Lectures on Data Science and Engineering
Lecture 1

Constantine Caramanis, Alex Dimakis
The University of Texas at Austin
Department of Electrical and Computer Engineering

Introduction

In this lecture we are going to understand the problem of **supervised binary classification**. We will work with an example and define **empirical risk** (training error) and **true risk** (generalization error). This will be key in understanding what **overfitting** is, a problem that is central for machine learning.

Suppose you are hired at a startup company making a new kind of nano-chip. Unfortunately, some of the nano-chips they manufacture explode after a month of use. This is a problem, since they get a lot of angry customer calls. They hire you as a *data scientist*[1] to solve their problem.

Their nano-manufacturing process introduces variability in the sizes of the chips and they believe this has something to do with the exploding problem. The first step is to collect some data: they manufacture five chips and let them run in a lab for one month. After that period they give you a dataset.

	height	**width**	**y=exploded?**
chip 1	0.8	0.8	1
chip 2	0.3	0.25	0
chip 3	0.2	0.8	0
chip 4	0.3	0.7	0
chip 5	0.9	0.7	1

Table 1: Your dataset. There is a special column (called y) that we are trying to predict using the other columns called features. Every row corresponds to one labeled nano-chip. The number of examples (a.k.a. Samples) is usually denoted by n and the number of features by p. In this example $n = 4$ and $p = 2$.

In this lecture we discuss the most common problem in statistics and machine learning, that of **prediction**. It is important to understand that our dataset was collected and brought in this form with pain and cost. For each row with a label you observe somebody had to wait one month for a microchip to explode. Usually, most of the actual work of a data

[1] Your job title would have been different a few years back, but you can get a better salary if you just stick to data scientist.
Here we quote the famous definition tweet by Josh Wills: Data Scientist (n.): Person who is better at statistics than any software engineer and better at software engineering than any statistician.

scientist is in collecting, cleaning and joining data to get in such a form of a matrix X and a vector y.

Our problem is to perform **prediction**. We will try to learn a function $h(\mathbf{x})$ that takes as input the height and the width of a microchip and tries to predict if it will explode or not. Prediction refers to the future: I present to you the features of an untested nanochip and you have to predict if it will explode or not in one month. This is different from performing well on the training dataset, which you can do simply by **memorization**. As you know from school, it is possible to memorize but not actually understand anything and hence have zero ability to make new predictions. This is the issue of overfitting that we will be understanding in the next two lectures.

Lets recap the frequently used jargon:

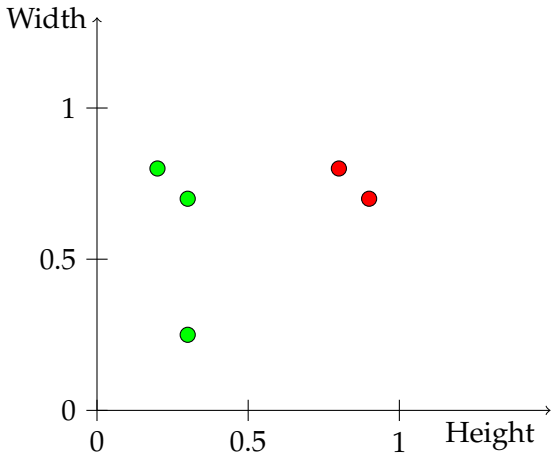
Jargon Box

The variable you are trying to predict y is called a **Label** or **Dependent variable**. Your observations (here: height and width) are called **features**, **dependent variables**,**predictors** or **covariates** in different bodies of literature.

When labels are given, it is called a **supervised** learning problem. When the labels are binary it is called **binary classification**. If we were trying to predict a continuous quantity (say y was temperature of the nano-chip) it is called **regression**.

The **Feature space** is the space where our data features live. Here it is $\mathbb{R}^2$ i.e., pairs of weight/height.

Therefore, our problem is a supervised learning problem and specifically a binary classiciation problem. Lets say we plot the data using colors for the two different labels:



It looks like the good and the bad nanochips are quite well separated on the feature space, anyone would be tempted to come up with a simple model, i.e., a simple test on the

features to predict if a new nanochip will be good or bad. Before we do that, lets try to put our problem in a precise mathematical framework, to understand what we are trying to do.

1 Statistical Learning Framework

Our goal is to define what constitutes a **prediction** and to precisely define and understand **overfitting**. We will learn the concepts of **True Risk** (a.k.a. **Generalization error or Test error**) versus **Empirical Risk** (a.k.a. **Training error**). This will lead to the main algorithm used in learning: **Empirical Risk Minimization (ERM)**.

We need *a mathematical model* of how data is generated and labeled.

- We assume we are given a distribution D over the feature space. Each sample $\mathbf{x}_i$ (weight and height of a nanochip) is assumed to be randomly and independently sampled from this distribution. We use bold for $\mathbf{x}$ because it is a vector of p numbers (the features), i.e., $\mathbf{x} \in \mathbb{R}^p$.
- We assume a true labeling function h_T. The universe samples a point $\mathbf{x}_i$ and computes the true label y_i (good or faulty nano-chip) by computing $y_i = h_T(\mathbf{x}_i)$.
- We need to choose how we count errors. This is called a **Loss function** $\ell(\hat{y}, y)$. This function takes a predicted label $\hat{y}$, and a true label y and tells you how much it costs to make the prediction $\hat{y}$ if the true label is y.

Our goal is to find the best model h. We define the **True risk** (a.k.a. **Generalization Error**) of a model h, denoted by $L_D(h)$ as follows:

$$L_D(h) = \mathrm{E}_{\mathbf{x}\sim D}[\ell(h(\mathbf{x}), h_T(\mathbf{x})].$$

The definition of true risk needs some attention: $\mathbf{x} \sim D$ means that it is now a random vector in $\mathbb{R}^p$, sampled using the distribution D. Therefore, the loss function is now a scalar random variable and we take its expectation. We will work through an example in a second.

In reality we would be only given a dataset a matrix $X \in \mathbb{R}^{n\times p}$ and labels $\mathbf{y} \in \mathbb{R}^n$. If we only have a dataset we cannot compute the true risk. Instead we try to approximate it by computing what is called **Empirical Risk** (a.k.a. Training error):

$$L_{\mathcal{S}}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(\mathbf{x}_i), y_i)).$$

Here we are summing over our dataset, example by example. For each i we have the feature vector $\mathbf{x}_i$ and training label y_i so we can compute if the model h is correct or not. We are simply averaging the performance of the model over the training set.

We are trying to *learn* the best model, which means find which h has the smallest empirical risk. This is called **Empirical Risk Minimization (ERM)** (a.k.a. Training).

2 Example: Computing Empirical Risk

Lets choose a simple model and compute its empirical risk. The model h_1 works as follows: It takes as input the features $\mathbf{x} = (\text{width}, \text{height})^T$ and predicts $y = 1$ if $width > 0.5$, and $y = 0$ otherwise. This is a simple decision tree of depth 1, and these short decision trees are called decision stumps. These are some of the simplest models we can define: Simply compare one feature with a threshold and assign a label. Lets repeat our dataset along with the predictions that it makes called $\hat{y}_1$. **Verify the prediction column yourself**:

	height	**width**	**y=exploded?**	**Model 1 prediction $\hat{y}_1$**
chip 1	0.8	0.8	1	1
chip 2	0.3	0.25	0	0
chip 3	0.2	0.8	0	1
chip 4	0.3	0.7	0	1
chip 5	0.9	0.7	1	1

Table 2: Your dataset $\mathcal{S}$. There is a special column (called y) that we are trying to predict using the other columns called features. Every row corresponds to one labeled nano-chip. The number of examples (a.k.a. Samples) is usually denoted by n and the number of features by p. In this example $n = 4$ and $p = 2$. The last column is the prediction of the binary decision stump h_1.

How well did model h_1 do? We want to compare y with the predictions $\hat{y}$, but that depends on the loss function we use. Lets simply use the $0-1$ loss now (denoted by ℓ_{01}) which simply charges 1 when the model is wrong and zero otherwise[2].

We can now compute the empirical risk of the model h_1 on this dataset $\mathcal{S}$:

$$L_{\mathcal{S}}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell_{01}(h(\mathbf{x}_i), y_i)) = \tag{1}$$

$$\frac{1}{5}(0+0+1+1+0) = \frac{1}{5}2 = \frac{2}{5}. \tag{2}$$

Exercise 1

Here is another model h_2: $h_2([w,h]) = 1 \text{ if } w \geq 0.75$, zero otherwise. In words, the model is another binary decision stump with threshold 0.75 for the feature *weight*.

- Compute the empirical risk of this model on our dataset.
- Is this a better or worse threshold for the stump?

[2]Note that in a real application, shipping a faulty nano-chip may be much more expensive compared to discarding a good one, so more sophisticated loss functions can be used, depending on the application

2.1 Example: Computing True Risk

Before, we computed the empirical risk of the model h_1, which is the training error on our tiny dataset.

$$L_{\mathcal{S}}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(h_1(\mathbf{x}_i), y_i)).$$

Now we would like to compute the true risk of the model h_1:

$$L_D(h) = \mathrm{E}_{\mathbf{x}\sim D}[\ell(h(\mathbf{x}), h_T(\mathbf{x})].$$

We cannot compute this expectation using only our dataset. We need to know the true distribution D that generates the data and also the true labeling function h_T. Note that if we knew the true labeling function h_T we would just use that to make perfect predictions so there would be no training needed. Still, we are going to work out how to compute the generalization error if D and h_T were known, to understand them better.

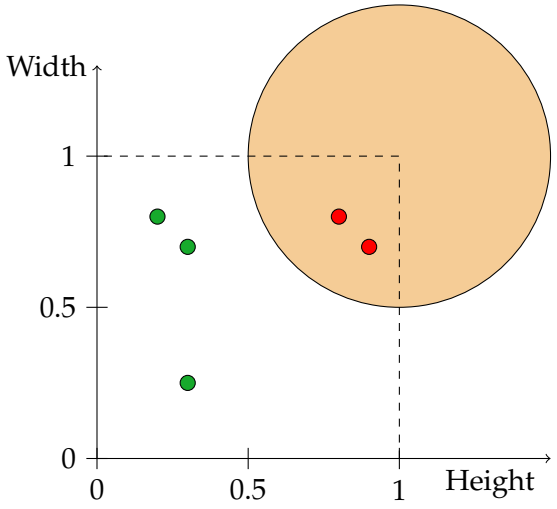
In this example lets assume:

• $D \sim$ Uniform$[0,1]x[0,1]$. In words, the weight and the height of the nano-chips are selected randomly uniformly and independently in $[0,1]$.

• Lets also assume a true labeling function:

$$h_T(w,h) = \begin{cases} 1 & \text{if } (w-1)^2 + (h-1)^2 \leq \frac{1}{4}, \\ 0 & \text{otherwise.} \end{cases}$$

This function will label nanochips as $h_T = 1$ (*exploding*) if their weight,height combination is within distance $1/2$ from the point $[1,1]$.



What is the true risk of the decision stump model h_1? Recall how h_1 labels points:

$$h_1(w,h) = \begin{cases} 1 & \text{if } w \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$$

Our goal is to compute the true risk of the model h_1:

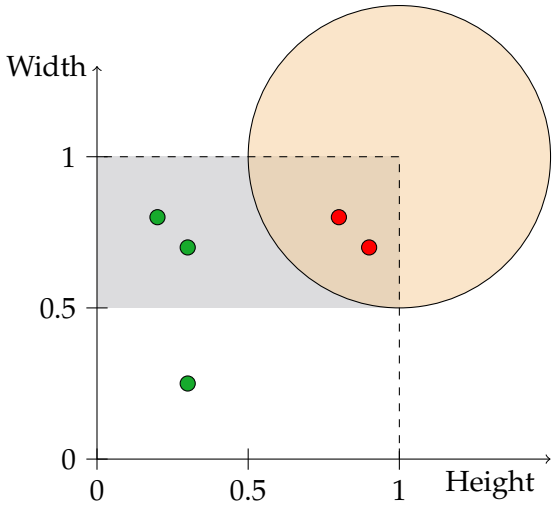
$$L_D(h) = \mathrm{E}_{\mathbf{x}\sim D}[\ell_{01}(h_1(\mathbf{x}), h_T(\mathbf{x})].$$

We are using the zero-one loss ℓ_{01} which takes as input a prediction $\hat{y}$ and a true value y and charges 1 when the prediction is wrong and zero otherwise:

$$\ell_{01}(\hat{y}, y) = \begin{cases} 0 & \text{if } \hat{y} = y, \\ 1 & \text{otherwise.} \end{cases}$$

Note that when the point $\mathbf{x}$ is chosen randomly from D, ℓ_{01} is a random variable that takes the values zero or one. The expectation of a zero-one random variable is exactly the probability that it is one. So to compute the true risk of the decision stump h_1 we simply need to find the probability that it makes a mistake if the data comes from D. The probability $\mathbb{P}[\ell_{01} = 1]$ is the probability that the random nanochip features land in the area of the feature space where h_1 is wrong, i.e., disagrees with h_T.

Lets draw the decision region where h_1 labels nanochips as exploding:



The light grey area is where h_1 labels exploding and the orange circle are true exploding, i.e., h_T labels exploding. The stump region h_1 is misclassifying the two green points on the upper left. Since the nanochip features are selected unformly in the $[0, 1]$ square, the probability that the loss is 1 is equal to the area of the grey rectangle minus the area of the quarter disk:

$$\mathbb{P}[\ell_{01} = 1] = 0.5 \times 1 - \frac{1}{4}\pi(\frac{1}{2})^2 = 0.30\ldots.$$

This exactly the true risk of the model h_1:

$$L_D(h) = \mathrm{E}_{\mathbf{x}\sim D}[\ell_{01}(h, \mathbf{x})] = 1 \times \mathbb{P}[\ell_{01} = 1] + 0 \times \mathbb{P}[\ell_{01} = 0] = 0.30\ldots.$$