CS 189 Introduction to Machine Learning
Fall 2017 Note 1

1 Machine Learning Principles

Have you ever tailored your exam preparation, to optimize exam performance? It seems likely that you would, and in fact, those preparations may range from "taking one practice exam" to "wearing lucky underwear". The question we're implicitly answering is: which of these actually betters exam performance? This is where machine learning comes in. In machine learning, we strive to find patterns in data, then make predictions using those patterns, just as the average student would do to better exam scores. We start by structuring an approach to understanding machine learning.

1.1 Levels of Abstraction

We will take an example from astronomy, below. How exactly do we define the parameters of the problem, and how do we leverage the machine learning framework to help with this problem? See below, for a concrete outline of what to think about.

1. **Applications and Data**: What are you trying to do? What is your data like? Here, identify your problem and the nature of your observations. Your data may be cartesian coordinates, an matrix of RGB values etc.

   Example: We have (x_1, y_1) coordinates. We want to compute each planet's orbit.

2. **Model**: What kind of pattern do you want to find? This could be a polynomial, a geometric figure, a set of dynamics governing a self-feedback loop etc.

   Example: An ellipse for an orbit.

3. **Optimization Problem**: Whatever problem you have, turn it into an optimization problem. We could minimize losses or maximize gains, but in either case, define an objective function that formally specifies what you care about.

   Example: $\min_{\vec{x}} ||A\vec{x} - \vec{b}||_2^2$

4. **Optimization Algorithm**: How do we minimize? Determine how exactly to solve the optimization problem that you've now proposed.

   Example: Solve $C\vec{x} = \vec{d}$, a system of linear equations.

In this course, we will study both **models** and **optimization problems** extensively. Being able to compartmentalize each of these will make it easier for you to frame and understand machine learning.

2 Ordinary Least Squares (OLS) Review

Let us now use the framework specified above to discuss ordinary least squares, a canonical optimization problem that we'll study in-depth from various perspectives. We'll then slowly add complexity to least squares, with a number of different techniques. We're most familiar with the 2-dimensional case, commonly called "linear regression". So, let's start there.

1. **Applications and Data**: We have n data points (a_i, b_i) and wish to determine how the a_i's determine the b_i's.
2. **Model**: With least squares, we assume a_i, b_i are linearly related. More concretely, we wish to find a model m, c such that $b_i \approx ma_i + c$ where $b_i, a_i, m, c \in \mathbb{R}$. In other words, we want to learn how the data determines m, c.
3. **Optimization Problem**: To solve this problem, we turn it into an optimization problem:
$$\min_{\vec{x}} \|(m\vec{a} + c\mathbb{1}) - \vec{b}\|_2^2$$
4. **Optimization Algorithm**: As it turns out, ordinary least squares has a closed-form solution for the optimal value of $\vec{x}$. We can solve for this a number of ways, and below, we'll see perspectives that each of these methods belies.

What if our samples are vectors, instead? Our model is then a matrix X, and we'll need to be more careful about our formulation of the least squares framework.

1. **Applications and Data**: We have n data points $(\vec{a}_i, \vec{b}_i)$ and wish to determine how the $\vec{a}_i$'s determine the $\vec{b}_i$'s.
2. **Model**: Assume $\vec{a}_i, \vec{b}_i$ are linearly related. Find X such that $\vec{b}_i \approx X\vec{a}_i$ where $\vec{b}_i \in \mathbb{R}^m, \vec{a}_i \in \mathbb{R}^l, X \in \mathbb{R}^{m \times l}$.
3. **Optimization Problem**: To solve this problem, we turn it into an optimization problem. We rewrite in the least squares framework by flattening X (i.e., stacking each column vector of X on top of each other) and using the $\vec{a}_i$'s to form diagonal block matrices as follows, to give us:

$$\min_{\vec{x}} \left\| \begin{bmatrix} \vec{a}_1^T & 0 & 0 & 0 \\ 0 & \vec{a}_1^T & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \vec{a}_1^T \\ \vec{a}_2^T & 0 & 0 & 0 \\ 0 & \vec{a}_2^T & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \vec{a}_2^T \\ \vec{a}_3^T & 0 & 0 & 0 \\ 0 & \vec{a}_3^T & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \vec{a}_3^T \\ \vdots & \vdots & \vdots & \vdots \\ \vec{a}_n^T & 0 & 0 & 0 \\ 0 & \vec{a}_n^T & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \vec{a}_n^T \end{bmatrix} \begin{bmatrix} -- \\ -- \\ -- \\ \vdots \\ -- \end{bmatrix} - \begin{bmatrix} \vec{b}_1 \\ \vec{b}_2 \\ \vdots \\ \vec{b}_n \end{bmatrix} \right\|^2$$

Using the notation $M_{m \times n}$ to denote a matrix with m rows and n columns (an m by n matrix), we see that the matrices and vectors above have the following dimensions:

$$A_{mn \times ml} \vec{x}_{ml \times 1} = \vec{b}_{mn \times 1}$$

4. **Optimization Algorithm**: Again, we cover this in the next section.

2.1 Vector Calculus

We consider the brute-force approach to deriving the solution, first. Just as you would find optimum for scalar-valued functions, take the derivative, set it equal to 0, and solve. Before we begin, keep in mind the following convention. See Note 0 for a primer on vector calculus.

$$\frac{\partial w^T x}{\partial x} = w, \frac{\partial x^T A x}{\partial x} = (A + A^T)x$$

Using these conventions, we can now take the gradient of our objective function. Apply the definition of the L2-norm to start, that $||x||_2^2 = x^T x$.

$$\begin{aligned}
&\frac{\partial}{\partial \vec{x}}\|A\vec{x}-\vec{b}\|_2^2 \\
&= \frac{\partial}{\partial \vec{x}}((A\vec{x}-\vec{b})^T(A\vec{x}-\vec{b})) \\
&= \frac{\partial}{\partial \vec{x}}(\vec{x}^T A^T A\vec{x} - (A\vec{x})^T b - b^T(A\vec{x}) + \vec{b}^T\vec{b}) \\
&= \frac{\partial}{\partial \vec{x}}(\vec{x}^T A^T A\vec{x} - 2\vec{x}^T A^T b + \vec{b}^T\vec{b}) && \text{recall } b^T a = a^T b \text{ for } a,b \in \mathbb{R}^d \\
&= 2A^T A\vec{x} - 2A^T b
\end{aligned}$$

Now, set the gradient to 0, and solve for x^*.

$$\begin{aligned}
2A^T A\vec{x} &= 2A^T b \\
A^T A\vec{x} &= A^T b \\
\vec{x}^* &= (A^T A)^{-1} A^T b
\end{aligned}$$

Finally, we have our closed form solution.

2.2 Projection

We want to find something, denoted $\hat{\vec{x}}$, spanned by the columns of A, that is closest to $\vec{b}$. As it turns out, this is precisely $\vec{b}$ projected onto the column space of A. Then, the error $\vec{b} - A\vec{x}$ is perpendicular to everything spanned by the columns of A. This gives us the following formulation:

$$\begin{aligned}
\vec{a}_i^T(\vec{b} - A\vec{x}) &= 0, \ \forall \vec{x} \\
A^T(\vec{b} - A\vec{x}) &= \vec{0} \\
A^T\vec{b} &= A^T A\vec{x} \\
\vec{x}^* &= (A^T A)^{-1} A^T \vec{b}
\end{aligned}$$