Mathematical Foundations of Data Sciences

Gabriel Peyré
CNRS & DMA
École Normale Supérieure
gabriel.peyre@ens.fr
https://mathematical-tours.github.io
www.numerical-tours.com

November 15, 2020

Chapter 12

Basics of Machine Learning

This chapter gives a rapid overview of the main concepts in machine learning. The goal is not to be exhaustive, but to highlight representative problems and insist on the distinction between unsupervised (vizualization and clustering) and supervised (regression and classification) setups. We also shed light on the tight connexions between machine learning and inverse problems.

While imaging science problems are generally concern with processing a single data (e.g. an image), machine learning problem is rather concern with analysing large collection of data. The focus (goal and performance measures) is thus radically different, but quite surprisingly, it uses very similar tools and algorithm (in particular linear models and convex optimization).

12.1 Unsupervised Learning

In unsupervised learning setups, one observes n points $(x_i)_{i=1}^n$. The problem is now to infer some properties for this points, typically for vizualization or unsupervised classication (often called clustering). For simplicity, we assume the data are points in Euclidean space $x_i \in \mathbb{R}^p$ (p is the so-called number of features). These points are conveniently stored as the rows of a matrix $X \in \mathbb{R}^{n \times d}$.

12.1.1 Dimensionality Reduction and PCA

Dimensionality reduction is useful for vizualization. It can also be understood as the problem of feature extraction (determining which are the relevant parameters) and this can be later used for doing other tasks more efficiently (faster and/or with better performances). The simplest method is the Principal Component Analysis (PCA), which performs an orthogonal linear projection on the principal axes (eigenvectors) of the covariance matrix.

Presentation of the method. The empirical mean is defined as

$$\hat{m} \stackrel{\text{def.}}{=} \frac{1}{n} \sum_{i=1}^n x_i \in \mathbb{R}^p$$

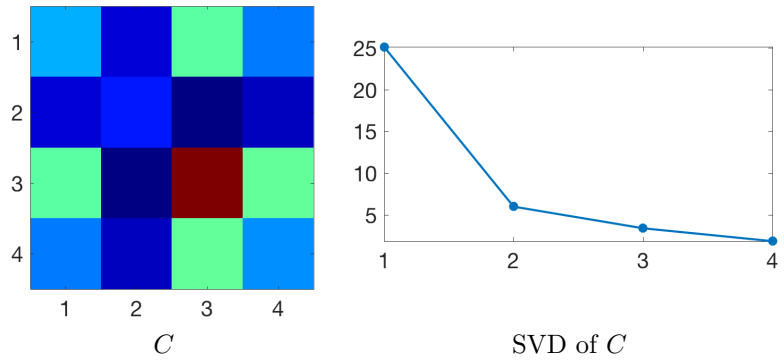
and covariance

$$\hat{C} \stackrel{\text{def.}}{=} \frac{1}{n} \sum_{i=1}^n (x_i - \hat{m})(x_i - \hat{m})^* \in \mathbb{R}^{p \times p}. \tag{12.1}$$

Denoting $\tilde{X} \stackrel{\text{def.}}{=} X - 1_p \hat{m}^*$, one has $\hat{C} = \tilde{X}^* \tilde{X}/n$.

Note that if the points $(x_i)_i$ are modelled as i.i.d. variables, and denoting $\mathbf{x}$ one of these random variables, one has, using the law of large numbers, the almost sure convergence as $n \to +\infty$

$$\hat{m} \to m \stackrel{\text{def.}}{=} \mathbb{E}(\mathbf{x}) \quad \text{and} \quad \hat{C} \to C \stackrel{\text{def.}}{=} \mathbb{E}((\mathbf{x} - m)(\mathbf{x} - m)^*). \tag{12.2}$$



C SVD of C

Figure 12.1: Empirical covariance of the data and its associated singular values.

Denoting μ the distribution (Radon measure) on $\mathbb{R}^p$ of $\mathbf{x}$, one can alternatively write

$$m = \int_{\mathbb{R}^p} x \mathrm{d}\mu(x) \quad \text{and} \quad C = \int_{\mathbb{R}^p} (x-m)(x-m)^* \mathrm{d}\mu(x).$$

The PCA ortho-basis, already introduced in Section 22, corresponds to the right singular vectors of the centred data matrix, as defined using the (reduced) SVD decomposition

$$\tilde{X} = \sqrt{n} U \operatorname{diag}(\sigma) V^*$$

where $U \in \mathbb{R}^{n \times r}$ and $V \in \mathbb{R}^{p \times r}$, and where $r = \operatorname{rank}(\tilde{X}) \leqslant \min(n,p)$. We denote $V = (v_k)_{k=1}^r$ the orthogonal columns (which forms an orthogonal system of eigenvectors of $\hat{C} = V \operatorname{diag}(\sigma^2) V^\top$), $v_k \in \mathbb{R}^p$. The intuition is that they are the main axes of "gravity" of the point cloud $(x_i)_i$ in $\mathbb{R}^p$. We assume the singular values are ordered, $\sigma_1 \geqslant \ldots \geqslant \sigma_r$, so that the first singular values capture most of the variance of the data.

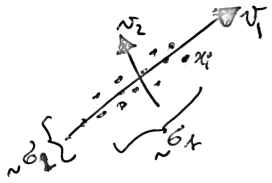


Figure 12.2: PCA main axes capture variance

Figure 12.1 displays an example of covariance and its associated spectrum σ. The points $(x_i)_i$ correspond to the celebrated IRIS dataset[1] of Fisher. This dataset consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). The dimensionality of the features is $p = 4$, and the dimensions corresponds to the length and the width of the sepals and petals.

The PCA dimensionality reduction embedding $x_i \in \mathbb{R}^p \mapsto z_i \in \mathbb{R}^d$ in dimension $d \leqslant p$ is obtained by projecting the data on the first d singular vector

$$z_i \overset{\text{def.}}{=} (\langle x_i - m, v_k \rangle)_{k=1}^d \in \mathbb{R}^d. \tag{12.3}$$

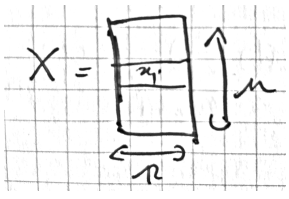
From these low-dimensional embedding, one can reconstruct back an approximation as

$$\tilde{x}_i \overset{\text{def.}}{=} m + \sum_k z_{i,k} v_k \in \mathbb{R}^p. \tag{12.4}$$

One has that $\tilde{x}_i = \operatorname{Proj}_{\tilde{T}}(x_i)$ where $\tilde{T} \overset{\text{def.}}{=} m + \operatorname{Span}_{k=1}^d(v_k)$ is an affine space.

Figure 12.3 shows an example of PCA for 2-D and 3-D vizualization.

Optimality analysis. We now show that among all possible linear dimensionality reduction method, PCA is optimal in sense of ℓ^2 error. To simplify, without loss of generality (since it can be subtracted from the data) we assume that empirical mean is zero $\hat{m} = 0$ so that $X = \tilde{X}$.



[1] https://en.wikipedia.org/wiki/Iris_flower_data_set

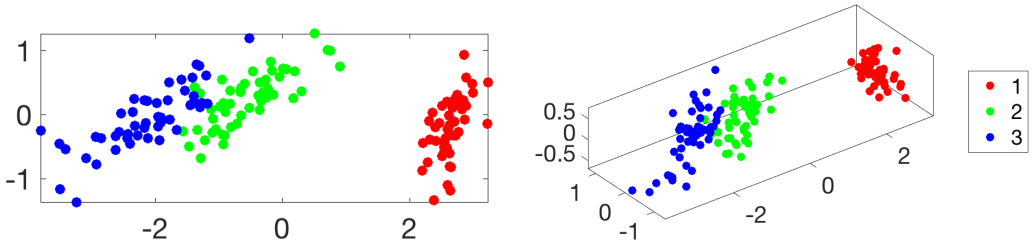


Figure 12.3: 2-D and 3-D PCA vizualization of the input clouds.

We recall that $X = \sqrt{n} U \operatorname{diag}(\sigma) V^\top$ and $\hat{C} = \frac{1}{n} X^\top X = U \Lambda U^\top$ where $\Lambda = \operatorname{diag}(\lambda_i = \sigma_i^2)$, where $\lambda_1 \geqslant \ldots \geqslant \lambda_r$.

The following proposition shows that PCA is optimal in term of ℓ^2 distance if one consider only affine spaces. This means that we consider the following compression/decompression to a dimension k (i.e. dimensionality reduction and then expansion)

$$\min_{R,S} \left\{ f(R,S) \overset{\text{def.}}{=} \sum_i \|x_i - R S^\top x_i\|_{\mathbb{R}^p}^2 \;;\; R, S \in \mathbb{R}^{p \times k} \right\} \tag{12.5}$$

Note that this minimization is a priori not trivial to solve because, although $f(\cdot, S)$ and $f(R, \cdot)$ are convex, f is not jointly convex. So iteratively minimizing on R and S might fail to converge to a global minimizer. This section aims at proving the following theorem.

Theorem 20. *A solution of (12.5) is* $S = R = V_{1:k} \overset{\text{def.}}{=} [v_1, \ldots, v_k]$.

Note that using such a compressor R and decompressor $R = S$ corresponds exactly to the PCA method (12.3) and (12.4).

We first prove that one can restrain its attention to orthogonal projection matrix.

Lemma 7. *One can assume* $S = R$ *and* $S^\top S = \mathrm{Id}_{k \times k}$.

Proof. We consider an arbitrary pair (R, S). Since the matrix $RS^\top$ has rank $k' \leqslant k$, let $W \in \mathbb{R}^{p \times k'}$ be an ortho-basis of $\operatorname{Im}(RS^\top)$, so that $W^\top W = \mathrm{Id}_{k' \times k'}$. We remark that

$$\operatorname*{argmin}_z \|x - Wz\|^2 = W^\top x$$

because the first order condition for this problem reads $W^\top(Wz - x) = 0$. Hence, denoting $RS^\top x_i = W z_i$ for some $z_i \in \mathbb{R}^{k'}$

$$f(R,S) = \sum_i \|x_i - RS^\top x_i\|^2 = \sum_i \|x_i - W z_i\|^2 \geqslant \sum_i \|x_i - WW^\top x_i\|^2 \geqslant f(\tilde{W}, \tilde{W}).$$

were we have extended W in an orthogonal matrix $\tilde{W} \in \mathbb{R}^{p \times k}$ where $\tilde{W}_{1:k'} = W$. □

Lemma 8. *Denoting* $C \overset{\text{def.}}{=} XX^\top \in \mathbb{R}^{p \times p}$, *an optimal* S *is obtained by solving*

$$\max_{S \in \mathbb{R}^{p \times k}} \left\{ \operatorname{tr}(S^\top C S^\top) \;;\; S^\top S = \mathrm{Id}_k \right\}.$$

Proof. Using the previous lemma, one can consider only $R = S$ with $S^\top S = \mathrm{Id}_k$ so that one needs to solve

$$f(S,S) = \sum_i \|x_i S S^\top x_i\|^2 = \sum_i \|x_i\|^2 - 2 x_i^\top S S^\top x_i + x_i^\top S (S^\top S) S^\top x_i.$$

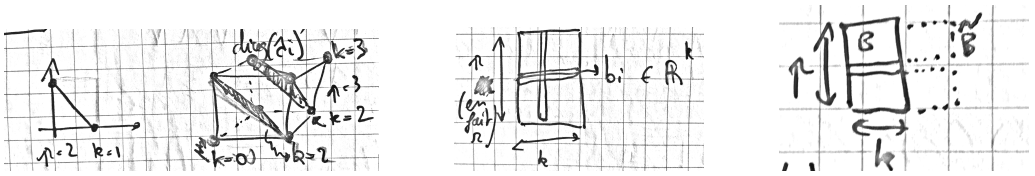


Figure 12.4: Left: proof of the rightmost inequality in (12.6). Middle: matrix B, right: matrix $\tilde{B}$.

Using that $S^\top S = \mathrm{Id}_k$, one has

$$f(S,S) = \mathrm{cst} - \sum_i x_i^\top S S^\top x_i = -\sum_i \mathrm{tr}(x_i^\top S S^\top x_i) = -\sum_i \mathrm{tr}(S^\top x_i x_i^\top S) = -\mathrm{tr}(S^\top (\sum_i x_i x_i^\top) S).$$

□

The next lemma provides an upper bound on the quantity being minimized as the solution of a convex optimization problem (a linear program). The proof of the theorem follows by showing that this upper bound is actually reached, which provides a certificate of optimality.

Lemma 9. *Denoting $C = V\Lambda V^\top$, one has*

$$\mathrm{tr}(S^\top C S) \leqslant \max_{\beta \in \mathbb{R}^p} \left\{ \sum_{i=1}^p \lambda_i \beta_i \;;\; 0 \leqslant \beta \leqslant 1, \sum_i \beta_i \leqslant k \right\} = \sum_{i=1}^k \lambda_i \tag{12.6}$$

i.e. the maximum on the right hand size is $\beta = (1, \ldots, 1, 0, \ldots, 0)$.

Proof. We extend $V \in \mathbb{R}^{p\times k}$ into an orthogonal matrix $\tilde{V} \in \mathbb{R}^{p\times p}$ (i.e. $\tilde{V}_{1:r} = V$) such that $\tilde{V}\tilde{V}^\top = \tilde{V}^\top V = \mathrm{Id}_p$. Similarly we extend Λ into $\tilde{\Lambda}$ by adding zeros, so that $C = \tilde{V}\tilde{\Lambda}\tilde{V}^\top$. One has

$$\mathrm{tr}(S^\top C S) = \mathrm{tr}(S^\top \tilde{V}\tilde{\Lambda}\tilde{V}^\top S) = \mathrm{tr}(B^\top \Lambda B) = \mathrm{tr}(\Lambda B B^\top) = \sum_{i=1}^p \lambda_i \|b_i\|^2 = \sum_i \lambda_i \beta_i$$

where we denoted $B \overset{\text{def.}}{=} V^\top S \in \mathbb{R}^{p\times k}$, $(b_i)_{i=1}^p$ with $b_i \in \mathbb{R}^k$ are the rows of B and $\beta_i \overset{\text{def.}}{=} \|b_i\|^2 \geqslant 0$. One has

$$B^\top B = S^\top \tilde{V}\tilde{V}^\top S = S^\top S = \mathrm{Id}_k,$$

so that the columns of B are orthogonal, and thus

$$\sum_i \beta_i = \sum_i \|b_i\|^2 = \|B\|_{\mathrm{Fro}}^2 = \mathrm{tr}(B^\top B) = \mathrm{tr}(B^\top B) = k.$$

We extend the k columns of b into an orthogonal basis $\tilde{B} \in \mathbb{R}^{p\times p}$ such that $\tilde{B}\tilde{B}^\top = \tilde{B}^\top \tilde{=} \mathrm{Id}_p$, so that

$$0 \leqslant \beta_i = \|b_i\|^2 \leqslant \|\tilde{b}_i\|^2 = 1$$

and hence $(\beta_i)_{i=1}^p$ satisfies the constraint of the considered optimization problem, hence $\mathrm{tr}(S^\top C S)$ is necessarily smaller than the maximum possible value.

For the proof of the second upper bound, we only verify it in 2D and 3D using a drawing, see Figure 12.4, left. □

Proof of Theorem 20. Setting $S = V_{1:k} = [v_1, \ldots, v_k]$, it satisfies $CS = V\Lambda V^\top V_{1:k} = V_{1:k}\,\mathrm{diag}(\lambda_i)_{i=1}^k$ and hence

$$\mathrm{tr}(S^\top C S) = \mathrm{tr}(S^\top S\,\mathrm{diag}(\lambda_i)_{i=1^k}) = \mathrm{tr}(\mathrm{Id}_k\,\mathrm{diag}(\lambda_i)_{i=1^k}) = \sum_{i=1}^k \lambda_i.$$

This value matches the right-most upper bound of Lemma 9, which shows that this S is optimal. □

12.1.2 Clustering and k-means

A typical unsupervised learning task is to infer a class label $y_i \in \{1, \dots, k\}$ for each input point x_i, and this is often called a clustering problem (since the set of points associated to a given label can be thought as a cluster).

k-means A way to infer these labels is by assuming that the clusters are compact, and optimizing some compactness criterion. Assuming for simplicity that the data are in Euclidean space (which can be relaxed to an arbitrary metric space, although the computations become more complicated), the k-means approach minimizes the distance between the points and their class centroids $c = (c_\ell)_{\ell=1}^k$, where each $c_\ell \in \mathbb{R}^p$. The corresponding variational problem becomes

$$\min_{(y,c)} \mathcal{E}(y,c) \stackrel{\text{def.}}{=} \sum_{\ell=1}^{k} \sum_{i:y_i=\ell} \|x_i - c_\ell\|^2.$$

The k-means algorithm can be seen as a block coordinate relaxation, which alternatively updates the class labels and the centroids. The centroids c are first initialized (more on this later), for instance, using a well-spread set of points from the samples. For a given set c of centroids, minimizing $y \mapsto \mathcal{E}(y,c)$ is obtained in closed form by assigning as class label the index of the closest centroids

$$\forall\, i \in \{1, \dots, n\}, \quad y_i \leftarrow \operatorname*{argmin}_{1 \leqslant \ell \leqslant k} \|x_i - c_\ell\|. \tag{12.7}$$

For a given set y of labels, minimizing $c \mapsto \mathcal{E}(y,c)$ is obtained in closed form by computing the barycenter of each class

$$\forall\, \ell \in \{1, \dots, k\}, \quad c_\ell \leftarrow \frac{\sum_{i:y_i=\ell} x_i}{|\{i\ ;\ y_i = \ell\}|} \tag{12.8}$$

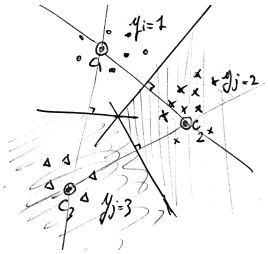


Figure 12.5: k-means clusters according to Vornoi cells.

If during the iterates, one of the cluster associated to some c_ℓ becomes empty, then one can either decide to destroy it and replace k by $k-1$, or try to "teleport" the center c_ℓ to another location (this might increase the objective function $\mathcal{E}$ however).

Since the energy $\mathcal{E}$ is decaying during each of these two steps, it is converging to some limit value. Since there is a finite number of possible labels assignments, it is actually constant after a finite number of iterations, and the algorithm stops.

Of course, since the energy is non-convex, little can be said about the property of the clusters output by k-means. To try to reach lower energy level, it is possible to "teleport" during the iterations centroids c_ℓ associated to clusters with high energy to locations within clusters with lower energy (because optimal solutions should somehow balance the energy).

Figure 12.6 shows an example of k-means iterations on the Iris dataset.

k-means++ To obtain good results when using k-means, it is crucial to have an efficient initialization scheme. In practice, the best results are obtained by seeding them as far as possible from one another (a greedy strategy works great in practice).

Quite surprisingly, there exists a randomized seeding strategy which can be shown to be close to optimal in term of value of $\mathcal{E}$, even without running the k-means iterations (although in practice it still needs to be used to polish the results). The corresponding k-means++ initialization is obtained by selecting c_1 uniformly at random among the x_i, and then assuming c_ℓ has been seeded, drawing $c_{\ell+1}$ among the sample according to the probability $\pi^{(\ell)}$ on $\{1, \dots, n\}$ proportional to the squared inverse of the distance to the previously seeded points

$$\forall\, i \in \{1, \dots, n\}, \quad \pi_i^{(\ell)} \stackrel{\text{def.}}{=} \frac{1/d_i^2}{\sum_{j=1}^n 1/d_j^2} \quad \text{where} \quad d_j \stackrel{\text{def.}}{=} \min_{1 \leqslant r \leqslant \ell-1} \|x_i - c_r\|.$$

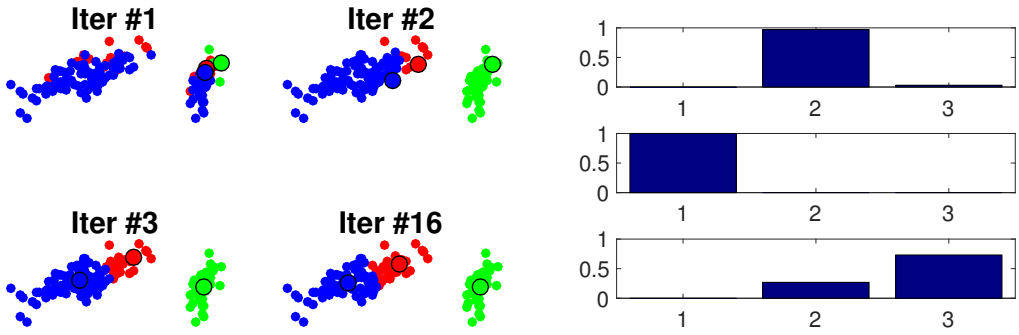


Figure 12.6: Left: iteration of k-means algorithm. Right: histogram of points belonging to each class after the k-means optimization.

This means that points which are located far away from the preciously seeded centers are more likely to be picked.

The following results, due to David Arthur and Sergei Vassilvitskii, shows that this seeding is optimal up to log factor on the energy. Note that finding a global optimum is known to be NP-hard.

Theorem 21. *For the centroids $c^\star$ defined by the k-means++ strategy, denoting $y^\star$ the associated nearest neighbor labels defined as in (12.7), one has*

$$\mathbb{E}(\mathcal{E}(y^\star, c^\star)) \leqslant 8(2+\log(k)) \min_{(y,c)} \mathcal{E}(y,v),$$

where the expectation is on the random draws performed by the algorithm.

Lloyd algorithm and continuous densities. The k-means iterations are also called "Lloyd" algorithm, which also find applications to optimal vector quantization for compression. It can also be used in the "continuous" setting where the empirical samples $(x_i)_i$ are replaced by an arbitrary measure over $\mathbb{R}^p$. The energy to minimize becomes

$$\min_{(\mathcal{V},c)} \sum_{\ell=1}^{k} \int_{\mathcal{V}_\ell} \|x-c_\ell\|^2 \mathrm{d}\mu(x)$$

where $(\mathcal{V}_\ell)_\ell$ is a partition of the domain. Step (12.7) is replaced by the computation of a Voronoi cell

$$\forall\, \ell \in \{1,\dots,k\}, \quad \mathcal{V}_\ell \overset{\text{def.}}{=} \{x \;;\; \forall\, \ell' \neq \ell, \|x-c_\ell\| \leqslant \|x-c_{\ell'}\|\}\,.$$

These Voronoi cells are polyhedra delimited by segments of mediatrix between centroids, and this Voronoi segmentation can be computed efficiently using tools from algorithmic geometry in low dimension. Step (12.8) are then replaced by

$$\forall\, \ell \in \{1,\dots,k\}, \quad c_\ell \leftarrow \frac{\int_{\mathcal{C}_\ell} x \mathrm{d}\mu(x)}{\int_{\mathcal{C}_\ell} \mathrm{d}\mu(x)}.$$

In the case of μ being uniform distribution, optimal solution corresponds to the hexagonal lattice. Figure 12.7 displays two examples of Lloyd iterations on 2-D densities on a square domain.

12.2 Empirical Risk Minimization

Before diving into the specifics of regression and classification problems, let us give describe a generic methodology which can be applied in both case (possibly with minor modification for classification, typically considering class probabilities instead of class labels).

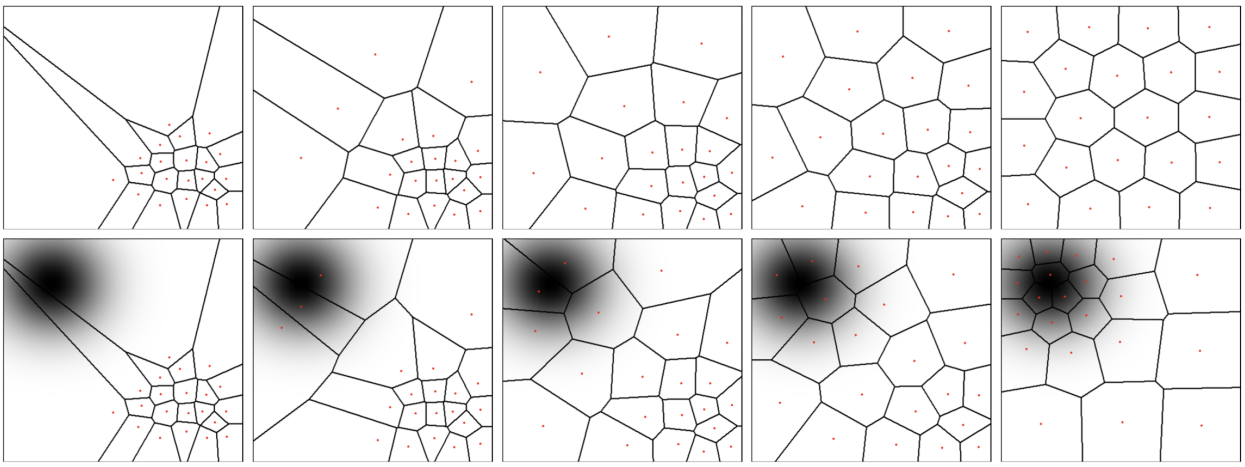


Figure 12.7: Iteration of k-means algorithm (Lloyd algorithm) on continuous densities μ. Top: uniform. Bottom: non-uniform (the densities of μ with respect to the Lebesgue measure is displayed as a grayscale image in the background).

In order to make the problem tractable computationally, and also in order to obtain efficient prediction scores, it is important to restrict the fit to the data $y_i \approx f(x_i)$ using a "small enough" class of functions. Intuitively, in order to avoid overfitting, the "size" of this class of functions should grows with the number n of samples.

12.2.1 Empirical Risk

Denoting $\mathcal{F}_n$ some class of functions (which depends on the number of available samples), one of the most usual way to do the learning is to perform an empirical risk minimization (ERM)

$$\hat{f} \in \operatorname*{argmin}_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i). \tag{12.9}$$

Here $L : \mathcal{Y}^2 \to \mathbb{R}^+$ is the so-called loss function, and it should typically satisfies $L(y, y') = 0$ if and only if $y = y'$. The specifics of L depend on the application at hand (in particular, one should use different losses for classification and regression tasks). To highlight the dependency of $\hat{f}$ on n, we occasionally write $\hat{f}_n$.

12.2.2 Prediction and Consistency

When doing a mathematically analysis, one usually assumes that (x_i, y_i) are drawn from a distribution π on $\mathcal{X} \times \mathcal{Y}$, and the large n limit defines the ideal estimator

$$\bar{f} \in \operatorname*{argmin}_{f \in \mathcal{F}_\infty} \int_{\mathcal{X} \times \mathcal{Y}} L(f(x), y) \mathrm{d}\pi(x, y) = \mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim \pi}(L(f(\mathbf{x}), \mathbf{y}). \tag{12.10}$$

Intuitively, one should have $\hat{f}_n \to \bar{f}$ as $n \to +\infty$, which can be captured in expectation of the prediction error over the samples $(x_i, y_i)_i$, i.e.

$$E_n \overset{\text{def.}}{=} \mathbb{E}(\tilde{L}(\hat{f}_n(\mathbf{x}), \bar{f}(\mathbf{x}))) \longrightarrow 0.$$

One should be careful that here the expectation is over both $\mathbf{x}$ (distributed according to the marginal $\pi_{\mathcal{X}}$ of π on $\mathcal{X}$), and also the n i.i.d. pairs $(x_i, y_i) \sim \pi$ used to define $\hat{f}_n$ (so a better notation should rather be

$(\mathbf{x}_i, \mathbf{y}_i)_i$. Here $\bar{L}$ is some loss function on $\mathcal{Y}$ (one can use $\bar{L} = L$ for instance). One can also study convergence in probability, i.e.

$$\forall\, \varepsilon > 0, \quad E_{\varepsilon,n} \overset{\text{def.}}{=} \mathbb{P}(\tilde{L}(\hat{f}_n(\mathbf{x}), \bar{f}(\mathbf{x})) > \varepsilon) \to 0.$$

If this holds, then one says that the estimation method is consistent (in expectation or in probability). The question is then to derive convergence rates, i.e. to upper bound E_n or $E_{\varepsilon,n}$ by some explicitly decay rate.

Note that when $\tilde{L}(y, y') = |y - y'|^r$, then convergence in expectation is stronger (implies) than convergence in probability since using Markov's inequality

$$E_{\varepsilon,n} = \mathbb{P}(|\hat{f}_n(\mathbf{x}) - f(\mathbf{x})|^r \geqslant \varepsilon) \leqslant \frac{1}{\varepsilon}\mathbb{E}(|\hat{f}_n(\mathbf{x}) - f(\mathbf{x})|^r) = \frac{E_n}{\varepsilon}.$$

12.2.3 Parametric Approaches and Regularization

Instead of directly defining the class $\mathcal{F}_n$ and using it as a constraint, it is possible to rather use a penalization using some prior to favor "simple" or "regular" functions. A typical way to achieve this is by using a parametric model $y \approx f(x, \beta)$ where $\beta \in \mathcal{B}$ parametrizes the function $f(\cdot, \beta) : \mathcal{X} \to \mathcal{Y}$. The empirical risk minimization procedure (12.9) now becomes

$$\hat{\beta} \in \operatorname*{argmin}_{\beta \in \mathcal{B}} \frac{1}{n}\sum_{i=1}^{n} L(f(x_i, \beta), y_i) + \lambda_n J(\beta). \tag{12.11}$$

where J is some regularization function, for instance $J = \|\cdot\|_2^2$ (to avoid blowing-up of the parameter) or $J = \|\cdot\|_1$ (to perform model selection, i.e. using only a sparse set of feature among a possibly very large pool of p features). Here $\lambda_n > 0$ is a regularization parameter, and it should tend to 0 when $n \to +\infty$.

Then one similarly defines the ideal parameter $\bar{\beta}$ as in (12.10) so that the limiting estimator as $n \to +\infty$ is of the form $\bar{f} = f(\cdot, \bar{\beta})$ for $\bar{\beta}$ defined as

$$\bar{\beta} \in \operatorname*{argmin}_{\beta} \int_{\mathcal{X}\times\mathcal{Y}} L(f(x, \beta), y)\mathrm{d}\pi(x, y) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\pi}(L(f(\mathbf{x}, \beta), \mathbf{y}). \tag{12.12}$$

Prediction vs. estimation risks. In this parametric approach, one could be interested in also studying how close $\hat{\theta}$ is to $\bar{\theta}$. This can be measured by controlling how fast some estimation error $\|\hat{\beta} - \bar{\beta}\|$ (for some norm $\|\cdot\|$) goes to zero. Note however that in most cases, controlling the estimation error is more difficult than doing the same for the prediction error. In general, doing a good parameter estimation implies doing a good prediction, but the converse is not true.

12.2.4 Testing Set and Cross-validation

It is not possible to access E_n or $E_{\varepsilon,n}$ because the optimal $\bar{f}$ is unknown. In order to tune some parameters of the methods (for instance the regularization parameter λ), one rather wants to minimize the risk $\mathbb{E}(L(\hat{f}(\mathbf{x}), \mathbf{y}))$, but this one should not be approximated using the training samples $(x_i, y_i)_i$.

One thus rather ressorts to a second set of data $(\bar{x}_j, \bar{y}_j)_{j=1}^{\bar{n}}$, called "testing set". From a modelling perspective, this set should also be distributed i.i.d. according to π. The validation (or testing) risk is then

$$R_{\bar{n}} = \frac{1}{\bar{n}}\sum_{j=1}^{\bar{n}} L(\hat{f}(\bar{x}_j), \bar{y}_j) \tag{12.13}$$

which converges to $\mathbb{E}(L(\hat{f}(\mathbf{x}), \mathbf{y}))$ for large $\bar{n}$. Minimizing $R_{\bar{n}}$ to setup to some meta-parameter of the method (for instance the regularization parameter λ_n) is called "cross validation" in the literature.

12.3 Supervised Learning: Regression

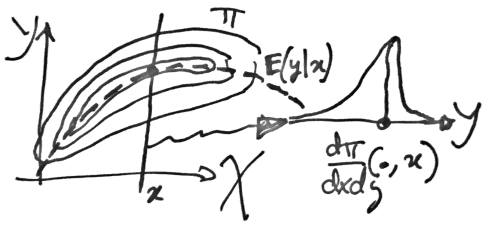


Figure 12.9: Conditional expectation.

In supervised learning, one has access to training data, consisting in pairs $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. Here $\mathcal{X} = \mathbb{R}^p$ for simplicity. The goal is to infer some relationship, typically of the form $y_i \approx f(x_i)$ for some deterministic function $f : \mathcal{X} \to \mathcal{Y}$, in order, when some un-observed data x without associated value in $\mathcal{Y}$ is given, to be able to "predict" the associated value using $y = f(x)$.

If the set $\mathcal{Y}$ is discrete and finite, then this problem is called a supervised classification problem, and this is studied in Section 12.4. The simplest example being the binary classification case, where $\mathcal{Y} = \{0, 1\}$. It finds applications for instance in medical diagnosis, where $y_i = 0$ indicates a healthy subject, why $y_i = 0$ a pathological one. If $\mathcal{Y}$ is continuous (the typical example being $\mathcal{Y} = \mathbb{R}$), then this problem is called a regression problem.

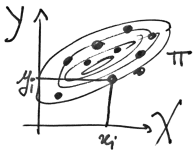


Figure 12.8: Probabilistic modelling.

12.3.1 Linear Regression

We now specialize the empirical risk minimization approach to regression problems, and even more specifically, we consider $\mathcal{Y} = \mathbb{R}$ and use a quadratic loss $L(y, y') = \frac{1}{2}|y - y'|^2$.

Note that non-linear regression can be achieved using approximation in dictionary (e.g. polynomial interpolation), and this is equivalent to using lifting to a higher dimensional space, and is also equivalent to kernelization technics studied in Section 12.5.

Least square and conditional expectation. If one do not put any constraint on f (beside being measurable), then the optimal limit estimator $\bar{f}(x)$ defined in (12.10) is simply averaging the values y sharing the same x, which is the so-called conditional expectation. Assuming for simplicity that π has some density $\frac{\mathrm{d}\pi}{\mathrm{d}x\mathrm{d}y}$ with respect to a tensor product measure $\mathrm{d}x\mathrm{d}y$ (for instance the Lebegues mesure), one has

$$\forall\, x \in \mathcal{X}, \quad \bar{f}(x) = \mathbb{E}(\mathbf{y}|\mathbf{x} = x) = \frac{\int_{\mathcal{Y}} y \frac{\mathrm{d}\pi}{\mathrm{d}x\mathrm{d}y}(x, y)\mathrm{d}y}{\int_{\mathcal{Y}} \frac{\mathrm{d}\pi}{\mathrm{d}x\mathrm{d}y}(x, y)\mathrm{d}y}$$

where $(\mathbf{x}, \mathbf{y})$ are distributed according to π.

In the simple case where $\mathcal{X}$ and $\mathcal{Y}$ are discrete, denoting $\pi_{x,y}$ the probability of $(\mathbf{x} = x, \mathbf{y} = y)$, one has

$$\forall\, x \in \mathcal{X}, \quad \bar{f}(x) = \frac{\sum_y y\pi_{x,y}}{\sum_y \pi_{x,y}}$$

and it is unspecified if the marginal of π along $\mathcal{X}$ vanishes at x.

The main issue is that this estimator $\hat{f}$ performs poorly on finite samples, and $f(x)$ is actually undefined if there is no sample x_i equal to x. This is due to the fact that the class of functions is too large, and one should impose some regularity or simplicity on the set of admissible f.

Penalized linear models. A very simple class of models is obtained by imposing that f is linear, and set $f(x,\beta) = \langle x,\, \beta\rangle$, for parameters $\beta \in \mathcal{B} = \mathbb{R}^p$. Note that one can also treat this way affine functions by remarking that $\langle x,\, \beta\rangle + \beta_0 = \langle (x,1),\, (\beta,\beta_0)\rangle$ and replacing x by $(x,1)$. So in the following, without loss of generality, we only treat the vectorial (non-affine) case.

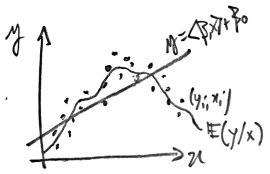


Figure 12.10: Linear regression.

Under the square loss, the regularized ERM (12.11) is conveniently rewritten as

$$\hat\beta \in \operatorname*{argmin}_{\beta\in\mathcal{B}} \frac{1}{2}\langle \hat C\beta,\, \beta\rangle - \langle \hat u,\, \beta\rangle + \lambda_n J(\beta) \tag{12.14}$$

where we introduced the empirical correlation (already introduced in (12.1)) and observations

$$\hat C \overset{\text{def.}}{=} \frac{1}{n}X^*X = \frac{1}{n}\sum_{i=1}^n x_i x_i^* \quad\text{and}\quad \hat u \overset{\text{def.}}{=} \frac{1}{n}\sum_{i=1}^n y_i x_i = \frac{1}{n}X^*y \in \mathbb{R}^p.$$

As $n\to 0$, under weak condition on π, one has with the law of large numbers the almost sure convergence

$$\hat C \to C \overset{\text{def.}}{=} \mathbb{E}(\mathbf{x}^*\mathbf{x}) \quad\text{and}\quad \hat u \to u \overset{\text{def.}}{=} \mathbb{E}(\mathbf{y}\mathbf{x}). \tag{12.15}$$

When considering $\lambda_n \to 0$, in some cases, one can shows that in the limit $n\to+\infty$, one retrieves the following ideal parameter

$$\bar\beta \in \operatorname*{argmin}_{\beta} \left\{ J(\beta)\;;\; C\beta = u \right\}.$$

Problem (12.14) is equivalent to the regularized resolution of inverse problems (8.9), with $\hat C$ in place of Φ and $\hat u$ in place of $\Phi^* y$. The major, and in fact only difference between machine learning and inverse problems is that the linear operator is also noisy since $\hat C$ can be viewed as a noisy version of C. The "noise level", in this setting, is $1/\sqrt{n}$ in the sense that

$$\mathbb{E}(\|\hat C - C\|) \sim \frac{1}{\sqrt{n}} \quad\text{and}\quad \mathbb{E}(\|\hat u - u\|) \sim \frac{1}{\sqrt{n}},$$

under the assumption that $\mathbb{E}(\mathbf{y}^4) < +\infty$, $\mathbb{E}(\|\mathbf{x}\|^4) < +\infty$ so ensure that one can use the central limit theorem on $\mathbf{x}^2$ and $\mathbf{xy}$. Note that, although we use here linear estimator, one does not need to assume a "linear" relation of the form $\mathbf{y} = \langle \mathbf{x},\, \beta\rangle + w$ with a noise w independent from $\mathbf{x}$, but rather hope to do "as best as possible", i.e. estimate a linear model as close as possible to $\bar\beta$.

The general take home message is that it is possible to generalize Theorems 10, 12 and 13 to cope with the noise on the covariance matrix to obtain prediction convergence rates of the form

$$\mathbb{E}(|\langle \hat\beta,\, \mathbf{x}\rangle - \langle \bar\beta,\, \mathbf{x}\rangle|^2) = O(n^{-\kappa})$$

and estimation rates of the form

$$\mathbb{E}(\|\hat\beta - \bar\beta\|^2) = O(n^{-\kappa'}),$$

under some suitable source condition involving C and u. Since the noise level is roughly $n^{-\frac{1}{2}}$, the ideal cases are when $\kappa = \kappa' = 1$, which is the so-called linear rate regime. It is also possible to derive sparsistency theorems by extending theorem 14. For the sake of simplicity, we now focus our attention to quadratic penalization, which is by far the most popular regression technic. It is fair to say that sparse (e.g. ℓ^1 type) methods are not routinely used in machine learning, because they typically do not improve the estimation performances, and are mostly useful to do model selection (isolate a few useful coordinates in the features). This is in sharp contrast with the situation for inverse problems in imaging sciences, where sparsity is a key feature because it corresponds to a modelling assumption on the structure of the data to recover.

Ridge regression (quadratic penalization). For $J = \|\cdot\|^2/2$, the estimator (12.14) is obtained in closed form as

$$\hat{\beta} = (X^*X + n\lambda_n \mathrm{Id}_p)^{-1} X^* y = (\hat{C} + n\lambda_n \mathrm{Id})^{-1} \hat{u}. \tag{12.16}$$

This is often called ridge regression in the literature. Note that thanks to the Woodbury formula, this estimator can also be re-written as

$$\hat{\beta} = X^*(XX^* + n\lambda_n \mathrm{Id}_n)^{-1} y. \tag{12.17}$$

If $n \gg p$ (which is the usual setup in machine learning), then (12.17) is preferable. In some cases however (in particular when using RKHS technics), it makes sense to consider very large p (even infinite dimensional), so that (12.16) must be used.

If $\lambda_n \to 0$, then using (12.15), one has the convergence in expectation and probability

$$\hat{\beta} \to \bar{\beta} = C^+ u.$$

Theorems 10 and 12 can be extended to this setting and one obtains the following result.

Theorem 22. *If*

$$\bar{\beta} = C^\gamma z \quad \textit{where} \quad \|z\| \leqslant \rho \tag{12.18}$$

for $0 < \gamma \leqslant 2$, *then*

$$\mathbb{E}(\|\hat{\beta} - \bar{\beta}\|^2) \leqslant C\rho^{2\frac{1}{\gamma+1}} n^{-\frac{\gamma}{\gamma+1}} \tag{12.19}$$

for a constant C *depending only on* γ.

It is important to note that, since $\bar{\beta} = C^+ u$, the source condition (12.18) is always satisfied. What trully matters here is that the rate (12.19) does not depend on the dimension p of the features, but rather only on ρ, which can be much smaller. This theoretical analysis actually works perfectly fine in infinite dimension $p = \infty$ (which is the setup considered when dealing with RKHS below).

12.4 Supervised Learning: Classification

We now focus on the case of discrete labels $y_i \in \mathcal{Y} = \{1, \ldots, k\}$, which is the classification setup. We now detail two popular classification methods: nearest neighbors and logistic classification. It is faire to say that a significant part of successful applications of machine learning technics consists in using one of these two approaches, which should be considered as baselines. Note that the nearest neighbors approach, while popular for classification could as well be used for regression.

12.4.1 Nearest Neighbors Classification

Probably the simplest method for supervised classification is R nearest neighbors (R-NN), where R is a parameter indexing the number of neighbors. Increasing R is important to cope with noise and obtain smoother decision boundaries, and hence better generalization performances. It should typically decreases as the number of training samples n increases. Despite its simplicity, k-NN is surprisingly successful in practice, specially in low dimension p.

The class $\hat{f}(x) \in \mathcal{Y}$ 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. This is a non-parametric method, and $\hat{f}$ depends on the numbers n of samples (its "complexity" increases with n).

One first compute the Euclidean distance between this x and all other x_i in the training set. Sorting the distances generates an indexing σ (a permutation of $\{1, \ldots, n\}$) such that

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

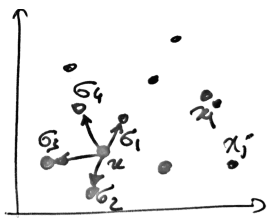
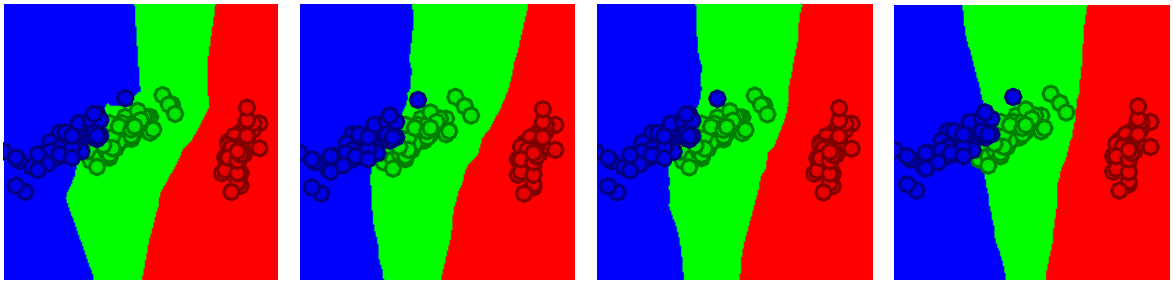


Figure 12.11: Nearest neighbors.



$k = 1$ $k = 5$ $k = 10$ $k = 40$

Figure 12.12: k-nearest-neighbor classification boundary function.

For a given R, one can compute the "local" histogram of classes around x

$$h_\ell(x) \stackrel{\text{def.}}{=} \frac{1}{R} \big\{ i \; ; \; y_{\sigma(i)} \in \{1, \ldots, R\} \big\}.$$

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

$$\hat{f}(x) \in \operatorname*{argmax}_\ell h_\ell(x).$$

In practice, the parameter R can be setup through cross-validation, by minimizing the testing risk $R_{\bar{n}}$ defined in (12.13), which typically uses a 0-1 loss for counting the number of mis-classifications

$$R_{\bar{n}} \stackrel{\text{def.}}{=} \sum_{j=1}^{\bar{n}} \delta(\bar{y}_j - \hat{f}(x_i))$$

where $\delta(0) = 0$ and $\delta(s) = 1$ if $s \neq 0$. Of course the method extends to arbitrary metric space in place of Euclidean space $\mathbb{R}^p$ for the features. Note also that instead of explicitly sorting all the Euclidean distance, one can use fast nearest neighbor search methods.

Figure 12.12 shows, for the IRIS dataset, the classification domains (i.e. $\{x \; ; \; f(x) = \ell\}$ for $\ell = 1, \ldots, k$) using a 2-D projection for vizualization. Increasing R leads to smoother class boundaries.

12.4.2 Two Classes Logistic Classification

The logistic classification method (for 2 classes and multi-classes) is one of the most popular (maybe "the" most) popular machine learning technics. This is due in large part of both its simplicity and because it also outputs a probability of belonging to each class (in place of just a class membership), which is useful to (somehow ...) quantify the "uncertainty" of the estimation. Note that logistic classification is actually called "logistic regression" in the literature, but it is in fact a classification method.

Another very popular (and very similar) approach is support vector machine (SVM). SVM is both more difficult to train (because the loss is non-smooth) and does not give class membership probability, so the general rule of thumb is that logistic classification is preferable.

To simplify the expression, classes indexes are set to $y_i \in \mathcal{Y} = \{-1, 1\}$ in the following. Note that for logistic classification, the prediction function $f(\cdot, \beta) \in [0, 1]$ outputs the probability of belonging to the first class, and not the class indexes. With a slight abuse of notation, we still denote it as f.

Approximate risk minimization. The hard classifier is defines from a linear predictor $\langle x, \beta \rangle$ as $\operatorname{sign}(\langle x, \beta \rangle) \in \{-1, +1\}$. The 0-1 loss error function (somehow the "ideal" loss) counts the number of miss-classifications, and can ideal classifier be computed as

$$\min_\beta \sum_{i=1}^{n} \ell_0(-y_i \langle x_i, \beta \rangle) \tag{12.20}$$

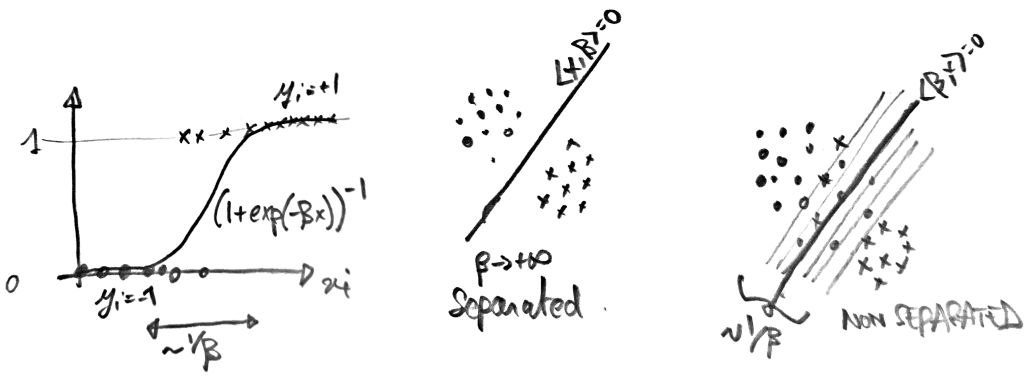


Figure 12.13: 1-D and 2-D logistic classification, showing the impact of $\|\beta\|$ on the sharpness of the classification boundary.

where $\ell_0 = 1_{\mathbb{R}^+}$. Indeed, miss classification corresponds to $\langle x_i, w\rangle$ and y_i having different signs, so that in this case $\ell_0(-y_i\langle x_i, w\rangle) = 1$ (and 0 otherwise for correct classification).

The function ℓ_0 is non-convex and hence problem (12.20) is itself non-convex, and in full generality, can be shown to be NP-hard to solve. One thus relies on some proxy, which are functions which upper-bounds ℓ_0 and are convex (and sometimes differentiable).

The most celebrated proxy are

$$\ell(u) = (1+u)_+ \quad \text{and} \quad \ell(u) = \log(1+\exp(u))/\log(2)$$

which are respectively the hinge loss corresponds to support vector machine (SVM, and is non-smooth) and the logistic loss (which is smooth). The $1/\log(2)$ is just a constant which makes $\ell_0 \leqslant \ell$. AdaBoost is using $\ell(u) = e^u$. Note that least square corresponds to using $\ell(u) = (1+u)^2$, but this is a poor proxy for ℓ_0 for negative values, although it might works well in practice. Note that SVM is a non-smooth problem, which can be cast as a linear program minimizing the co-called classification margin

$$\min_{u\geqslant 0,\beta} \left\{ \sum_i u_i \;;\; 1+u_i = y_i\langle x_i, \beta\rangle \right\}.$$

Logistic loss probabilistic interpretation. Logistic classification can be understood as a linear model as introduced in Section 12.3.1, although the decision function $f(\cdot,\beta)$ is not linear. Indeed, one needs to "remap" the linear value $\langle x, \beta\rangle$ in the interval $[0,1]$. In logistic classification, we define the predicted probability of x belonging to class with label -1 as

$$f(x,\beta) \overset{\text{def.}}{=} \theta(\langle x, \beta\rangle) \quad \text{where} \quad \theta(s) \overset{\text{def.}}{=} \frac{e^s}{1+e^s} = (1+e^{-s})^{-1}, \tag{12.21}$$

which is often called the "logit" model. Using a linear decision model might seems overly simplistic, but in high dimension p, the number of degrees of freedom is actually enough to reach surprisingly good classification performances. Note that the probability of belonging to the second class is $1 - f(x,\beta) = \theta(-s)$. This symmetry of the θ function is important because it means that both classes are treated equally, which makes sense for "balanced" problem (where the total mass of each class are roughly equal).

Intuitively, $\beta/\|\beta\|$ controls the separating hyperplane direction, while $1/\|\beta\|$ is roughly the fuzziness of the the separation. As $\|\beta\| \to +\infty$, one obtains sharp devision boundary, and logistic classification ressembles SVM.

Note that $f(x,\beta)$ can be interpreted as a single layer perceptron with a logistic (sigmoid) rectifying unit, more details on this in Chapter 15.

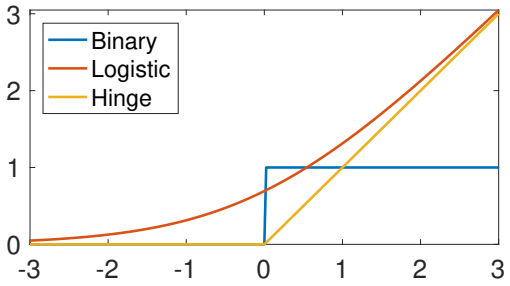


Figure 12.14: Comparison of loss functions. **[ToDo: Re-do the figure, it is not correct, they should upper bound ℓ_0]**

Since the (x_i, y_i) are modeled as i.i.d. variables, it makes sense to define $\hat{\beta}$ from the observation using a maximum likelihood, assuming that each y_i conditioned on x_i is a Bernoulli variable with associated probability $(p_i, 1-p_i)$ with $p_i = f(x_i, \beta)$. The probability of observing $y_i \in \{0,1\}$ is thus, denoting $s_i = \langle x_i, \beta \rangle$

$$\mathbb{P}(\mathbf{y} = y_i | \mathbf{x} = x_i) = p_i^{1-\bar{y}_i}(1-p_i)^{\bar{y}_i} = \left(\frac{e^{s_i}}{1+e^{s_i}}\right)^{1-\bar{y}_i}\left(\frac{1}{1+e^{s_i}}\right)^{\bar{y}_i}$$

where we denoted $\bar{y}_i = \frac{y_i+1}{2} \in \{0,1\}$.

One can then minimize minus the sum of the log of the likelihoods, which reads

$$\hat{\beta} \in \operatorname*{argmin}_{\beta \in \mathbb{R}^p} -\sum_{i=1}^n \log(\mathbb{P}(\mathbf{y} = y_i | \mathbf{x} = x_i)) = \sum_{i=1}^n -(1-\bar{y}_i)\log\frac{e^{s_i}}{1+e^{s_i}} - \bar{y}_i \log\frac{1}{1+e^{s_i}}$$

Some algebraic manipulations shows that this is equivalent to an ERM-type form (12.11) with a logistic loss function

$$\hat{\beta} \in \operatorname*{argmin}_{\beta \in \mathbb{R}^p} E(\beta) = \frac{1}{n}\sum_{i=1}^n L(\langle x_i, \beta \rangle, y_i) \tag{12.22}$$

where the logistic loss reads

$$L(s,y) \overset{\text{def.}}{=} \log(1+\exp(-sy)). \tag{12.23}$$

Problem (12.22) is a smooth convex minimization. If X is injective, E is also strictly convex, hence it has a single global minimum.

Figure (12.14) compares the binary (ideal) 0-1 loss, the logistic loss and the hinge loss (the one used for SVM).

Gradient descent method. Re-writing the energy to minimize

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

its gradient reads

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

where $\odot$ is the pointwise multiplication operator, i.e. `.*` in Matlab. Once $\beta^{(\ell=0)} \in \mathbb{R}^p$ is initialized (for instance at 0_p), one step of gradient descent (17.2) reads

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

To understand the behavior of the method, in Figure 12.15 we generate synthetic data distributed according to a mixture of Gaussian with an overlap governed by an offset ω. One can display the data overlaid on top of the classification probability, this highlight the separating hyperplane $\{x \; ; \; \langle \beta, x \rangle = 0\}$.

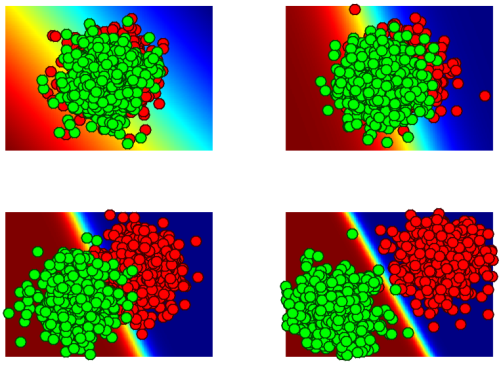


Figure 12.15: Influence on the separation distance between the class on the classification probability.

12.4.3 Multi-Classes Logistic Classification

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

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

$$f(x, \beta) = \left(\frac{e^{-\langle x, \beta_\ell \rangle}}{\sum_m e^{-\langle x, \beta_m \rangle}} \right)_\ell$$

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 β is obtained by solving a maximum likelihood estimator

$$\max_{\beta \in \mathbb{R}^{p \times k}} \frac{1}{n} \sum_{i=1}^n \log(f(x_i, \beta)_{y_i})$$

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

This is conveniently rewritten as

$$\min_{\beta \in \mathbb{R}^{p \times k}} \mathcal{E}(\beta) \stackrel{\text{def.}}{=} \sum_i \mathrm{LSE}(X\beta)_i - \langle X\beta, D \rangle$$

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

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

and LSE is the log-sum-exp operator

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

Note that in the case of $k = 2$ classes $\mathcal{Y} = \{-1, 1\}$, this model can be shown to be equivalent to the two-classes logistic classifications methods exposed in Section (12.4.2), with a solution vector being equal to $\beta_1 - \beta_2$ (so it is computationally more efficient to only consider a single vector as we did).

The computation of the LSE operator is unstable for large value of $S_{i,\ell}$ (numerical overflow, producing NaN), but this can be fixed by subtracting the largest element in each row, since

$$\mathrm{LSE}(S + a) = \mathrm{LSE}(S) + a$$

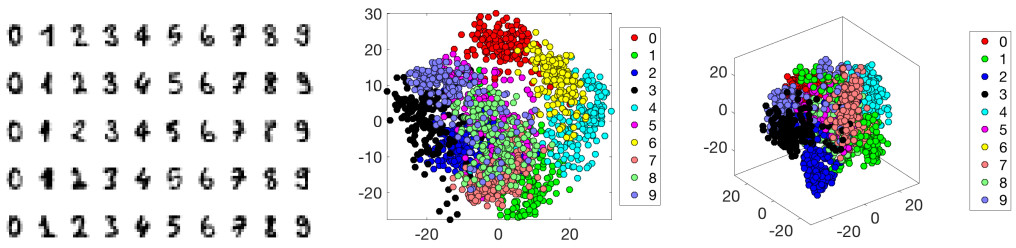


Figure 12.16: 2-D and 3-D PCA vizualization of the digits images.

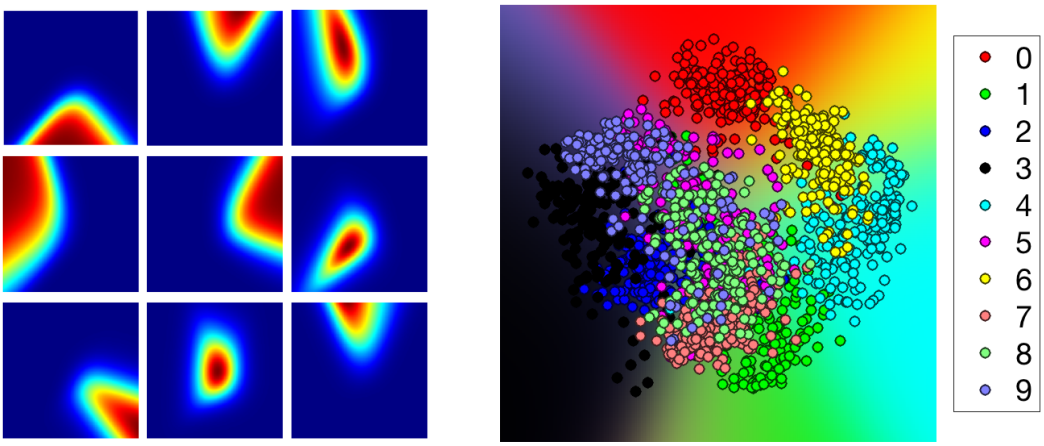


Figure 12.17: Results of digit classification Left: probability $h(x)_\ell$ of belonging to each of the 9 first classes (displayed over a 2-D PCA space). Right: colors reflect probability $h(x)$ of belonging to classes.

if a is constant along the rows. This is often referred to as the "LSE trick" and is very important to use in practice (in particular if some classes are well separated, since the corresponding β_ℓ vector might become large).

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

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

Similarly to the LSE, it needs to be stabilized by subtracting the maximum value along rows before computation.

Once D matrix is computed, the gradient of $\mathcal{E}$ is computed as

$$\nabla \mathcal{E}(\beta) = \frac{1}{n} X^* (\mathrm{SM}(X\beta) - D).$$

and one can minimize $\mathcal{E}$ using for instance a gradient descent scheme.

To illustrate the method, we use a dataset of n images of size $p = 8 \times 8$, representing digits from 0 to 9 (so there are $k = 10$ classes). Figure 12.16 displays a few representative examples as well as 2-D and 3-D PCA projections. Figure (12.17) displays the "fuzzy" decision boundaries by vizualizing the value of $h(x)$ using colors on an image regular grid.

12.5 Kernel Methods

Linear methods are parametric and cannot generate complex regression or decision functions. The linearity assumption is often too restrictive and in some case the geometry of the input functions or classes is

not well capture by these models. In many cases (e.g. for text data) the input data is not even in a linear space, so one cannot even apply these model.

Kernel method is a simple yet surprisingly powerful remedy for these issues. By lifting the features to a high dimensional embedding space, it allows to generate non-linear decision and regression functions, but still re-use the machinery (linear system solvers or convex optimization algorithm) of linear models. Also, by the use of the so-called "kernel-trick", the computation cost does not depends on the dimension of the embedding space, but of the number n of points. It is the perfect example of so-called "non-parametric" methods, where the number of degrees of freedom (number of variables involved when fitting the model) grows with the number of samples. This is often desirable when one wants the precisions of the result to improve with n, and also to mathematically model the data using "continuous" models (e.g. functional spaces such as Sobolev).

The general rule of thumb is that any machine learning algorithm which only makes use of inner products (and not directly of the features x_i themselves) can be "kernelized" to obtain a non-parametric algorithm. This is for instance the case for linear and nearest neighbor regression, SVM classification, logistic classification and PCA dimensionality reduction. We first explain the general machinery, and instantiate this in two representative setup (ridge regression, nearest-neighbor regression and logistic classification)

12.5.1 Reproducing Kernel Hilbert Space

We consider a general lifting $\varphi : x \in \mathbb{R}^p \to \bar{x} = \varphi(x) \in \mathcal{H}$ where $\mathcal{H}$ is a Hilbert space. A typical example of lift for $1-D$ values $p = 1$ is $\varphi(x) = (1, x, x^2, \ldots, x^k) \in \mathbb{R}^k$ to perform polynomial regression (this can be extended to any dimension p using higher dimensional polynomials). We denote $\bar{X} = (\bar{x}_i^* \overset{\text{def.}}{=} \varphi(x_i)^*)_{i=1}^n$ the "matrix" where each row is a lifted feature $\varphi(x_i)$. For instance, if $\mathcal{H} = \mathbb{R}^{\bar{p}}$ is finite dimensional, one can view this as a matrix $\bar{X} \in \mathbb{R}^{n \times \bar{p}}$, but the rows of the matrix can be infinite dimensional vectors.

The following proposition is the crux of the RKHS approaches. When using a regularization which is a squared Euclidean norm, $\|\cdot\|_{\mathcal{H}}^2$, it states that the solutions actually belongs to a data-driven linear sub-space of dimension n. Although the proof is straightforward, its implications are very profound, since it leads to tractable algorithms even when using an infinite dimensional lifting space $\mathcal{H}$. as we elaborate next. It is often called the "representer" theorem in RKHS theory.

Proposition 37. *The solution $\beta^\star \in \mathcal{H}$ of*

$$\min_{\beta \in \mathcal{H}} \mathcal{L}(\bar{X}\beta, y) + \frac{\lambda}{2}\|\beta\|_{\mathcal{H}}^2 \tag{12.24}$$

is unique and can be written as

$$\beta = \bar{X}^* q^\star = \sum_i q_i^\star \varphi(x_i) \in \mathcal{H} \tag{12.25}$$

where $q \in \mathbb{R}^N$ is a solution of

$$\min_{q \in \mathbb{R}^N} \mathcal{L}(Kp, y) + \frac{\lambda}{2}\langle Kq,\, q\rangle_{\mathbb{R}^n} \tag{12.26}$$

where we defined

$$K \overset{\text{def.}}{=} \bar{X}^*\bar{X} = (\langle \varphi(x_i),\, \varphi(x_j)\rangle_{\mathcal{H}})_{i,j=1}^n \in \mathbb{R}^{n \times n}.$$

Proof. The first order condition of (12.24) reads

$$0 \in \bar{X}^* \partial \mathcal{L}(\bar{X}^*\beta^\star, y) + \lambda\beta^\star = 0$$

i.e. there exists $u^\star \in \partial\mathcal{L}(\bar{X}^*\beta^\star, y)$ such that

$$\beta^\star = -\frac{1}{\lambda}\bar{X}^* u^\star \in \mathrm{Im}(\bar{X}^*)$$

which is the desired result. □

Equation (12.25) expresses the fact that the solution only lives in the n dimensional space spanned by the lifted observed points $\varphi(x_i)$. A crucial by product of this results is that all the computations as well as the prediction procedure can be expressed using the so-called kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ associated to φ

$$\forall\, (x, x') \in \mathcal{X}^2, \quad \kappa(x, x') \overset{\text{def.}}{=} \langle \varphi(x'), \varphi(x') \rangle_{\mathcal{H}}.$$

Indeed, one has $K = (\kappa(x_i, x_j))_{i,j}$ and the prediction operator, as a function of x and not $\varphi(x)$ (which makes it non-linear) is a weighted sum of kernel functions centered at the x_i

$$\langle \bar{x},\, \beta^\star \rangle_{\mathcal{H}} = \sum_{i=1}^{n} p_i^\star \langle \varphi(x),\, \varphi(x_i) \rangle_{\mathcal{H}} = \sum_{i=1}^{n} p_i^\star \kappa(x_i, x). \tag{12.27}$$

This means that one actually never needs to manipulate quantities in $\mathcal{H}$ (which can be infinite dimensional).

But more importantly, one can reverse the process, and instead of starting from a lifting φ, directly consider a kernel $\kappa(x, x')$. This is actually the way this is done in practice, since it is easier to design kernel and think in term of their geometrical properties (for instance, one can sum kernels). In order for this to make sense, the kernel needs to be positive definite, i.e. one should have that $(\kappa(x_i, x_j))_{i,j}$ should be symmetric positive definite for any choice of sampling points $(x_i)_i$. This can be shown to be equivalent to the existence of a lifting function φ generating the kernel. Note that such a kernel can be defined on arbitrary space (not necessarily Euclidean).

When using the linear kernel $\kappa(x, y) = \langle x,\, y \rangle$, one retrieves the linear models studied in the previous section, and the lifting is trivial $\varphi(x) = x$. A family of popular kernels are polynomial ones, $\kappa(x, x') = (\langle x,\, y \rangle + c)^a$ for $a \in \mathbb{N}^*$ and $c > 0$, which corresponds to a lifting in finite dimension. For instance, for $a = 2$ and $p = 2$, one has a lifting in dimension 6

$$\kappa(x, x') = (x_1 x_1' + x_1 x_1' + c)^2 = \langle \varphi(x),\, \varphi(x') \rangle \quad \text{where} \quad \varphi(x) = (x_1^2, x_2^2, \sqrt{2} x_1 x_2, \sqrt{2c} x_1, \sqrt{2c} x_2, c)^* \in \mathbb{R}^6.$$

In Euclidean spaces, the gaussian kernel is the most well known and used kernel

$$\kappa(x, y) \overset{\text{def.}}{=} e^{-\frac{\|x-y\|^2}{2\sigma^2}}. \tag{12.28}$$

The bandwidth parameter $\sigma > 0$ is crucial and controls the locality of the model. It is typically tuned through cross validation. It corresponds to an infinite dimensional lifting $x \mapsto e^{-\frac{\|x-\cdot\|^2}{2(\sigma/2)^2}} \in L^2(\mathbb{R}^p)$. Another related popular kernel is the Laplacian kernel $\exp(-\|x-y\|/\sigma)$. More generally, when considering translation invariant kernels $\kappa(x, x') = k(x - x')$ on $\mathbb{R}^p$, being positive definite is equivalent to $\hat{k}(\omega) > 0$ where $\hat{k}$ is the Fourier transform, and the associated lifting is obtained by considering $\hat{h} = \sqrt{\hat{\kappa}}$ and $\varphi(x) = h(x - \cdot) \in L^2(\mathbb{R}^p)$.

12.5.2 Examples of Kernelized Algorithms

We illustrate this general machinery by applying it to three typical problems.

Kernelized ridge regression. The simplest instantiation of this kernelization approach is when using the square loss $L(y, y') = \frac{1}{2}|y - y'|^2$, which is the ridge regression problem studied in Section 12.3.1. The obtain regression model (12.27) corresponds to approximating the data using a weighted sum of data-centered kernel function $\kappa(x_i, \cdot)$. When using a Gaussian kernel (12.28), the bandwidth σ controls the smoothness of the approximation. This is illustrated in Figure 12.18.

In this special case of a square loss, one can solve in closed form (12.26) by solving a $n \times n$ linear system

$$q^\star = (KK + \lambda K)^{-1} K y = (K + \lambda \mathrm{Id}_N)^{-1} y$$

This expression matches exactly (12.17) when using K in place of $\hat{C}$

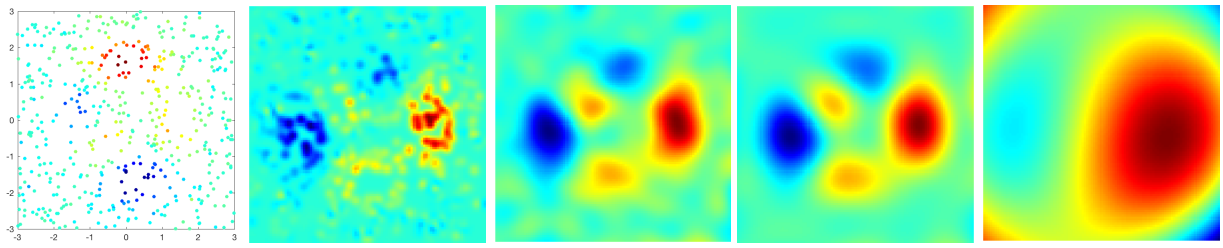


Figure 12.18: Regression using a Gaussian kernel.

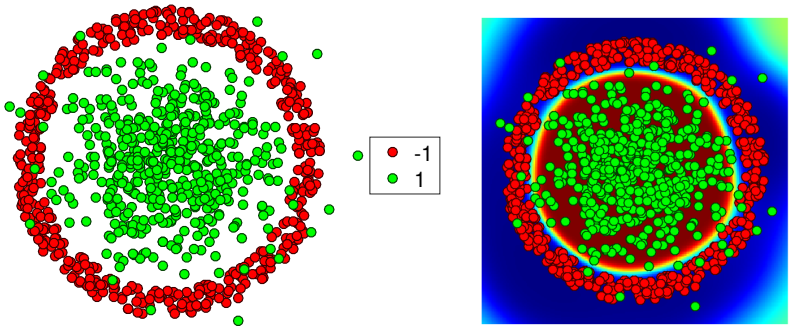


Figure 12.19: Non-linear classification using a Gaussian kernel.

Kernelized logistic classification. Logistic classification tries to separate the classes using a linear separating hyperplane $\{x\ ;\ \langle \beta,\ x\rangle = 0\}$. In order to generate a non-linear decision boundary, one can replace the parametric linear model by a non-linear non-parametric model, thanks to kernelization. This allows in particular to generate decision boundaries of arbitrary complexity.

In the two class problem, as detailed in Section 12.4.2, one solves (12.26) using the logistic loss (12.23). This can be for instance achieved by a gradient descent method. Once the solution $q^\star$ is obtained, the probability of x belonging to the first class is then

$$\theta(\sum_{i=1}^{n} q_i^\star \kappa(x_i, x)).$$

Figure 12.19 illustrate such a non-linear decision function on a simple 2-D problem.

Kernelized nearest-neihbors. It is also possible to extend nearest neighbor classification (as detailed in Section 12.4.1) and regression over a lifted space by making use only of kernel evaluation, simply noticing that

$$\|\varphi(x_i) - \varphi(x_j)\|_{\mathcal{H}}^2 = \kappa(x_i, x_i) + \kappa(x_j, x_j) - 2\kappa(x_i, x_j).$$

Kernel on strings. **[ToDo: write me]**

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. Candès and D. Donoho. New tight frames of curvelets and optimal representations of objects with piecewise C^2 singularities. *Commun. on Pure and Appl. Math.*, 57(2):219–266, 2004.

[5] E. J. Candès, L. Demanet, D. L. Donoho, and L. Ying. Fast discrete curvelet transforms. *SIAM Multiscale Modeling and Simulation*, 5:861–899, 2005.

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

[7] 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.

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

[9] S.S. Chen, D.L. Donoho, and M.A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1999.

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

[11] P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *SIAM Multiscale Modeling and Simulation*, 4(4), 2005.

[12] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Commun. on Pure and Appl. Math.*, 57:1413–1541, 2004.

[13] D. Donoho and I. Johnstone. Ideal spatial adaptation via wavelet shrinkage. *Biometrika*, 81:425–455, Dec 1994.

[14] Heinz Werner Engl, Martin Hanke, and Andreas Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.

[15] M. Figueiredo and R. Nowak. An EM Algorithm for Wavelet-Based Image Restoration. *IEEE Trans. Image Proc.*, 12(8):906–916, 2003.

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

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

[18] D. Mumford and J. Shah. Optimal approximation by piecewise smooth functions and associated variational problems. *Commun. on Pure and Appl. Math.*, 42:577–685, 1989.

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

[20] Gabriel Peyré. *L'algèbre discrète de la transformée de Fourier*. Ellipses, 2004.

[21] J. Portilla, V. Strela, M.J. Wainwright, and Simoncelli E.P. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Trans. Image Proc.*, 12(11):1338–1351, November 2003.

[22] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1-4):259–268, 1992.

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

[24] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[25] Jean-Luc Starck, Fionn Murtagh, and Jalal Fadili. *Sparse image and signal processing: Wavelets and related geometric multiscale analysis*. Cambridge university press, 2015.