Introduction

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, of course, since they get a lot of angry customer calls. They hire you as a data scientist 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.

<table>
<thead>
<tr>
<th></th>
<th>height</th>
<th>width</th>
<th>y=exploded?</th>
</tr>
</thead>
<tbody>
<tr>
<td>chip 1</td>
<td>0.8</td>
<td>0.8</td>
<td>1</td>
</tr>
<tr>
<td>chip 2</td>
<td>0.3</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>chip 3</td>
<td>0.2</td>
<td>0.8</td>
<td>0</td>
</tr>
<tr>
<td>chip 4</td>
<td>0.3</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>chip 5</td>
<td>0.9</td>
<td>0.7</td>
<td>1</td>
</tr>
</tbody>
</table>

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 (aka 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. Lets recap the frequently used jargon:

<table>
<thead>
<tr>
<th>Jargon Box</th>
</tr>
</thead>
<tbody>
<tr>
<td>The variable you are trying to predict ( y ) is called a <strong>Label</strong> or <strong>Dependent variable</strong>. Your observations (here: height and width) are called <strong>features</strong>, <strong>dependent variables</strong>, <strong>predictors</strong> or <strong>covariates</strong> in different bodies of literature. When labels are given, it is called a <strong>supervised</strong> learning problem. When the labels are binary it is called <strong>binary classification</strong>. If we were trying to predict a continuous quantity (say ( y ) was temperature of the nano-chip) it is called <strong>regression</strong>. The <strong>Feature space</strong> is the space where our data features live. Here it is ( \mathbb{R}^2 ) i.e. pairs of weight/height.</td>
</tr>
</tbody>
</table>
Therefore, our problem is a supervised learning problem and specifically a binary classification problem. Let's say we plot the data using colors for the two different labels:

![Graph](image)

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, let's 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 (aka Generalization error) versus Empirical Risk (aka Test 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 $x_i$ (weight and height of a nanochip) is assumed to be randomly and independently sampled from this distribution. We use bold for $x$ because it is a vector of $p$ numbers (the features), i.e. $x \in \mathbb{R}^p$.

- We assume a true labeling function $h_T$. The universe samples a point $x_i$ and computes the true label $y_i$ (good or faulty nano-chip) by computing $y_i = h_T(x_i)$.

- We need to choose how we count errors. This is called a Loss function $\ell(h, x)$. This function takes a model $h$, a data point $x$ and penalizes the model when it makes mistakes.

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

$$L_D(h) = E_{x \sim D}[\ell(h, x)].$$
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 $\mathbf{X} \in \mathbb{R}^{n \times p}$ 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 (aka Training error):

$$L_S(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i).$$

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) (aka Training).

### 2 Example: Computing Empirical Risk

Let’s choose a simple model and compute its empirical risk. The model $h_1$ works as follows: it takes as input the features $\mathbf{x} = (\text{width, height})^T$ and predicts $y = 1$ if $\text{width} > 0.5$, 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. Let’s repeat our dataset along with the predictions that it makes called $\hat{y}_1$.

<table>
<thead>
<tr>
<th></th>
<th>height</th>
<th>width</th>
<th>$y=$exploded?</th>
<th>Model 1 prediction $\hat{y}_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>chip 1</td>
<td>0.8</td>
<td>0.8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>chip 2</td>
<td>0.3</td>
<td>0.25</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>chip 3</td>
<td>0.2</td>
<td>0.8</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>chip 4</td>
<td>0.3</td>
<td>0.7</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>chip 5</td>
<td>0.9</td>
<td>0.7</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

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 (aka 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. Let’s simply use the $0-1$ loss now (denoted by $\ell_{01}$) which simply charges 1 when the model is wrong and zero otherwise.\(^1\)

\(^1\)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.
We can now compute the empirical risk of the model $h_1$ on this dataset $S$:

$$L_S(h) = \frac{1}{n} \sum_{i=1}^{n} \ell_{01}(h(x_i), y_i)] = \frac{1}{5}(0 + 0 + 1 + 1 + 0) = \frac{1}{5}3 = \frac{3}{5}.$$ 

**Exercise**

Here is another model $h_2$: $h_2([w, h]) = 1$ 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?

### 2.1 Example: Computing True Risk

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

$$L_S(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h_1(x_i), y_i).$$

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

$$L_D(h) = E_{x \sim D}[\ell(h, x)].$$

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 $h_T$. In this example lets assume:

- $D \sim \text{Uniform}[0, 1]x[0, 1]$. 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:

$$L_D(h) = E_{x \sim D}[\ell_{01}(h, x)].$$

We are using the zero-one loss $\ell_{01}$ 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 $x$ is chosen randomly from $D$, $\ell_{01}$ is a random variable that takes the values zero or one.

The probability $P(\ell_{01} = 1)$ is just 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 uniformly 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:

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

This exactly the true risk of the model $h_1$:

$$L_D(h) = E_{x \sim D}[\ell_{01}(h,x)] = 1 \times P[\ell_{01} = 1] + 0 \times P[\ell_{01} = 0] = 0.30 \ldots$$