Statistical learning and applications

September 25, 2014

Contents

1 Introduction **1**
1.1 Examples 2

2 Learning with high dimensional data **4**
2.1 Bias/variance trade-off and how to deal with it 4
2.2 Summary 6

3 Supervised learning **6**

4 Unsupervised learning **6**

5 Statistical learning theory **6**

About this document

These are lecture notes of the 2014-2015 course on statistical learning, which is a M2 course at ENS Lyon.

Slides of the course are available at `http://lear.inrialpes.fr/people/mairal/teaching/2014-2015/M2ENS/`, the present notes only complement the slides.

1 Introduction

Scribes: Baptiste Jonglez and Stéphane Durand

References for the course:

- Freedman, Hastie, Tibshirani: *The elements of statistical learning*

- `http://www.stat.berkeley.edu/bartlett/courses/2013springstat210b/` (theoretical part)
- `http://www.di.ens.fr/arlot/2013orsay.htm`
- Boyd Vandenberghe: *Convex optimization*
- *Matrix cookbook*: many formulas about matrices (PDF document, to be found online)

1.1 Examples

Simple application of statistical learning: classification, where you are given labels for some objects, and you must guess the label of new objects. Classification is a subpart of supervised learning.

Supervised learning: we are given labels for objects during training. Unsupervised: we don't.

1.1.1 Recommendation systems

Given your rating of movies, recommend new movies that you will probably like.

Netflix: the 1$ million challenge (improve their recommendation algorithm by 10%). The whole machine learning community worked on this for 5 years.

1.1.2 Search engine: relevance of results

This can be seen as a prediction problem.

1.1.3 Natural Language Processing

Predict the topic of a natural text, spam filtering, machine translation, ...

Remark: here (and in the other examples of this section), objects are much more complex than 2-dimension points. We must find **descriptors** for the objects (e.g. for an email: occurrences of a word, set of letters, set of words, etc)

1.1.4 Biology

When dealing with genes, ARN, etc, we have tons of information, but it's hard to sift through it (high-dimensional, etc)

Tumor classification for prognosis The goal is to predict whether the tumor will reappear in a few years, so that you know which treatment to give (chemotherapy). Figure of slide 11: x are genes, y are persons. A green point represents low expression of a gene, red is high expression of a gene. The top of the figure shows bad prognosis patients, the bottom shows good patients. When you have a new sample (= gene expression of a person), then you can try to classify it as either good or bad prognosis.

Molecule classification for drug design A target is a protein whose function we want to alter, because it's not working well. We add a molecule to change the behaviour. Some molecules are active against the target protein, some are not, but we want to find new active molecules (maybe because the current ones have side-effect). Classification problem: find good candidates for the new molecules.

Gene expression clustering Unsupervised learning. Similar to the tumor classification, but we don't have examples of the classes we are trying to identify. Difficult problem. Application: subtypes of cancer (for instance, "breast cancer" can be many different diseases)

DNA reconstruction Phylogenetic tree: if we have the DNA of descendants of a specie, we can look at what's common and reconstruct the genome of the extinct specie. This doesn't work for T-rex or raptors, because they didn't left any descendance (we could get a common ancestor of one of these species and other species which left a descendance). LUCA: Last Universal Common Ancestor, what does it look like?

1.1.5 Computer vision

Image imprinting, not really a prediction/statistical learning, but same formulation.

1.1.6 Music recognition

Popular application of statistical learning (Shazam, etc)

1.1.7 Neuroscience

Record brain activity of people watching video (training).

Then, you look at the activity of their brain when watching a video you don't know, and you can reconstruct an approximate version of the video.

It's not thought-reading, because we only reconstruct the visual input of the brain, not the thoughts themselves.

2 Learning with high dimensional data

Scribes: Baptiste Jonglez and Stéphane Durand

We are dealing with complex objects, with large number of features, but a reduced number of sample.

2.1 Bias/variance trade-off and how to deal with it

2.1.1 Over-fitting, bias/variance trade-off: what is the problem?

Example Regression example on a set of points in 2D (this is not classification). We can try to build a very simple function (affine), or a more complex function (quadratic, polynomial of high degree).

For n points, a polynomial of degree $n-1$ is "optimal", because it goes through all of our points: but it's intuitively a very bad predictor for new points, because it varies a lot. Maybe our goal is wrong: we don't want to decrease the error on the sample points, but on the rest of the population.

Intuition Much liberty in choosing the model. Existence of many "perfect" solution (for distance/error criterion). Re-sampling will change the parameter of these more than it will simple functions.

$\rightarrow$ generalization issue.

Summary The bias/variance tradeoff: too simple functions do not describe the data well, while too complex functions are too specific to the observed sample (**over-fitting**) and generalise poorly to new data.

2.1.2 Complexity vs dimension

Complexity and dimension are related notions. We can sometimes switch from the second to the first by a change of descriptors.

All functions of the previous example were linear when you look hard at them. For a polynomial degree 2 in dimension 1, just take the vector (X, X^2): then the regression problem becomes linear, in dimension 2.

More generally, if you have multiple descriptors, you can sometimes increase dimension to simplify.

2.1.3 Models

No (probabilistic) model for now. Non necessary, but can improve problem understanding.
Some models can be justified using probabilistic models but were originally introduced without these models (e.g. least squares).

2.1.4 Bias-variance decomposition

Assume the data follows y = f(x) + ϵ

ϵ is a **random noise** term of null expectancy.

Both f^ and y are random, the expectation is taken over both. On the other hand, x is not random, the variable of study is $y|x$.

The average mean square error can be decomposed in three terms. A term of variance, the Bayesian error (not reducible), a bias term and a variance term. Proof: use the König-Huygens relation. Was let as exercise.

For the previous example, too simple functions have a large **bias**, while too complex functions have a large **variance**: they will vary a lot for different samples, which produces error.

2.1.5 Generalization

Definitions We call $\mathcal{L} : Y \times Y \mapsto \mathbb{R}$ the *loss function* it is the distance used to evaluate the error. (eg : $||.||_2^2$)
Given an estimator $f : X \mapsto Y$, we write

$$R(f) = \int_{X \times Y} L(y, f(x)) d\mathbb{P} = \mathbb{E}(L(y, f(y))$$

the risk.
One should note that R is not computable unless the joint distribution of (X, Y) is known. We will have to estimate it. If we pose H the function space where to search for f, we have :
The Bayes regret is $R(f) - R^* = (R(f) - \min_{g \in H} R(g)) + (\min_{g \in H} R(g) - R^*)$ with R^* the Bayes risk. The second term is the approximation error.
The first term comes from the fact that R has to be estimated.

[Slide 43] These terms depend on the size of the set of functions: the largest the set of functions, the larger the error term will become.

2.1.6 Rademacher complexity of H

$$R(H) = \mathbb{E}_{\epsilon,Z}\left[\sup|\frac{1}{n}\sum \epsilon_i f(Z_i)|\right]$$

Intuition: we see if H can provide functions that align well with noise (epsilon term of the definition, which is a vector in $\{-1,1\}^n$ and fixed in advance). We don't want our functions to align well with the noise.

It is increasing with H and decreasing with n. It can bound the error of the empirical risk minimization estimator.

$$\mathbb{E}_{x,y}(R(\tilde{f} - R^*) \leq \min_{g\in H}(R(g) - R^*) + 4R(H)$$

2.1.7 Practical implications

How to design estimators: build small classes H which we think contain good approximations.

2.2 Summary

Risk: decomposed as a term of bias/approximation and a term of variance/estimation. This highlights the tradeoff, related to the complexity of the set of functions we consider (either too simple or too complex).

More accurate definitions of complexity: Rademacher, VC [slide 57]
VC dimension := $\sup\{n, \exists(Z_1 \dots Z_n)|\{f(Z_1)\dots f(Z_n)\}| = 2^n\}$
Definition H shatter $(Z_1 \dots Z_n)$ iff $\{f(Z_1)\dots f(Z_n)\}| = 2^n$
Property : $VC(\mathcal{L}(\mathbb{R}^p, \mathbb{R}) = p$

3 Supervised learning

4 Unsupervised learning

5 Statistical learning theory