta\|$?)

6. **Linear Support Vector Machines**

The training objective for the Support Vector Machine (with slack) can be seen as optimizing a balance between the average hinge loss over the examples and a regularization term that tries to keep the parameters small (increase the margin). This balance is set by the regularization parameter λ. Here we only consider the case without the offset parameter θ_0 (setting it to zero) so that the training objective is given by

$$\left[\frac{1}{n}\sum_{i=1}^{n} Loss_h(y^{(i)}\,\theta \cdot x^{(i)}\,)\right] + \frac{\lambda}{2}||\theta||^2 = \frac{1}{n}\sum_{i=1}^{n}\left[Loss_h(y^{(i)}\,\theta \cdot x^{(i)}\,) + \frac{\lambda}{2}||\theta||^2\right] \tag{2}$$

where $\text{Loss}_h(y(\theta \cdot x)) = \max\{0, 1 - y(\theta \cdot x)\}$ is the hinge loss. Because the objective function can be expressed as a summation over the training points (right-hand side of equation), we can minimize this objective function, e.g., with the Pegasos algorithm that iteratively selects a training point at random and applies a gradient descent update rule based on the corresponding term inside the brackets on the right hand side.

Here we try to understand the optimization problem itself and how the regularization parameter λ affects the result. To this end, consider only a single training example (x, y). We try to find θ so as to minimize

$$Loss_h(y\,\theta \cdot x\,) + \frac{\lambda}{2}||\theta||^2 \tag{3}$$

(a) Show that the solution is necessarily of the form $\hat{\theta} = \eta\, yx$ for some real $\eta > 0$.

(b) If $Loss_h(y\,\hat{\theta} \cdot x\,) = 0$ holds for the solution, determine $\hat{\theta}$ explicitly as a function of (x, y).

(c) Now, let $\hat{\theta} = \hat{\theta}(\lambda)$ be the solution as a function of λ. Is it possible that for some value of λ the training example (x, y) remains miss-classified by $\hat{\theta}(\lambda)$?

(d) Is it possible that the resulting margin boundaries $\{x : \hat{\theta}(\lambda) \cdot x = 1\}$ and $\{x : \hat{\theta}(\lambda) \cdot x = -1\}$ extend past the single training point x (i.e. x lies in between the margin boundary lines) for some value of λ?