ISML-2: Machine Learning **Spring 2014**

Lecture 2- Statistical Learning Theory

Lecturer: L. Rosasco *Scribe: Lorenzo Rosasco*

Machine Learning deals with systems that are trained from data rather than being explicitly programmed. Here we describe the data model considered in statistical learning theory.

2.1 Data

The goal of supervised learning is to find an underlying input-output relation

$$f(x_{\text{new}}) \sim y,$$

given data.
The data, called *training set*, is a set of n input-output pairs,

$$S = \{(x_1, y_1), \dots, (x_n, y_n)\}.$$

Each pair is called an example. We consider the approach to machine learning based on the so called *learning from examples* paradigm.

Given the training set, the goal is to *learn* a corresponding input-output relation. To make sense of this task we have to postulate the existence of a model for the data. The model should take into account the possible *uncertainty* in the task and in the data.

2.2 Probabilistic Data Model

The inputs belong to an input space X, we assume throughout that $X \subseteq \mathbb{R}^D$. The outputs belong to an output space Y. We consider several possible situations: regression $Y \subseteq \mathbb{R}$, binary classification $Y = \{-1, 1\}$ and multi-category (multiclass) classification $Y = \{1, 2, \dots, T\}$. The space $X \times Y$ is called the *data space*.

We assume there exists a fixed unknown data distribution $p(x, y)$ according to which the data are identically and independently distributed (i.i.d.). The probability distribution p models different sources of uncertainty. We assume that it factorizes as $p(x, y) = p_X(x)p(y|x)$, where

- the conditional distribution $p(y|x)$, see Figure 2.1, describes a *non deterministic* relation between input and output.
- The marginal distribution $p_X(x)$ models uncertainty in the sampling of the input points.

We provide two classical examples of data model, namely regression and classification.

Example 1 (Regression). *In regression the following model is often considered $y = f^*(x) + \epsilon$. Here f^* is a fixed unknown function, for example a linear function $f^*(x) = x^T w^*$ for some $w^* \in \mathbb{R}^d$ and ϵ is random noise, e.g. standard Gaussian $\mathcal{N}(0, \sigma)$, $\sigma \in [0, \infty)$.*

Example 2 (Classification). *In binary classification a basic example of data model is a mixture of two Gaussians, i.e. $p(x|y = -1) = \frac{1}{Z}\mathcal{N}(0, \sigma_-)$, $\sigma_- \in [0, \infty)$ and $p(x|y = 1) = \frac{1}{Z}\mathcal{N}(0, \sigma_+)$, $\sigma_+ \in [0, \infty)$, where $\frac{1}{Z}$ is a suitable normalization. For example In classification, a noiseless situation corresponds to $p(1|x) = 1$ or 0 for all x.*

2.3 Loss Function and and Expected Risk

The goal of learning is to estimate the "best" input-output relation– rather than the whole distribution p.

More precisely, we need to fix a *loss function*

$$\ell : Y \times Y \to [0, \infty),$$

which is a (point-wise) measure of the error $\ell(y, f(x))$ we incur in when predicting $f(x)$ in place of y. Given a loss function, the "best" input-output relation is the *target function* $f^* : X \to Y$ minimizing the *expected loss (or expected risk)*

$$\mathcal{E}(f) = \mathbb{E}[\ell(y, f(x))] = \int dxdyp(x, y)\ell(y, f(x)).$$

which can be seen as a measure of the error on past as well as future data. The target function cannot be computed since the probability distribution p is unknown. A (good) learning algorithm should provide a solution that behaves similarly to the target function, and predict/classify well new data. In this case, we say that the algorithm *generalizes.*

Remark 1 (Decision Surface/Boundary). *In classification we often visualize the so called decision boundary (or surface) of a classification solution f. The decision boundary is the level set of points x for which $f(x) = 0$.*

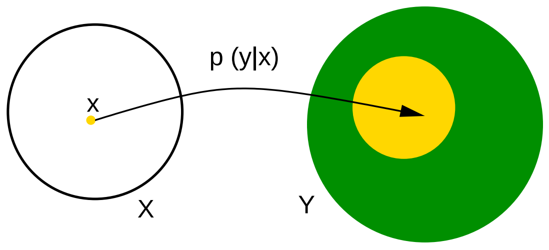

Figure 2.1. For each input x there is a distribution of possible outputs $p(y|x)$.

2.4 Stability, Overfitting and Regularization

A *learning algorithm* is a procedure that given a training set S computes an estimator f_S. Ideally an estimator should *mimic* the target function, in the sense that $\mathcal{E}(f_S) \approx \mathcal{E}(f^*)$. The latter requirement needs some care since f_S depends on the training set and hence is random. For example one possibility is to require an algorithm to be good in *expectation*, in the sense that

$$\mathbb{E}_S \mathcal{E}(f_S) - \mathcal{E}(f^*),$$

is small.

More intuitively, a good learning algorithm should be able to describe well (fit) the data, and at the same time be stable with respect to noise and sampling. Indeed, a key to ensure good generalization property is to avoid overfitting, that is having estimators which are highly dependent on the data (unstable), possibly with a low error on the training set and yet a large error on future data. Most learning algorithms depend one (or more) regularization parameter that control the trade-off between data-fitting and stability. We broadly refer to this class of approaches as regularization algorithms, their study is our main topic of discussion.