Introduction to Machine Learning (incomplete)

Aswin

Contents

1 Overview

I made this document as a way to learn ML for my Maters' thesis . Its not an original work,but a compilation of scribed notes of the ML course by Dr Sanjoy Dasgupta(UCSD), [SD] which I audited.Some parts are drawn directly/inspired from Andrew NG's Stanford CS229 course notes, [ANG] and Kilian weinberger's Cornell CS4780 course notes[KW].My primary reference was 'The Elements of Statistical Learning' by Trevor Hastie,Robert Tibshirani,Jerome Friedman, [HTF].So there will be considerable overlap with the aforementioned materials.This note might lack mathematical rigor but I have tried to give proofs and supplementary topics wherever necessary.For each algorithm, I have given a url to Github repo containing the implemetation using one of the three datasets viz Fisher's Iris dataset, Wisconsin Breast Cancer Dataset and MNIST handwritten digits dataset. Please write to me if you find any inaccuracies.I hope this proves at least moderately interesting or useful for you.

2 Supervised Learning

In a typical scenario, we have an outcome measurement, usually quantitative (such as a stock price) or categorical (such as heart attack/no heart attack), that we wish to predict based on a set of features (such as diet and clinical measurements). We have a training set of data, in which we observe the outcome and feature measurements for a set of objects (such as people). Using this data we build a prediction model, or learner, which will enable us to predict the outcome for new unseen objects. A good learner is one that accurately predicts such an outcome. The examples above describe what is called the supervised learning problem. It is called supervised because of the presence of the outcome variable to guide the learning process.[HTF]

2.1 Variable Types and Terminology

The outcome measurement which we wish to predict denoted as ***outputs*** depend on a set of variebles denoted as ***inputs***. Classicaly,the *inputs* are independent varibales whereas *outputs* are dependent variables. The term *features* will be used interchangeably with inputs.

The *outputs* which we wish to predict can be qualitative or quantitative(as in blood sugar level). When the *outputs* are qualitative (as in spams or not spams),it is referred as categorical or discrete variables and are typically represented numerically by codes,as in -spam or not spam can be coded as -1 or 1. Depending upon the kind of output varible, the prediction task can be of two types: *regression* when we predict quantitative outputs and *classification* when we predict qualitative outputs.

The input variables/features are denoted by $x^{(i)}$ and the space of all such $x^{(i)}$ is X. The output variable that we are trying to predict is denoted as $y^{(i)}$ and the space of all such $y^{(i)}$ is Y. A pair $(x^{(i)}, y^{(i)})$ is called a training example and the dataset, we will be using to learn - a collection of n training examples $\{(x^{(i)}, y^{(i)}); i = 1, 2, ..., n\}$ - is called a training set. The superscript (i) in the notation is simply an index into the training set.

3 Linear Regression

Given a vector of inputs, $x^T = (x_1, x_2, ..., x_d)$.We need to know functions/hypotheses h that can approximate y as a linear function of x:

$$h_\theta(x) = \theta_0 + \sum_{i=1}^{d} x_j \theta_j$$

θ_is are parameters/weight parametrizing the space of linear functions mapping from X to Y.The term θ_0 is the intercept, also known as the *bias* in machine learning.It is covenient to include θ_0 in the vector of weights θ and add constant variable 1 to the vector x, so that

$$h_\theta(x) = \theta^T x$$

w (weights) and b (bias) can be interchangably used with θ and θ_0 respectively.

For a training set,we have to learn the parameters θ, so that we can predict y. One reasonable method seems to be to make $h(x)$ close to y,for the training examples.To formalize this, we will define a function that measures, for each value of the s, how close the $h(x^{(i)})'$ s are to the corrsponding $y^{(i)}$'s.

We define Cost/Loss function:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)})^2,$$

this is the Least Squares cost function.In this approach, we pick the coefficients θ to minimize the cost function J.The LS cost function is quadratic function in weights, θ and hence it's minimum always exist, but may not be unique. Given a set of training examples $(x^{(i)}, y^{(i)})$ i.e training set ,define a matrix X to be the m-by-n matrix that contains the input values of training examples in it's rows:

$$\begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix}$$

Let $\mathbf{y}$ be the m-dimensional vector containing the target/output values

from the training set:

$$\begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

Since $h_\theta(x^{(i)}) = (x^{(i)})^T\theta$,we can verify that

$$X\theta - \mathbf{y} = \begin{bmatrix} (x^{(1)})^T\theta \\ \vdots \\ (x^{(m)})^T\theta \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$$= \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}$$

$\frac{1}{2}(X\theta - \mathbf{y})^T(X\theta - \mathbf{y}) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 = J(\theta)$ This is the cost function.To minimize J,we have to find the derivatives with respect to θ

Some matrix derivative results

$$\nabla_A trAB = B^T$$

$$\nabla_A |A| = |A|(A^{-1})^T$$

$$\nabla_{A^T} f(A) = (\nabla_A f(A))^T$$

$$\nabla_A trABA^TC = CAB + C^TAB^T$$

Using the last two results

$$\nabla_{A^T} trABA^TC = B^TA^TC^T + BA^TC$$

The cost function, $J(\theta) = \frac{1}{2}(X\theta - \mathbf{y})^T(X\theta - \mathbf{y})$

$$\nabla_\theta J(\theta) = \nabla_\theta \frac{1}{2}(X\theta - \mathbf{y})^T(X\theta - \mathbf{y})$$

$$= \frac{1}{2}\nabla_\theta(\theta^TX^TX\theta - \theta^TX^T\mathbf{y} - \mathbf{y}^TX\theta + \mathbf{y}^T\mathbf{y})$$

$$= \frac{1}{2}\nabla_\theta tr(\theta^TX^TX\theta - \theta^TX^T\mathbf{y} - \mathbf{y}^TX\theta + \mathbf{y}^T\mathbf{y})$$

$$= \frac{1}{2}\nabla_\theta(tr\theta^T X^T X\theta - 2tr\mathbf{y}^T X\theta)$$

$$= \frac{1}{2}(X^T X\theta + X^T X\theta - 2X^T\mathbf{y}) = X^T X\theta - X^T\mathbf{y}$$

To minimize J, we set the derivative to zero and obtain:$X^T(X\theta - \mathbf{y}) = 0$. If $X^T X$ is nonsingular then the value of θ that minimizes $J(\theta)$ is given in closed form by the equation,

$$\theta = (X^T X)^{-1} X^T\mathbf{y}$$

Now that we have the parameters,we can predict the output corresponding to the input $x^{(i)}$ as $y^{(i)} = \theta^T x^{(i)}$

4 The Perceptron

In a binary classification problem , where dataset(D) is $(x, y) \in \mathbb{R}^d \times \{-1, 1\}$, the learning problem is to find a hyperplane which separates the data into two classes, assuming the data is linearly classifiable. The hyperplane is parametrized by $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that $w.x + b = 0$, so the poblem is equivalent to learning the parameters w and b.On point x, we predict the label as **sign(w.x + b)**.

$$(w.x + b) > 0 \implies y = +1 \therefore y(w.x + b) > 0$$

$$(w.x + b) < 0 \implies y = -1 \therefore y(w.x + b) > 0$$

i.e., If the true label of x is y, then $y(w.x + b) > 0$, whereas for a misclassified point$y(w.x + b) \leq 0$. The Loss function for the perceptron can be defined as

$$Loss = \begin{cases} 0, & \text{if } y(w.x + b) > 0 \\ -y(w.x + b), & \text{if } y(w.x + b) \leq 0 \end{cases} \tag{1}$$

This loss function is a convex function of $y(w.x + b)$, and we can use stochastic gradient descent to find the value of parameters that minimizes the loss function. The update on the parameter w can be written as $w := w - \eta\nabla L(w)$,where $w = [w, b]$.The derivative of Loss function with respect to

the parameters(assuming bias term is not absorbed into the weight vector) are

$$\frac{\partial L}{\partial w} = -yx, \frac{\partial L}{\partial b} = -y$$

.

So the update rule will be $w := w + \eta yx$ and $b := b + \eta y$. For $w = [w, b]$, this is equivalent to $w := w + \eta yx$. If $\eta = 1$, then the update rule will be $w := w + yx$.

Perceptron Algorithm

```
Initialize w⃗ = 0
while TRUE do
    m = 0
    for (x_i, y_i) ∈ D do
        if y_i(w⃗^T . x_i) ≤ 0 then
            w⃗ ← w⃗ + yx
            m ← m + 1
        end if
    end for
    if m = 0 then
        break
    end if
end while
```

If the taining data is linearly classifiable, Perceptron is guaranteed to converge after finite number of steps and return a seperating hyperplane with zero training error.

Margin γ of a hyperplane w is defined as $\gamma = min_{(x_i,y_i)\in D}\frac{|x_i^T w|}{\|w\|_2}$, i.e it is the distance to the closest data point from the hyperplane paramtrized by w.

Theorem. *If a data set is linearly separable, the Perceptron will find a separating hyperplane in a finite number of updates.*

Proof. Suppose $\exists w^*$ such that $y_i(x^T w^*) > 0\ \forall (x_i, y_i) \in D$.Suppose that we rescale each data point and the w^* such that

$$\|w^*\| = 1 \ and \ \|x_i\| \leq 1 \forall x_i \in D$$

So the margin γ for the hyperplane w^* becomes $\gamma = min_{(x_i,y_i)\in D}|x_i^T w^*|$. After rescaling, all inputs x_i lies in a unit sphere in d-dimensional space.The separating hyperplane is defined by w^* with $\|w\|^* = 1$ i.e is w^* lies exactly on the unit sphere.

We claim that if the above assumptions hold,then the Perceptron algorithm makes atmost $\frac{1}{\gamma^2}$ mistakes.The update on w is only done in the instance of misclassification, i.e when $y(x^T w) \leq 0$ holds.As w^* is a seperating hyperplane and classifies all points correctly, $y(x^T w) > 0\ \forall x$.

Consider the effect of an update $w \leftarrow w + xy$ on the two terms $w^T w^*$ and $w^T w$.

$w^T w^* = (w + xy)^T w^* = w^T w^* + y(x^T w^*) \geq w^T w^* + \gamma$

The inequality follows from the fact that, for w^*, the distance from the hyperplane defined by w^* to x must be at least γ i.e $y(x^T w^*) = |x^T w^*| \geq \gamma$.This implies that with each update $w^T w^*$ grows atleast by γ

$w^T w = (w+xy)^T(w+xy) = w^T w + \underbrace{2y(w^T x)}_{\leq 0} + \underbrace{y^2(x^T x)}_{0 \leq \leq 1} \leq w^T w + 1$. This inequality follows the fact that, $2y(w^T x) \leq 0$, as we had to make an update, meaning x was misclassified. $0 \leq y^2(x^T x) \leq 1$ as $y^2 = 1$ always and all $x^T x \leq 1$ as $\|x\| \leq 1$,(rescaled). This implies that $w^T w$ grows at most by 1.

After M updates, the two inequalities becomes

$$w^T w^* \geq M\gamma$$

$$w^T w \leq M$$

$$M\gamma \leq w^T w^* \leq |w^T w^*| \leq \|w^T\| \underbrace{\|w^*\|}_{1} = \sqrt{w^T w}$$

$w^T w$ can most be M, as w is initialized with 0 and with each update $w^T w$ grows at most by 1.

$$\implies M\gamma \leq \sqrt{M}$$
$$\implies M^2\gamma^2 \leq M$$
$$\implies M \leq \frac{1}{\gamma^2}$$

Hence the number of updates M is bounded from above by a constant.

□

5 Boosting

A Weak Classifier is the one with accuracy marginally better than random guessing. For a binary weak classifier this means, $P(h(x) \neq y) = \frac{1}{2} - \epsilon$. An learning algorithm which consistently generate such weak classifier is called a **weak learner**.
Boosting is a machine learning approach where such weak learners are combined to get better prediction acuracy.

5.1 AdaBoost

AdaBoost Algorithm

1. Given $(x^{(i)}, y^{(i)}), ..., (x^{(N)}, y^{(N)})$, where $y^{(i)} \in \{-1, +1\}$
2. Initialize the observation weights $w_i = \frac{1}{N}, i = 1, 2, ..., N$.
3. For m =1 to M:
   (a) Fit a classifier $h_m(x)$ to the trainng data using weights w_i.
   (b) Compute

$$err_m = \frac{\sum_{i=1}^{N} w_i y^{(i)} h_m(x^{(i)})}{\sum_{i=1}^{N} w_i}$$

   (c) compute $\alpha_m = \log((1 - err_m)/err_m)$
   (d) set $w_i \leftarrow w_i.exp[-\alpha_m.y^{(i)} h_m(x^{(i)})],\ i = 1, 2, ..., N$
4. Output $H(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$

6 Nearest Neighbour Methods

Nearest-neighbour methods use those observations in the taining set $\mathcal{T}$ closest in input space to x to form $\hat{Y}$. The k-nearest neighbour fit for $\hat{Y}$ is defined as follows:

$$\hat{Y(x)} = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i,$$

where N_k is neighbourhood of x defined by k closest points x_i in the training sample.

Closeness is quantified by a metric, which for instance can be assumed as Euclidean distance.We find k observations with x_i closes to x in the input space of training set, and average their responses.

The notion of distance between two vectors is defined by a norm. A norm is a function from a vector space over the real or complex numbers to the nonnegative real numbers that satisfies certain properties pertaining to scalability and additivity, and takes the value zero if only the input vector is zero.

As mentioned above we will use the Euclidean norm for all practical purposes.The Euclidean norm is a specific norm on a vector space, that is strongly related with the Euclidean distance, and equals the square root of the inner product of a vector with itself.On an n-dimensional Euclidean space R^n, the intuitive notion of length of the vector $x = (x1, x2, ..., xn)$ is captured by the formula

$$||x||_2 = \sqrt{x_1^2 + \dots x_n^2}$$

This definition gives distance between two vectors as the euclidean norm of the component wise difference i.e if $x, y \in R^n$ then $\| (x - y)||_2 = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$

7 Support Vector Machines

The Perceptron algorithm assures to return a linear seperating hyperplane if one exists.Existence of one such hyperplane implies that there is infinite such hyperplanes, and the perceptron doesn't guarantee to return the optimal seperating hyperplane i.e, the one with maximum margin. Support vector machines(SVM) finds maximum margin hyperplane.