MACHINE LEARNING NOTES

ANDREW B MAURER

Abstract. The goal of this document is to present my own own understanding of machine learning models from a somewhat theoretical standpoint. I want to describe the models rigorously and mathematically, as well as point out some key difference between models which seem similar. For a more applied view, I will have a Github repo dedicated to Python examples of these algorithms, which may be my own implementation in `numpy` or simply using pre-existing libraries (e.g., `sklearn` or `pytorch`) on publicly available data-sets to get a feel for the techniques. At some point I may take one or more of these sections and make it into its own document, or make this document into a book. That's a problem for another day. :)

Contents

1. Data and Inference 1
2. Optimization 2
3. Linear Regression 3
4. Logistic Regression 3
5. Support Vector Machines 3
6. Random Forests 3
7. Principal Component Analysis 4
Appendix A. Linear Algebra 4

1. Data and Inference

1.1. **Data.** Generally speaking, data may be in one of two forms: numerical or categorical. Numerical data consist of scalars x or vectors $\vec{x} \in \mathbb{R}^p$. Categorical take on a discrete set of values in some set S.

The following are examples of numerical data:

(1)

While these are examples of categorical data:

(1)

There are ways to apply numerical techniques to categorical data. One such method is called *one-hot encoding*. In this case, the elements of the discrete set S are enumerated $s_1, \ldots, s_r$, and an observation of the event s_j is encoded by the unit vector $e_j \in \mathbb{R}^r$. This is a hugely inefficient process, and it is often adventageous to reduce the dimension of this numerical embedding of categorical data. Such techniques are covered in Section [??].

Date: March 12, 2019.

1.2. **Supervised Models.** The data presented in a *supervised learning problem* will consist of data vectors $\vec{x}$ The goal when modeling is to choose a smoothly parametrized set of functions F, whose input resembles your data and whose output corresponds with the associated observations.

2. Optimization

2.1. **Motivation.** In machine learning we have a data set $D \subset \mathbb{R}^n$ and a cost function $J = J(D; \Theta)$, which depends on a number of parameters $\Theta = (\theta_1, \ldots, \theta_n)$. As the data set is given, the goal is to find parameter Θ^* which minimizes cost given the data. Symbolically, optimization is the search for the following parameter:

$$\Theta^* = \arg\min_{\Theta} J(D; \Theta) \tag{2.1.1}$$

We should note that perhaps Θ consists of many parameters relative to the data set, meaning the prediction function may overfit the training data. Overtrained models are highly specialized to the training data, and as such may perform poorly on previously unseen data. The problem of finding a model which fits the training data while generalizing to test data is called the *bias-variance tradeoff*, and will be explored in Section [??].

2.2. **Gradient Descent.** We have seen that the goal of machine learning is to optimize the cost function within a certain space of functions, parametrized by $\Theta = (\theta_1, \ldots, \theta_n)$. Let $f(\Theta)$ be the function we wish to minimize. Recall this definition from calculus:

Definition 2.2.1. The *gradient* of $f(\Theta)$ at a point Θ_0 is the vector

$$\nabla f(\Theta_0) = \left\langle \frac{\partial f}{\partial \theta_1}(\Theta_0), \ldots, \frac{\partial f}{\partial \theta_n}(\Theta_0) \right\rangle \tag{2.2.1}$$

It is a basic fact of calculus that in a neighborhood of the point Θ_0, $f(\Theta)$ increases fastest in the direction of the gradient $\nabla f(\Theta_0)$ and decreases fastest in the direction of the negative gradient $-\nabla f(\Theta_0)$.

Definition 2.2.2. *Gradient descent* is an algorithm which minimizes a differentiable function $f(\Theta)$ by taking small steps in the direction of steepest decrease.

This direction of steepest decrease is identified by using the gradient. Because the gradient may be large in magnitude, we choose a number $\alpha > 0$ and move in the direction of the gradient, with magnitude scaled by α.

```
gradient_descent(func, dim, gradient, alpha):
    Theta = random_vector(dim)
    for 1,2,...,num_iterations:
        grad = gradient(func, Theta)
        Theta = Theta - alpha * grad
    return Theta
```

Figure 1. Pseudo-Code for Gradient Descent

2.3. **Adam Optimizer.**

3. Linear Regression

4. Logistic Regression

4.1. **Overview.** Logistic regression is a machine learning technique that can be used for both classification or regression. At its heart, logistic regression is the composition of an affine transformation $A : \mathbb{R}^n \to \mathbb{R}$ with the sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$. In general, this sort of function may be written as

$$f(x) = \frac{1}{1 + \exp(-w^T \cdot x + b)}. \tag{4.1.1}$$

Typically, if $f(x) \geq 0.5$, we predict $\hat{y} = 1$, and if $f(x) < 0.5$, we predict $\hat{y} = 0$. The *decision boundary* is thus $\{f(x) = 0.5\} = \{w^T \cdot x = b\}$. This is a hyperplane $H(w, b) \subseteq \mathbb{R}^n$. Points close to the decision boundary are more ambiguous, and points with higher orthogonal distance to the decision boundary yields a more extreme estimate.

5. Support Vector Machines

5.1. **Basic Definitions.** When a binary classification dataset $D \subseteq \mathbb{R}^d$ is *linearly separable*, it becomes advantageous to choose not just *a* separating hyperplane, but rather a separating hyperplane which maintains maximal distance from the data set.

Definition 5.1.1. A *hyperplane* in $\mathbb{R}^d$ is the zero-set of a linear equation. With $w \in \mathbb{R}^d$, $c \in \mathbb{R}$:

$$H(w, c) = \left\{x \in \mathbb{R}^d \mid w \cdot x = c\right\} \tag{5.1.1}$$

A dataset $D = D^+ \sqcup D^-$ is *linearly separable* if there exists a hyperplane $H(w, c)$ such that

$$D^+ = \{x \in D \mid w \cdot x > c\} \text{ and } D^- = \{x \in D \mid w \cdot x < c\}. \tag{5.1.2}$$

The hyperplane H is called a *separating hyperplane*.

Definition 5.1.2. A *support vector machine* is a machine learning algorithm used on linearly separable data to find a separating hyperplane $H(w, c)$ which *maximizes* the distance between points in the data-set and the plane $H(w, c)$.

6. Random Forests

6.1. **Overview.** Random forests are an *ensemble learning method* which employs *bagging*. The idea is to build many high-variance, low-bias decision trees and average (or vote on) the results to reduce variance. The individual trees are constructed so that each tree captures different predictive power, because ensemble techniques work best when

6.2. **Decision Trees.** Decision trees are a learning method which can be used on both categorical and regression data.

7. Principal Component Analysis

7.1. **Overview and Key Points.** Principal Component Analysis (PCA) is an unsupervised algorithm which performs dimensionality reduction on numerical variables. This is very similar to singular value decomposition of linear algebra.

Essentially we take an orthogonal transformation $F : \mathbb{R}^n \to \mathbb{R}^n$ such that

7.2. **Algorithm.**

Appendix A. Linear Algebra

Email address: andrew.b.maurer@gmail.com
URL: andrewmaurer.github.io