Statistical Learning Theory

Massimiliano Pontil

Istituto Italiano di Tecnologia
and
University College London

Optimization and Big Data Summer School, Veroli, July 3-7, 2017

Lectures

- Lecture 1: Statistical learning theory
- Lecture 2: Multitask and transfer learning 1
- Lecture 3: Multitask and transfer learning 2
- Lecture 4: Advanced topics
  - Matrix completion algorithm
  - Gradient-based hyperparameter optimization

Today's plan

- Supervised learning
- Statistical learning theory
- Regularization
- Introduction to multitask learning

Learning from data

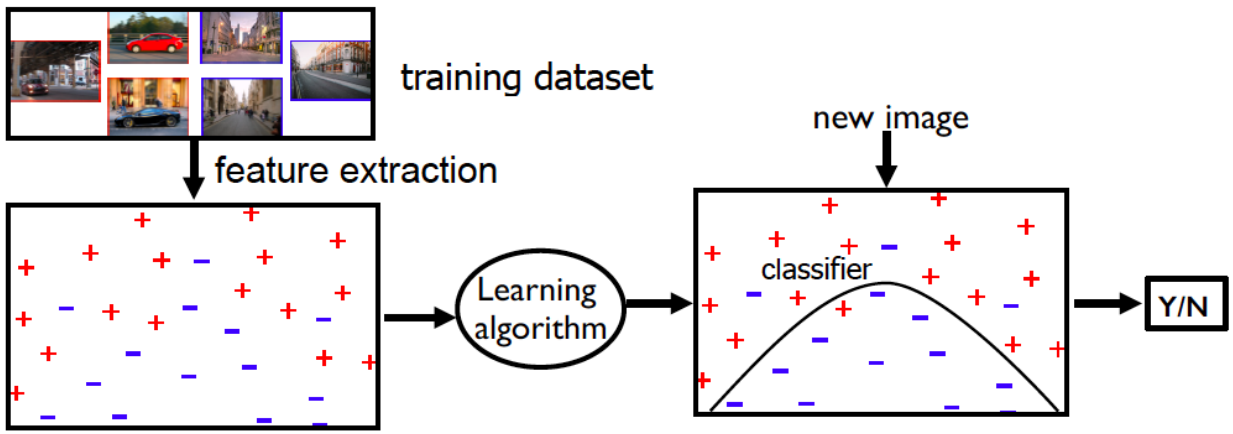
- Let p be a probability distribution on a set $\mathcal{Z}$
- p is unknown, but we can draw from it

$$z_1, ..., z_n \sim p$$

- We wish to learn "properties" of p from the data:
  - density estimation
  - find a "low dimensional" representation of the data
  - supervised learning (prediction): $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$

Supervised learning

- Learn from data to perform a task (as opposed to manually programming it)
- Function approximation problem: map inputs to outputs



- Different settings:
  - binary classification: $\{-1, 1\}$
  - regression: $\mathcal{Y} \subset \mathbb{R}$ (or a subset of it)
  - multiclass classification: $\mathcal{Y} = \{c_1, \ldots, c_k\}$
  - vector-valued regression: $\mathcal{Y} \subset \mathbb{R}^k$
  - structured output prediction: $\mathcal{Y} \neq$ vector space, e.g. a set of sequences

Statistical learning framework

- We assume that there exists a fixed but unknown probability distribution p on $\mathcal{X} \times \mathcal{Y}$
- Let $S = \{(x_i, y_i)\}_{i=1}^{n}$ be an i.i.d. sample (training set) from p
- From S we wish to *estimate* a function $f_S : \mathcal{X} \to \mathcal{Y}$, which captures the relationship between the input and the output
- We see the mapping $S \mapsto f_S$ as a learning algorithm

Statistical learning framework

- Under mild assumptions we can be factorize

$$p(x, y) = p(x)p(y|x)$$

- The marginal probability $p(x)$ models the uncertainty in choosing the inputs
- The conditional probability $p(y|x)$ modes the noise on the outputs (the I/O relationship is in general nondeterministic!)
- **Ex. 1** (regression): $y = f^*(x) + \epsilon$ with $\epsilon \sim N(0, \sigma^2)$
- **Ex. 2** (binary classification): $y = \mathrm{sign}(f^*(x))\epsilon$, with ϵ a binary r.v. modeling labeling noise: $\mathrm{Prob}\{\epsilon = -1\}$ is the probability of flipping a label

Loss function and expected risk

- Let $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be a **loss function**: $\ell(f(x), y)$ measures the loss incurred in predicting the output of x as $f(x)$ when the true output is y
- Example (square loss for regression): $\ell(y', y) = (y' - y)^2$,
- Expected risk:

$$R(f) = \mathbb{E}\big[\ell(f(X), Y)\big] = \int \ell(f(x), y) dp(x, y)$$

- $R(f)$ measures the predictive ability of f
- Our aim is to design an estimator f_S such that:

$$R(f_S) \approx R(f^*) = \min_f R(f)$$

Regression with the square loss

- If $\mathcal{Y} \subseteq \mathbb{R}$ and ℓ is the square loss then $f^*(x) = \int y p(y|x) dy$
- Using the decomposition $p(x, y) = p(x)p(y|x)$ and adding and subtracting f^* inside the loss gives

$$R(f) = \int (f(x) - f^*(x))^2 p(x) dx + R(f^*)$$

- A similar reasoning applies to binary classification with the 0-1-loss: $\ell(y', y) = 1_{y' \neq y}$. Here f^* is called the Bayes classifier

$$f^*(x) = \begin{cases} 1 & \text{if } p(1|x) > \frac{1}{2} \\ -1 & \text{otherwise} \end{cases}$$

Empirical risk minimization (ERM)

The best predictor f^* cannot be computed (because p is unknown). Our aim is to design an estimator f_S, which mimics f^* as well as possible. A natural approach is the following:

- choose a set of functions $\mathcal{F}$ (e.g. set of polynomials)
- search within $\mathcal{F}$ for a function minimizing the **empirical risk**:

$$R_S(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)$$

- let $f_S \in \underset{f \in \mathcal{F}}{\operatorname{argmin}} R_S(f)$ be a solution
- we aim to choose $\mathcal{F}$ so that $R(f_S)$ is as close as possible to $R(f^*)$ (or $f_S \approx f^*$ for regression with the square loss)

Empirical risk minimization (ERM)

$$f_S = \operatorname*{argmin}_{f \in \mathcal{F}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)}_{\textit{empirical error}}$$

Three key problems:

- Function representation/approximation: which $\mathcal{F}$?
- Numerical optimization: iterative schemes to find f_S
- Statistical analysis: derive high probability bounds on the generalization error

$$R(f_S) \leq R_S(f_S) + \epsilon(n, \delta, \mathcal{F})$$

Approximation and sample error

- Frequent choice: $\mathcal{F} = \{f : \|f\| < \infty\}$
- Typically $\mathcal{F}$ is too big, so we consider a subset, e.g. a ball
- Let $f_\alpha \in \underset{\|f\| \leq \alpha}{\operatorname{argmin}} R(f)$
- Decomposition of the excess error (or risk):

$$\underbrace{R(f_S) - R(f^*)}_{\textit{excess error}} \leq \underbrace{2 \sup_{\|f\| \leq \alpha} \{R(f) - R_S(f)\}}_{\textit{sample error}} + \underbrace{\{R(f_\alpha) - R(f^*)\}}_{\textit{approximation error}}$$

Approximation and sample error (cont.)

Proof. Add and subtract $R_S(f_S)$, $R_S(f_\alpha)$ and $R(f_\alpha)$, analyse and simplify:

$$\begin{aligned}
\underbrace{R(f_S) - R(f^*)}_{\textit{excess error}} &= \underbrace{\{R(f_S) - R_S(f_S)\}}_{\textit{generalization error}} + \underbrace{\{R_S(f_S) - R_S(f_\alpha)\}}_{\leq 0} \\
&\quad + \{R_S(f_\alpha) - R(f_\alpha)\} + \{R(f_\alpha) - R(f^*)\} \\
&\leq \sup_{\|f\| \leq \alpha} \Big\{ R(f) - R_S(f) \Big\} + \big\{ R_S(f_\alpha) - R(f_\alpha) \big\} \\
&\quad + \big\{ R(f_\alpha) - R(f^*) \big\} \\
&\leq \underbrace{2 \sup_{\|f\| \leq \alpha} \{R(f) - R_S(f)\}}_{\textit{sample error}} + \underbrace{\{R(f_\alpha) - R(f^*)\}}_{\textit{approximation error}}
\end{aligned}$$

Overfitting and underfitting

- Overfitting: the sample error is too large. The space $\mathcal{F}$ within which we solve ERM is too large (e.g. α is large) $\Rightarrow$ we can always find a function with zero empirical risk
- Underfitting: the approximation error is too large. The space $\mathcal{F}$ is too small (e.g. α is too "small") so even though the sample error is small, we potentially have a poor fit to the data
- Related to the bias/variance decomposition in statistics

Two important learning methods

- L2-norm regularization: encourage smooth solutions
  - Square loss: ridge regression / regularized least squares
  - Leading to kernel methods and SVMs
- L1-norm regularization: encourage sparse solutions
  - Square loss: LASSO
- Both methods are of the form $\min_{\|f\| \leq \alpha} \sum_{i=1}^{n} \ell(f(x_i), y_i)$ or

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \ell(f(x_i), y_i) + \lambda \|f\|$$

- These (and in general any learning algorithm) depend on a number of hyperparamters (here the regularization parameters λ or α) and their choice is key. A common approach is to use a validation set or a cross validation procedure

L2 Regularization and kernel methods

- Choose a *feature map* $\phi : \mathcal{X} \to \mathbb{R}^N$ and solve:

$$\underset{w \in \mathbb{R}^N}{\text{minimize}} \sum_{i=1}^{n} \ell\big(\langle w, \phi(x_i)\rangle, y_i\big) + \lambda \|w\|_2^2$$

- Regularizer favors **smooth** functions, e.g. small Sobolev norm
- Define the **kernel function**: $K(x, x') = \langle \phi(x), \phi(x')\rangle$

  e.g. the Gaussian: $k(x, x') = e^{-\beta \|x - x'\|^2}$
- Solution has the form $w = \sum_{i=1}^{n} c_i \phi(x_i)$, hence

$$f_S(x) = \sum_{i=1}^{n} c_i K(x_i, x)$$

L1 Regularization and sparsity

- Consider the regression model: $y = \sum_{j=1}^{N} w_j^* \phi_j(x) + \epsilon$, where ϵ is a zero mean noise vector
- $w^* \in \mathbb{R}^N$ is assumed to be **sparse** (or approximately so)
- Goal: to estimate w^* or its sparsity pattern
- Use the ℓ_1 norm to encourage sparse solutions

$$\min_{w \in \mathbb{R}^N} \sum_{i=1}^{n} \ell(\langle w, \phi(x_i) \rangle, y_i)^2 + \lambda \|w\|_1$$

Multitask learning (MTL)

- Lack of training data in specific applications
- Uniform lower bounds tell us that learning is not possible without strong prior knowledge
- Often multiple datasets/sources studying similar or the same task/problem are available
- Multitask learning (informal reasoning): can we leverage similarities across the tasks?

Multitask learning (MTL)

- Fix probability measures $p_1, \ldots, p_T$ on $\mathbb{R}^d \times \mathbb{R}$
- Draw data: $(x_{t1}, y_{t1}), \ldots, (x_{tn}, y_{tn}) \sim p_t, \quad t = 1,...,T$
  (in practice n may vary with t)
- Learning method: $$\min_{(f_1,...,f_T)\in\mathcal{F}} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{n}\sum_{i=1}^{n}\ell(f_t(x_{ti}), y_{ti})$$
- $\mathcal{F}$ is a set of vector-valued functions and provides a means to model interactions between the tasks
- Multitask risk: $R(f_1, ..., f_T) = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{(x,y)\sim p_t} \ell(f_t(x), y)$

Key questions

- How to model or encourage interactions between the tasks? (which set $\mathcal{F}$ or associated regularizer Ω?)
- Design efficient algorithms to solve the associated optimization problem
- Under which condition is MTL is advantageous (small excess risk)?
- Transfer learning: how to use knowledge accumulated by solving a set of related tasks in order to solve a new one?

Applications

- **User modelling:**
  - ◇ each task is to predict a user's ratings of products
  - ◇ the ways different people make decisions about products are related

- **Multiple object detection in scenes:**
  - ◇ detection of each object corresponds to a binary classification task: $y_{ti} \in \{-1, 1\}$
  - ◇ learning common features enhances performance
  - ◇ early work in using multilayer neural networks (now called deep networks) with shared hidden weights

Many more: affective computing, bioinformatics, health informatics, marketing science, neuroimaging, NLP, speech,...

Linear MTL

- "task" = "linear model"
  - Regression: $y_{ti} = \langle w_t^*, x_{ti} \rangle + \epsilon_{ti}$
  - Binary classification: $y_{ti} = \mathrm{sign}\langle w_t^*, x_{ti} \rangle \epsilon_{ti}$
- Learning method: $$\min_{[w_1,\ldots,w_T] \in \mathcal{C}} \frac{1}{T} \sum_{t=1}^{T} \frac{1}{n} \sum_{i=1}^{n} \ell(\langle w_t, x_{ti} \rangle, y_{ti})$$
- Set $\mathcal{C}$ encourages "common structure" among tasks, e.g. the ball of a matrix norm or other regularizer
- Independent task learning (ITL): $\mathcal{C} = \underbrace{\mathcal{B} \times \cdots \times \mathcal{B}}_{\text{T times}}$

Linear MTL via matrix regularization

$$\min_{[w_1,\dots,w_T]\in\mathcal{C}} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{n}\sum_{i=1}^{n}\ell(\langle w_t, x_{ti}\rangle, y_{ti})$$

Often we drop the constraint and consider penalty methods

$$\min_{w_1,\dots,w_T} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{n}\sum_{i=1}^{n}\ell(\langle w_t, x_{ti}\rangle, y_{ti}) + \lambda\Omega(w_1,\dots,w_T)$$

- This leads to a matrix regularization problem.
- We'll see that different regularizers encourage different types of commonalities between the tasks:
  - Small variance: encourages closeness to the mean
  - Sparsity: encourages few shared variables
  - Low rankness: encourage few shared features

Summary

In MTL we typically have **many tasks** but only **few examples per task**. If $n < d$ we don't have enough data to learn the tasks one by one. However if the tasks are *"related"* and the constraint set $\mathcal{C}$ or the associated regularizer Ω capture such relationships, learning the tasks *jointly* improves over ITL. In the next lectures we will present some of the mainstream MTL methods and address their computational and statistical proprieties.