Mathematical Foundations of Data Sciences

Gabriel Peyré
CNRS & DMA
École Normale Supérieure
gabriel.peyre@ens.fr
www.gpeyre.com
www.numerical-tours.com

October 7, 2017

Chapter 4

Machine Learning

4.1 Supervised vs Unsupervised Learning

TOOD: describe the general settings and problems.

4.2 PCA, Nearest-Neighbors Classification and Clustering

4.2.1 Dimenionality Reduction and PCA

We detail Principal Component Analysis (dimentionality reduction), supervised classification using nearest neighbors and unsupervised clustering using k-means.

We use here the famous IRIS dataset of Fisher. The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres.

The feature are stored as the row of a matrix $X \in \mathbb{R}^{n \times p}$

n is the number of samples, p is the dimensionality of the features, k is the number of classes.

In order to display in 2-D or 3-D the data, dimensionality reduction is needed. The simplest method is the Principal Component Analysis (PCA), which perform an orthogonal linear projection on the principal axis (eigenvector) of the covariance matrix.

We compute the empirical mean

$$m = \frac{1}{n} \sum_{i=1}^{n} x_i \in \mathbb{R}^p$$

and covariance

$$C = \frac{1}{n} \sum_{i=1}^{n} (x_i - m)(x_i - m)^\top \in \mathbb{R}^{p \times p}.$$

Denoting $\tilde{X} = X - 1_p m^\top$, one has $C = \tilde{X}^\top \tilde{X}$.

We compute PCA ortho-basis using the SVD decomposition

$$\tilde{X} = U \operatorname{diag}(d) V$$

where $U \in \mathbb{R}^{n \times p}$ and $V \in \mathbb{R}^{p \times p}$ have orthonormal columns. V are the principal directions of variance and are order by decreasing variances.

We then compute the feature in the PCA basis, $z_i = V^\top (x_i - m)$, stored in matrix format as $Z = \tilde{X} V$.

One can plot the singular values of the covariances, which corresponds to the standard deviation of the data along the principal directions.

The first dimensions of the z_i are the optimal way to linearly embed the data in a low dimensional space. This can be used for display in 2-D using the first two dimension. One can do a similar display in 3-D.

4.2.2 Supervised Learning: Nearest Neighbor Classification

Probably the simplest method for supervised classification is Nearest Neighbor (R-NN), where R is a parameter indexing the number of neighbor. Increasing R is important to cope with noise and obtain smoother decision boundary, and hence better generalization performance.

The class predicted for a point x is the one which is the most represented among the R points $(x_i)_i$ which are the closed to x.

The data needs to be split into training and testing.

One then compute Euclidean distance between some x and all other $x_{1,j}$ in the training set. Then sort the distance and generate the list of sorted classes $y_\sigma = (y_{\sigma(i)})_i$. This generate an indexing σ (a permutation of $\{1, \ldots, n\}$) such that

$$\|x - x_{\sigma(1)}\| \leqslant \|x - x_{\sigma(2)}\| \leqslant \ldots \leqslant \|x - x_{\sigma(n)}\|.$$

For a given R, one can compute the histogram of class apparition

$$h_\ell \overset{\text{def.}}{=} \frac{1}{R} \{i \,;\, \sigma(i) \in \{1, \ldots, R\}\} = \sharp \sigma^{-1}(\{1, \ldots, R\}).$$

The decision class for x is then the maximum of the histogram

$$c(x) \overset{\text{def.}}{=} \operatorname*{argmax}_\ell h_\ell$$

One can display the histogram $(h_\ell)_\ell$ of repartition of class indexes as R grows. Then perform the NN classification for all the points in the test set, and for varying R. And show how the classification score S (number of correctly classified points) evolves with R.

One can also display, as a function of the position in 2-D PCA space, the class output by the R-NN method when applied in 2-D.

4.2.3 Unsupervised Learning: k-means

In an un-supervised setting, the class information y is not available. The basic problem is then to recover class information from the knowledge of x only. This corresponds to the clustering problem.

The most basic algorithm is the k-means, which tries to recover the class index $\bar{y}_i = \ell$ from the distance $\|x_i - c_\ell\|$ between the feature point x_i and the class centroid c_ℓ (which are the unknown of the problem).

It does so by minimizing the following non-convex energy

$$\min_{(c_\ell)_\ell} \sum_i \min_\ell \|x_i - c_\ell\|^2$$

We first initialize the class centroids $(c_\ell)_\ell$ at random among the points. They are stored in as the row of a matrix $C \in \mathbb{R}^{k \times p}$.

The k-means algorithm iterate between first determining the class of each point using the distance to the centroids

$$\forall i \in \{1, \ldots, n\}, \quad \bar{y}_i \leftarrow \operatorname*{argmin}_\ell \|x_i - c_\ell\|.$$

One can display the centroids and the classes using colors. This corresponds to a Voronoi diagram segmentation in the high dimensional space, but here the display is done in 2D.

The second step of the k-means algorithm is to update the centroids position to be the mean of the points inside each class

$$\forall \ell \in \{1, \ldots, k\}, \quad c_\ell \leftarrow \frac{\sum_{i : y_i = \ell} x_i}{\sharp\{i : y_i = \ell\}}.$$

One can then perform several step of the k-means algorithm. And display the histogram of (true, i.e. according to y) class inside each estimated class (i.e. according to $\bar{y}$).

It is possible to implement better initialization strategies such as farthest point sampling or k-means++.

4.3 Linear Regression and Kernel Methods

We now study linear regression method, and its non-linear variant using kernelization.

4.3.1 Linear Regression

We test the method on the Boston house prices dataset, consisting in $n = 506$ samples with features $x_i \in \mathbb{R}^p$ in dimension $p = 13$. The goal is to predict the price value $y_i \in \mathbb{R}$.

We separate the features X from the data y to predict information. n is the number of samples, p is the dimensionality of the features,

In order to display in 2-D or 3-D the data, dimensionality is needed. The simplest method is the principal component analysis, which perform an orthogonal linear projection on the principal axsis (eigenvector) of the covariance matrix.

We look for a linear relationship $y_i = \langle w, x_i \rangle$ written in matrix format $y = Xw$ where the rows of $X \in \mathbb{R}^{n \times p}$ stores the features $x_i \in \mathbb{R}^p$.

Since here $n > p$, this is an over-determined system, which can solved in the least square sense

$$\min_w \|Xw - y\|^2,$$

whose solution is given using the Moore-Penrose pseudo-inverse

$$w = (X^\top X)^{-1} X^\top y.$$

Regularization is obtained by introducing a penalty. It is often called ridge regression, and is defined as

$$\min_w \|Xw - y\|^2 + \lambda \|w\|^2$$

where $\lambda > 0$ is the regularization parameter.

The solution is given using the following equivalent formula

$$w = (X^\top X + \lambda \mathrm{Id}_p)^{-1} X^\top y,$$

$$w = X^\top (XX^\top + \lambda \mathrm{Id}_n)^{-1} y,$$

When $p < n$ (which is the case here), the first formula should be preferred.

In contrast, when the dimensionality p of the feature is very large and there is little data, the second is faster. Furthermore, this second expression is generalizable to Kernel Hilbert space setting, corresponding possibly to $p = +\infty$ for some kernels.

4.3.2 Kernelized Ridge Regression

In order to perform non-linear and non-parametric regression, it is possible to use kernelization. It is non-parametric in the sense that the number of parameter grows with the number n of samples (while for the initial linear method, the number of parameter is p). This allows in particular to generate estimator of arbitrary complexity.

Given a kernel $\kappa(x, z) \in \mathbb{R}$ defined for $(x, z) \in \mathbb{R}^p \times \mathbb{R}^p$, the kernelized method replace the linear approximation functional $f(x) = \langle x, w \rangle$ by a sum of kernel centered on the samples

$$f_h(x) = \sum_{i=1}^n h_i k(x_i, x)$$

where $h \in \mathbb{R}^n$ is the unknown vector of weight to find.

When using the linear kernel $\kappa(x, y) = \langle x, y \rangle$, one retrieves the previously studied linear method.

The gaussian kernel is the most well known and used kernel

$$\kappa(x,y) \overset{\text{def.}}{=} e^{-\frac{\|x-y\|^2}{2\sigma^2}}.$$

The bandwidth parameter $\sigma > 0$ is crucial and controls the locality of the model. It is typically tuned through cross validation.

Once evaluated on grid points, the kernel define a matrix

$$K = (\kappa(x_i, x_j))_{i,j=1}^n \in \mathbb{R}^{n\times n}.$$

The weights $h \in \mathbb{R}^n$ are solutions of

$$\min_h \|Kh - y\|^2 + \lambda\langle Kh,\, h\rangle$$

and hence can be computed by solving a linear system

$$h = (K + \lambda \mathrm{Id}_n)^{-1} y$$

4.4 Logistic Classification

We now detail the logistic classification method (for 2 classes and multi-classes).

Note thatlLogisitic classification is actually called "logistic regression" in the literature, but it is in fact a classification method.

4.4.1 Two Classes Logistic Classification

Logistic classification is, with support vector machine (SVM), the baseline method to perform classification. Its main advantage over SVM is that is is a smooth minimization problem, and that it also output class probabity, offering a probabilistic interpretation of the classification.

To understand the behavior of the method, we generate synthetic data distributed according to a mixture of Gaussian with an overlap governed by an offset ω. Here classes indexes are set to $y_i \in \{-1, 1\}$ to simplify the equations.

Logistic classification minimizes a logistic loss in place of the usual ℓ^2 loss for regression

$$\min_w E(w) \overset{\text{def.}}{=} \frac{1}{n}\sum_{i=1}^n L(\langle x_i,\, w\rangle, y_i)$$

where the logistic loss reads

$$L(s,y) \overset{\text{def.}}{=} \log(1 + \exp(-sy))$$

This corresponds to a smooth convex minimization. If X is injective, this is also strictly convex, hence it has a single global minimum.

One can compare the binary (ideal) 0-1 loss, the logistic loss and the hinge loss (the one used for SVM).

This can be interpreted as a maximum likelihood estimator when one models the probability of belonging to the two classes for sample x_i as

$$h(x_i) \overset{\text{def.}}{=} (\theta(x_i), 1 - \theta(x_i)) \quad \text{where} \quad \theta(s) \overset{\text{def.}}{=} \frac{e^s}{1+e^s} = (1+e^{-s})^{-1}$$

Re-writting the energy to minimize

$$E(w) = \mathcal{L}(Xw, y) \quad \text{where} \quad \mathcal{L}(s,y) = \frac{1}{n}\sum_i L(s_i, y_i),$$

its gradient reads

$$\nabla E(w) = X^\top \nabla \mathcal{L}(Xw, y) \quad \text{where} \quad \nabla \mathcal{L}(s, y) = \frac{y}{n} \odot \theta(-y \odot s),$$

where $\odot$ is the pointwise multiplication operator, i.e. `.*` in Matlab.

In order to improve performance, it is important (especially in low dimension p) to add a constant bias term $w_{p+1} \in \mathbb{R}$, and replace $\langle x_i, w \rangle$ by $\langle x_i, w \rangle + w_{p+1}$. This is equivalently achieved by adding an extra $(p+1)^{\text{th}}$ dimension equal to 1 to each x_i, which we do using a convenient macro.

With this added bias term, once $w_{\ell=0} \in \mathbb{R}^{p+1}$ initialized (for instance at 0_{p+1}),

one step of gradient descent reads

$$w_{\ell+1} = w_\ell - \tau_\ell \nabla E(w_\ell).$$

One can then implement a gradient descent

$$w_{\ell+1} = w_\ell - \tau_\ell \nabla E(w_\ell).$$

One can display the data overlaid on top of the classification probability, this highlight the separating hyperplane $\{x \,;\, \langle w, x \rangle = 0\}$.

4.4.2 Kernelized Logistic Classification

Logistic classification tries to separate the classes using a linear separating hyperplane $\{x \,;\, \langle w, x \rangle = 0\}$.

In order to generate a non-linear descision boundary, one can replace the parametric linear model by a non-linear non-parametric model, thanks to kernelization. It is non-parametric in the sense that the number of parameter grows with the number n of sample (while for the basic method, the number of parameter is p. This allows in particular to generate decision boundary of arbitrary complexity.

The downside is that the numerical complexity of the method grows (at least) quadratically with n.

The good news however is that thanks to the theory of reproducing kernel Hilbert spaces (RKHS), one can still compute this non-linear decision function using (almost) the same numerical algorithm.

Given a kernel $\kappa(x, z) \in \mathbb{R}$ defined for $(x, z) \in \mathbb{R}^p$, the kernelized method replace the linear decision functional $f(x) = \langle x, w \rangle$ by a sum of kernel centered on the samples

$$f_h(x) = \sum_{i=1}^{p} h_i k(x_i, x)$$

where $h \in \mathbb{R}^n$ is the unknown vector of weight to find.

When using the linear kernel $\kappa(x, y) = \langle x, y \rangle$, one retrieves the previously studied linear method.

The gaussian kernel is the most well known and used kernel

$$\kappa(x, y) \overset{\text{def.}}{=} e^{-\frac{\|x-y\|^2}{2\sigma^2}}.$$

The bandwidth parameter $\sigma > 0$ is crucial and controls the locality of the model. It is typically tuned through cross validation.

Once evaluated on grid points, the kernel define a matrix

$$K = (\kappa(x_i, x_j))_{i,j=1}^n \in \mathbb{R}^{n \times n}.$$

Valid kernels are those that gives rise to positive symmetric matrices K. The linear and Gaussian kernel are valid kernel functions. Other popular kernels include the polynomial kernel $\langle x, y \rangle^a$ for $a \geqslant 1$ and the Laplacian kernel $\exp(-\|x - y\|^2/\sigma)$.

The kernelized Logistic minimization reads

$$\min_h F(h) \overset{\text{def.}}{=} \mathcal{L}(Kh, y).$$

This minimization can be related to an infinite dimensional optimization problem where one minimizes directly over the function f. This is shown to be equivalent to the above finite-dimenisonal optimization problem thanks to the theory of RKHS. One can then use a gradient descent to minimize $F(h)$.

Once this optimal h has been found, class probability at a point x are obtained as

$$(\theta(f_h(x)), 1 - \theta(f_h(x))$$

where f_h has been defined above.

One can separate the dataset into a training set and a testing set. Evaluate the classification performance for varying σ. Try to introduce regularization and minmize

$$\min_h F(h) \overset{\text{def.}}{=} \mathcal{L}(Kh, y) + \lambda R(h)$$

where for instance $R = \|\cdot\|_2^2$ or $R = \|\cdot\|_1$.

4.4.3 Multi-Classes Logistic Classification

The logistic classification method is extended to an arbitrary number k of classes by considering a family of weight vectors $(w_\ell)_{\ell=1}^k$, which are conveniently stored as columns of matrix $W \in \mathbb{R}^{p\times k}$.

This allows to model probabilistically the belonging of a point $x \in \mathbb{R}^p$ to a the classes using an exponential model

$$h(x) = \left(\frac{e^{-\langle x,\, w_\ell \rangle}}{\sum_m e^{-\langle x,\, w_m \rangle}} \right)_\ell$$

This vector $h(x) \in [0,1]^k$ describes the probability of x belonging to the different classes, and $\sum_\ell h(x)_\ell = 1$.

The computation of w is obtained by solving a maximum likelihood estimator

$$\max_{w\in\mathbb{R}^k} \frac{1}{n} \sum_{i=1}^n \log(h(x_i)_{y_i})$$

where we recall that $y_i \in \{1, \ldots, k\}$ is the class index of point x_i.

This is conveniently rewritten as

$$\min_w \sum_i \mathrm{LSE}(XW)_i - \langle XW,\, D \rangle$$

where $D \in \{0,1\}^{n\times k}$ is the binary class index matrices

$$D_{i,\ell} = \begin{cases} 1 & \text{if} \quad y_i = \ell, \\ 0 & \text{otherwise.} \end{cases}$$

and LSE is the log-sum-exp operator

$$\mathrm{LSE}(S) = \log\left(\sum_\ell \exp(S_{i,\ell}) \right) \in \mathbb{R}^n.$$

The computation of LSE is unstable for large value of $S_{i,\ell}$ (numerical overflow, producing NaN), but this can be fixed by substracting the largest element in each row, since $\mathrm{LSE}(S + a) = \mathrm{LSE}(S) + a$ if a is constant along rows. This is the LSE trick.

The gradient of the LSE operator is the soft-max operator

$$\nabla \mathrm{LSE}(S) = \mathrm{SM}(S) \overset{\text{def.}}{=} \left(\frac{e^{S_{i,\ell}}}{\sum_m e^{S_{i,m}}} \right)$$

Similarly to the LSE, it needs to be stabilized.

We load a dataset of n images of size $p = 8 \times 8$, representing digits from 0 to 9 (so there are $k = 10$ classes). Separate the features X from the data y to predict information. n is the number of samples, p is the dimensionality of the features, k the number of classes. One can display a few samples digits

We compute the D matrix and define the energy $E(W)$. Then define its gradients

$$\nabla E(W) = \frac{1}{n} X^\top (\mathrm{SM}(XW) - D).$$

and one can use a gradient descent

$$W_{\ell+1} = W_\ell - \tau_\ell \nabla E(W_\ell).$$

We can evaluate class probability associated to weight vectors on this grid.

Bibliography

[1] Amir Beck. *Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications with MATLAB*. SIAM, 2014.

[2] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[4] E. J. Candès. The restricted isometry property and its implications for compressed sensing. *Compte Rendus de l'Académie des Sciences*, Serie I(346):589–592, 2006.

[5] A. Chambolle. An algorithm for total variation minimization and applications. *J. Math. Imaging Vis.*, 20:89–97, 2004.

[6] Antonin Chambolle, Vicent Caselles, Daniel Cremers, Matteo Novaga, and Thomas Pock. An introduction to total variation for image analysis. *Theoretical foundations and numerical methods for sparse recovery*, 9(263-340):227, 2010.

[7] Antonin Chambolle and Thomas Pock. An introduction to continuous optimization for imaging. *Acta Numerica*, 25:161–319, 2016.

[8] Philippe G Ciarlet. Introduction à l'analyse numérique matricielle et à l'optimisation. 1982.

[9] Simon Foucart and Holger Rauhut. *A mathematical introduction to compressive sensing*, volume 1. Birkhäuser Basel, 2013.

[10] Stephane Mallat. *A wavelet tour of signal processing: the sparse way*. Academic press, 2008.

[11] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[12] Otmar Scherzer, Markus Grasmair, Harald Grossauer, Markus Haltmeier, Frank Lenzen, and L Sirovich. *Variational methods in imaging*. Springer, 2009.