Lecture 2

Lecturer: Alex Dimakis
Jan 26 2017,

Introduction

In the previous lecture we learned what is the Training error (Empirical Risk) and the Generalization Error (True Risk) and how to compute it for a model h.

Recall: We are given a dataset $\mathcal{S}$ of n labeled examples and the Empirical Risk is

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

which is averaging the loss over our training set. We hope that this will be a good approximation to the **True risk** of a model h, denoted by $L_D(h)$ as follows:

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

For any given model we can compute its training error. The challenge of training is to find the *best possible model*. In other words, we would ideally like to search over all possible models (*i.e.* search over all possible python functions that take the features $\mathbf{x}$ and produce a label y) to choose the one with the smallest empirical risk on the training set $\mathcal{S}$. This is called Empirical Risk Minimization (ERM):

$$\min_h L_S(h) = \min_h \frac{1}{n} \sum_{i=1}^{n} \ell(h_1(\mathbf{x}_i), y_i)].$$

where we are searching over all models to find the one that minimizes the loss.

Let's remember our dataset $\mathcal{S}$:

	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 (aka Samples) is usually denoted by n and the number of features by p. In this example $n = 4$ and $p = 2$.

Lets use 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:

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

Exercise

How small can you make the training error for this dataset $\mathcal{S}$ for the zero-one loss ℓ_{01}? You can use any model h you want.

Think about the previous exercise before continuing.

The problem is that we can always make the training error zero. One way to do this is by a model h that memorizes the dataset $\mathcal{S}$ and produces labels as follows:

Stupid Memorization Model h_m

- For a given input $\mathbf{x}$, if the same feature vector $\mathbf{x}$is in the training set, output the training label as a prediction: $h_m(\mathbf{x}) = y$.
- For a given input $\mathbf{x}$that is not in the training set, make the prediction $h_m(\mathbf{x}) = 0$

This model h_m achieves zero empirical loss but is a terrible model that will always predict 0 unless it has seen the example before. Using the framework of the previous lecture you can compute the true risk of h_m (for the D and true labeling function h_T given in Lect.1) and you will find that it is 1, i.e. the worst possible risk. This model has simply memorized the training set but has no predictive power: This is an example of **overfitting**.

1 How to avoid overfitting: Inductive Bias

The way we usually avoid overfitting is through hope: the hope that the universe is simple. Instead of minimizing the empirical risk over *all possible models* we limit our search within *simple* models. We postulate that the true labeling function is also simple and hence our search over simple models will find it, or find a model close to it[1].

[1] This is all formalized in the field of learning theory, where complexity is measured by the concept of VC dimension and its extensions.

1.1 Example: ERM over Stumps

In this example we will search over all decision stumps that look only at the variable *width*: Lets consider decision stumps h_θ that lebel points as follows:

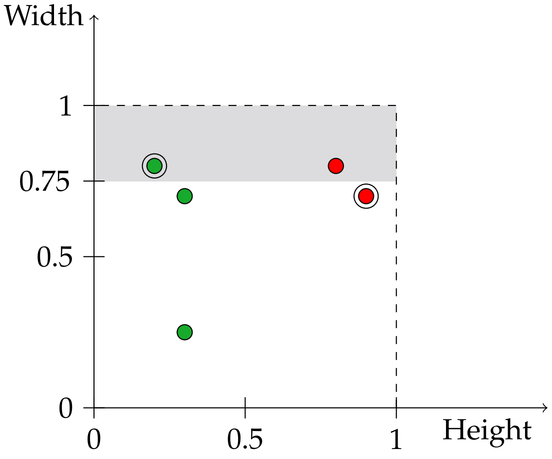
$$h_\theta(w, h) = \begin{cases} 1 & \text{if } w \ \geq \theta, \\ 0 & \text{otherwise.} \end{cases}$$

This is now a family of models $\mathcal{H}$. This is called a *hypothesis class* and this particular one is quite simple and is parametrized by one scalar parameter θ, the threshold we use.

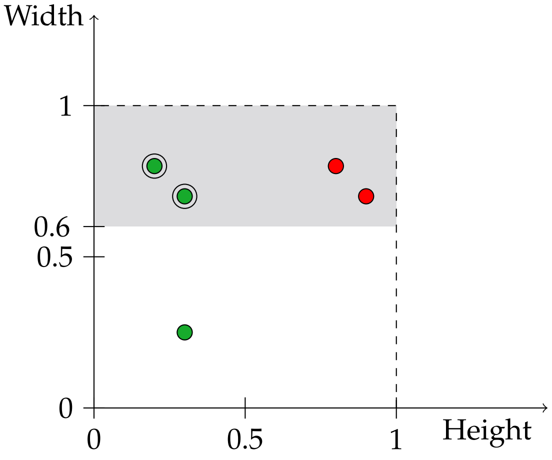
We will now perform ERM over this hypothesis class:

$$\min_{h \in \mathcal{H}} L_S(h_\theta) = \min_{\theta \in [0,1]} L_S(h_\theta).$$

Lets draw the decision region for h_θ when $\theta = 0.75$:



For $\theta = 0.75$ the model is misclassifying two points shown circled. So the emprical risk is $L_S(h_{0.75}) = \frac{1}{5}2$.
If we choose $\theta = 0.6$ we have the decision region:



For $\theta = 0.6$ the model is misclassifying again two points shown circled. So the emprical risk is the same: $L_S(h_{0.75}) = \frac{1}{5}2$.

You can see that for this dataset, there is no decision stump on the feature *width* that will misclassify fewer than 2 points. So $\theta* = 0.6$ or $\theta* = 0.7$ can be selected as an ERM optimum. If instead one uses a decision stump on the variable height, thresholding height on 0.5 will produce zero training error.

Exercise

- Think of an algorithm for training binary decision stump models.
- What is the running time in terms of the number of samples n and number of features p?

Exercise

Assume a data generation model as in Lecture 1:

- $D \sim \text{Uniform}[0,1]x[0,1]$. In words, the weight and the height of the nano-chips are selected randomly uniformly and independently in $[0,1]$. Assume the true labeling function to be:

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

This function will label nanochips as $h_T = 1$ (*exploding*) if their weight,height combination is within distance $1/2$ from the point $[1,1]$. We are using 0-1 loss throughout.

- Perform true risk minimization to find $\theta*$ for stumps on *width*.
- Perform true risk minimization over either *width* or *height*. What is the lowest possible true risk?