Chapter X: Application to Data Analysis

In this chapter, we will see that many tasks in data analysis reduce to solving a convex optimization problem. Throughout we assume that we have observed some data $y_1, \ldots, y_m \in \mathbb{R}$. Each observation y_i is associated with a vector $\boldsymbol{x}_i \in \mathbb{R}^n$ of features: it is assumed that $\boldsymbol{x}_i = [x_{i1}, \ldots, x_{in}]$ are quantitative variables that carry some information about the observation y_i, and we search a model to *explain* the relation between the $\boldsymbol{x}_i$'s and y_i's. These models are typically used for

- *understanding* how the features explain the observed data;
- predict the observation y_* associated with some unseen feature $\boldsymbol{x}_*$.

In the machine learning terminology, the set of samples $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, m$, is called the *training set*: this is the data we use to *train* the model. Then, the set of new samples (e.g., $(\boldsymbol{x}_*, y_*)$) for which we use the model is called the *testing set* (of course, the algorithm only uses $\boldsymbol{x}_*$ to make predictions, but we can *test* the performance of an algorithm by comparing the predicted value $\hat{y_*}$ to the true value y_*).

1 Linear Regression

One of the simplest and most fundamental models is the linear regression model (and its many variants). Here, we assume that the relation between the observations and the features is approximately linear, i.e., there exists a vector $\boldsymbol{\theta} \in \mathbb{R}^n$ such that:

$$y_i \simeq \boldsymbol{x}_i^T \boldsymbol{\theta}.$$

> **Note:** It is straightforward to extend this model to deal with nonlinear relationships of the form $y_i \simeq \varphi(\boldsymbol{x}_i)^T \boldsymbol{\theta}$, where φ is some function of $\mathbb{R}^n \to \mathbb{R}^{n'}$. For example, if an affine relationship between y_i and the x_{ik}'s is sought, i.e., $y_i \simeq \theta_0 + \boldsymbol{\theta}^T \boldsymbol{x}_i$, we simply need to add a constant feature $x_{i0} = 1$ for each sample: indeed we have $\theta_0 + \boldsymbol{\theta}^T \boldsymbol{x}_i = \boldsymbol{x}_i'^T \boldsymbol{\theta}'$, where $\boldsymbol{x}_i' = \varphi(\boldsymbol{x}_i) := [1, \boldsymbol{x}_i^T]^T$ and $\boldsymbol{\theta}' = [\theta_0, \boldsymbol{\theta}^T]^T$. Similarly, in quadratic regression we look for a vector of coefficients $\boldsymbol{\theta}$ such that $y_i \simeq \theta_0 + \sum_k \theta_k x_{ik} + \sum_k \sum_\ell \theta_{k,\ell} x_{ik} x_{i\ell}$. This model remains linear with respect to the vector of modified features
>
> $$\boldsymbol{x}_i' = \varphi(\boldsymbol{x}_i) := [1, (x_i)_{i\in[n]}, (x_{ik}x_{i\ell})_{1\leq k\leq \ell\leq n}] \in \mathbb{R}^{1+n+n(n+1)/2}.$$

In vector notation, we can write

$$\boldsymbol{y} = X\boldsymbol{\theta} + \boldsymbol{\epsilon},$$

where $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m]^T \in \mathbb{R}^{m\times n}$ is the matrix whose rows are the feature vectors. The vector $\boldsymbol{\epsilon} \in \mathbb{R}^m$ contains the *errors* between the model and the true observations.

1.1 Least Squares Estimator

The least squares approach consists in choosing $\boldsymbol{\theta}$ so as to minimize the euclidean norm of $\boldsymbol{\epsilon}$. In other words, we want to solve

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^n}{\textbf{minimize}} \quad \|X\boldsymbol{\theta} - \boldsymbol{y}\|^2 \tag{1}$$

The optimal solution of this problem is called the *least squares estimator* of $\boldsymbol{\theta}$, or OLS estimator (for *ordinary least squares*). We show below that it can be computed by solving a simple linear equation, and then we will discuss some of its properties.

Now, let us assume that the matrix X has *full column rank* (i.e., $m \geq n$ and $\operatorname{rank} X = n$). The objective function of Problem (1) can be rewritten as $\boldsymbol{\theta} \mapsto \boldsymbol{\theta}^T X^T X \boldsymbol{\theta} - 2\boldsymbol{\theta}^T X^T \boldsymbol{y} + \boldsymbol{y}^T \boldsymbol{y}$. This is an unconstrained optimization problem, so its solution $\boldsymbol{\theta}^*$ is obtained when the gradient vanishes:

$$2(X^T X \boldsymbol{\theta}^* - X^T \boldsymbol{y}) = 0.$$

The full rank assumption ensures that $X^T X$ is invertible (because[1] $\mathbf{rank}\, X^T X = \mathbf{rank}\, X = n$). Hence, the optimal $\boldsymbol{\theta}^* = \hat{\boldsymbol{\theta}}_{\mathrm{LS}}$ (the subscript LS stands for *least squares*) is the unique solution of the linear system $X^T X \boldsymbol{\theta} = X^T \boldsymbol{y}$, i.e.,

$$\boldsymbol{\theta}^* = (X^T X)^{-1} X^T \boldsymbol{y}. \tag{2}$$

We may ask ourselves why we chosed to minimize the sum of squares of the model errors ϵ_i's (for example, we could also have minimized the sum of the absolute deviations $|\epsilon_i|$, i.e., the L_1-norm of $\boldsymbol{\epsilon}$. One obvious reason is that the choice of the L_2 norm gives the nice analytical solution (2). But there are other, deep statistical reasons for this choice. In particular, we show in Sections 1.2 and 1.3 that simple probabilistic models lead to the same estimator. But minimizing other loss functions of $\boldsymbol{\epsilon}$ also makes sense for particular applications, as will be seen in Section 2.

1.2 Maximum Likelihood Estimator

Assume that all errors ϵ_i are random noise, drawn independently from the same distribution (we say that the ϵ_i are *independently and identically distributed* (i.i.d.)). Let us further assume that the noise is centered ($\mathbb{E}[\epsilon_i] = 0$) and the variance is some (unknown) constant $\mathbb{V}[\epsilon_i] = \sigma^2$. This means that y_i is the realization of a random variable Y_i with $\mathbb{E}[Y_i] = \boldsymbol{x}_i^T \theta$. The probability density of $(Y_1, \ldots, Y_m)$ at $(y_1, \ldots, y_m)$ can be expressed as a function of $\boldsymbol{\theta}$: This is the *likelihood*

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) = \prod_{i=1}^m f_{Y_i}(y_i),$$

where f_{Y_i} is the probability distribution function (PDF) of Y_i.

Now, let us assume that the errors are normally distributed, i.e., $\epsilon_i \sim \mathcal{N}(\mathbf{0}, \sigma^2)$, which implies $Y_i \sim \mathcal{N}(\boldsymbol{x}_i^T \boldsymbol{\theta}, \sigma^2)$. Under the Gaussian assumption,

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) = \prod_{i=1}^m C \cdot \exp\Big(-\frac{1}{2\sigma^2}(y_i - \boldsymbol{x}_i^T \boldsymbol{\theta})^2 \Big),$$

where C is some constant. The *maximum likelihood estimator* $\hat{\boldsymbol{\theta}}$ is the vector that maximizes the above expression. Taking the log, we see that $\hat{\boldsymbol{\theta}}$ is the solution of the optimization problem

$$\underset{\boldsymbol{\theta} \in \mathbb{R}^n}{\mathbf{maximize}} \quad \sum_{i=1}^m -\frac{1}{2\sigma^2}(y_i - \boldsymbol{x}_i^T \boldsymbol{\theta})^2, \tag{3}$$

which is the same problem (up to a constant multiplication factor $-1/2\sigma^2$) as Problem (1). So we have shown that when the errors are normally distributed, the maximimum likelihood estimator of $\boldsymbol{\theta}$ coincides with the OLS estimate.

[1] To see this, one can show that X and $X^T X$ have the same nullspaces. Indeed $X\boldsymbol{u} = \mathbf{0} \implies X^T X \boldsymbol{u} = \mathbf{0}$, and conversely, $X^T X \boldsymbol{u} = \mathbf{0} \implies \boldsymbol{u}^T X^T X \boldsymbol{u} = 0 \implies \|X\boldsymbol{u}\| = 0 \implies X\boldsymbol{u} = \mathbf{0}$.

1.3 Best Linear Unbiased Estimator

More generally (without the normality assumption), we have the Gauss-Markov theorem: This result shows that the least squares estimator is optimal in a very broad sense: its variance is minimal with respect to the $\preceq_{\mathbb{S}^n_+}$- ordering in the class of all unbiased estimator!

Theorem 1. *Let X be a matrix of full column rank. Assume that we have observations $\boldsymbol{y}$ drawn from a random variable Y with $\mathbb{E}[Y] = X\boldsymbol{\theta}$, $\mathbb{V}[Y] = \sigma^2 I_m$. Let $\hat{\boldsymbol{\theta}} = LY$ be an unbiased estimator for $\boldsymbol{\theta}$, that is, $\mathbb{E}[\hat{\boldsymbol{\theta}}] = \boldsymbol{\theta}$ (Note that the estimator $\hat{\boldsymbol{\theta}}$ is a random variable). Then, we have*

$$\mathbb{V}[\hat{\boldsymbol{\theta}}] = L\mathbb{V}[Y]L^T = \sigma^2 LL^T \succeq \sigma^2\,(X^TX)^{-1},$$

where $\succeq$ is the order relative to $\mathbb{S}^n_+$. Moreover, the lower bound is attained for $L^* = (X^TX)^{-1}X^T$. In other words, the best linear unbiased estimator (BLUE) for $\boldsymbol{\theta}$ coincides with the OLS estimate of $\boldsymbol{\theta}$.

Proof. The fact that the lower bound is attained for L^* is clear by substituting its expression in $L^{*T}L^*$.

Hence, the only thing to prove is the matrix inequality. Note that $\mathbb{E}[\hat{\boldsymbol{\theta}}] = LX\boldsymbol{\theta}$. It follows that $\hat{\boldsymbol{\theta}}$ is unbiased if and only if $LX = I_n$. Then, we can see that the matrix

$$\begin{bmatrix} X^TX & I_n \\ I_n & LL^T \end{bmatrix}$$

is positive semidefinite, because it can be written as

$$\begin{pmatrix} X^T \\ L \end{pmatrix}\begin{pmatrix} X^T \\ L \end{pmatrix}^T.$$

Hence, since $X^TX \succ 0$, the Schur complement lemma gives us

$$LL^T \succeq (X^TX)^{-1}.$$

□

2 Beyond least-squares

2.1 Huber regression

Imagine we want to fit a model of the form

$$\boldsymbol{y} = X\boldsymbol{\theta} + \boldsymbol{\epsilon}.$$

Often, real-world data contains *outliers*. Since the square function grows rapidly, minimizing $\sum_i \epsilon_i^2$ might not be a good idea, since the ϵ_i's corresponding to a corrupted sample $(\boldsymbol{x}_i, y_i)$ might be very large. Therefore, it has been proposed to minimize $\sum_i L_\delta(\epsilon_i)$ instead, where $L_\delta : \mathbb{R} \to \mathbb{R}_+$ is the Huber's loss function, defined by

$$L_\delta(x) = \begin{cases} x^2 & \text{for } |x| \leq \delta, \\ \delta(2\,|x| - \delta), & \text{otherwise.} \end{cases}$$

For a threshold parameter $\delta > 0$, L_δ is a convex function which is quadratic near 0, and linear away from 0, with continuous derivatives at $\pm\delta$.

The Huber loss function is SOCP-representable (this requires the introduction of auxiliary variables):

Indeed, it can be seen that

$$L_\delta(x) \leq t \iff \quad \exists a, b \in \mathbb{R} : \begin{cases} a^2 + 2\delta|b| \leq t \\ a + b = x. \end{cases}$$

Hence, the Huber regression problem "**minimize**$_{\boldsymbol{\theta}}$ $\sum_i L_\delta(\boldsymbol{x}_i^T\boldsymbol{\theta} - y_i)$" can be recast as an SOCP. (*Why?*)

2.2 Ridge regression

Recall that in Section 1, we assumed that X had a full column rank, and hence that XX^T was invertible. In practice, it can happen that XX^T is nearly singular/badly conditionned. As a result, the least squares estimate $\boldsymbol{\theta}^*$ can vary a lot if some data x_{ik} is perturbed just a little bit. Even worse, in some situations we might have more features than observations ($n > m$); in this case, X cannot have full column rank.

The ridge regression consists in adding a penalty to the cost function:

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^n}{\textbf{minimize}} \ \|X\boldsymbol{\theta} - \boldsymbol{y}\|^2 + \lambda\|\boldsymbol{\theta}\|^2.$$

Then, the solution of this problem is

$$\hat{\boldsymbol{\theta}}_{\text{ridge}} = (X^TX + \lambda I_n)^{-1}X^T\boldsymbol{y}.$$

The matrix $(X^TX + \lambda I_n)$ is nonsingular for any $\lambda > 0$, so $\hat{\boldsymbol{\theta}}_{\text{ridge}}$ is well defined. There is a tradeoff in the choice of λ: larger values of λ will tend produce larger deviations $|\epsilon_i|$, but the matrix $(X^TX + \lambda I_n)$ is better conditionned, hence the estimator $\hat{\boldsymbol{\theta}}_{\text{ridge}}$ will be more robust. Intuitively, large λ favours smaller values of $\boldsymbol{\theta}$. The larger is λ, the more biased is the estimation, but the smaller is the variance. For $\lambda \geq \lambda' \geq 0$, we can show that

$$\mathbb{V}[\hat{\boldsymbol{\theta}}^{\lambda}_{\text{ridge}}] \preceq \mathbb{V}[\hat{\boldsymbol{\theta}}^{\lambda'}_{\text{ridge}}] \preceq \mathbb{V}[\hat{\boldsymbol{\theta}}_{\text{LS}}],$$

where

$$\mathbb{V}[\hat{\boldsymbol{\theta}}^{\lambda}_{\text{ridge}}] := \sigma^2(X^TX + \lambda I_n)^{-1} \ X^TX \ (X^TX + \lambda I_n)^{-1}.$$

Hint: to show this, we must use the fact that X^TX and $X^TX + \lambda I_n$ commute, hence these matrices can be diagonalized in the same basis.

2.3 Lasso regression

In some situations, the number n of features can be very large, and we want to find a *sparse model*

$$y_i \simeq \sum_{k\in I} \theta_k x_{ik},$$

where I is a small subset of $[n]$. This is equivalent to searching for a model of the form $y_i = \boldsymbol{\theta}^T\boldsymbol{x}_i + \epsilon_i$ for a sparse vector $\boldsymbol{\theta}$ (i.e., a vector $\boldsymbol{\theta}$ with many elements equal to 0. For this purpose, the lasso estimator has been proposed: It solves the optimization problem

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^n}{\textbf{minimize}} \ \|X\boldsymbol{\theta} - \boldsymbol{y}\|^2 + \lambda\|\boldsymbol{\theta}\|_1.$$

Contrarily to standard least squares (or ridge regression), there is no simple closed-form expression for this problem. However, this is a convex optimization problem which can be solved by standard solvers (e.g., it can be reformulated as an SOCP). In practice, we observe that the lasso estimator $\hat{\boldsymbol{\theta}}_{\text{lasso}}$ has more zeroes when λ grows. One reason is that $\boldsymbol{\theta} \mapsto \|\boldsymbol{\theta}\|_1$ serves as a good convex approximation of the ℓ_0 "norm": $\|\boldsymbol{\theta}\|_0 := |\{i \in [n] : \theta_i \neq 0\}|$.

2.4 Elastic-net regression

To combine the advantages of ridge and lasso regressions, the elastic-net estimator solves an optimization problem of the form

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^n}{\textbf{minimize}}\ \|X\boldsymbol{\theta}-\boldsymbol{y}\|^2+\lambda_1\|\boldsymbol{\theta}\|_1+\lambda_2\|\boldsymbol{\theta}\|_2^2,$$

for some parameters $\lambda_1,\lambda_2\geq 0$.

3 Classification

In many machine-learning tasks, the data y_i is binary ($y_i\in\{0,1\}$) and indicates the membership of the ith sample to one of two classes C_0 (if $y_i=0$) or C_1 (if $y_i=1$). We next discuss two methods to compute a *classifier*, that is, a function able to predicts whether a new sample $\boldsymbol{x}$ belongs to the class C_0 or C_1.

3.1 Logistic regression

In logistic regression, we use maximum likelihood estimation to fit a probabilistic model of the form

$$\mathbb{P}[y_i=1]=f(\boldsymbol{x}_i^T\boldsymbol{\theta}),$$

where $f:\mathbb{R}\to[0,1]$ is the logistic function $f:u\mapsto\frac{e^u}{1+e^u}$. Then, if we are given a new sample $\boldsymbol{x}_*$, we can predict that it belongs to the class C_1 with probability $f(\boldsymbol{x}_*^T\boldsymbol{\theta})$.

If the observations are independent, we can write the likelihood of the parameter $\boldsymbol{\theta}\in\mathbb{R}^n$ as the product

$$\mathcal{L}(\boldsymbol{\theta}|X,\boldsymbol{y})=\prod_{\substack{i\in[m]\\ y_i=1}}\mathbb{P}[y_i=1]\cdot\prod_{\substack{i\in[m]\\ y_i=0}}\mathbb{P}[y_i=0]=\prod_{i=1}^m\frac{\exp(y_i\cdot\boldsymbol{x}_i^T\boldsymbol{\theta})}{1+\exp(\boldsymbol{x}_i^T\boldsymbol{\theta})}.$$

Taking the log, we obtain a convex optimization problem:

$$\underset{\boldsymbol{\theta}\in\mathbb{R}^n}{\textbf{maximize}}\ \log\mathcal{L}(\boldsymbol{\theta}|X,\boldsymbol{y})=\sum_{i=1}^m y_i\cdot\boldsymbol{x}_i^T\boldsymbol{\theta}-\log(1+\exp(\boldsymbol{x}_i^T\boldsymbol{\theta}))$$

The convexity of the above problem follows from the convexity of the log-sum-exp function: The function $u\mapsto\log(1+\exp(u))=\log(\exp(0)+\exp(u))$ is convex, so after composition with a linear map, we see that the objective function used in logistic regression is concave.

3.2 Support vector machines

Another *deterministic approach* consists in trying to *geometrically* separate the points $(\boldsymbol{x}_i)_{i\in C_0}$ from the points $(\boldsymbol{x}_i)_{i\in C_1}$. The simplest approach is to search for a hyperplane in $\mathbb{R}^n$ which separates those two groups of points. We will see in the next section how to extend the result of this section to search for nonlinear separators.

For the ease of notation, in this section we assume that $y_i=-1$ for samples in the class C_0, and $y_i=1$ for samples in the class C_1. Then, a separating hyperplane is a nonzero vector $\boldsymbol{w}\in\mathbb{R}^n$ and a scalar b such that

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i-b)>0,\qquad(\forall i\in[n]).$$

The next theorem explains when the data can be separated by a hyperplane:

Theorem 2. *The data* $(\boldsymbol{x}_i, y_i),\ i = 1, \ldots, m$, *is linearly separable if and only if the sets* $\mathbf{conv}\,\{\boldsymbol{x}_i : i \in C_0\}$ *and* $\mathbf{conv}\,\{\boldsymbol{x}_j : j \in C_1\}$ *do not intersect.*

Proof. $\Longleftarrow$ If the convex hulls do not intersect, then they can be strictly separated by a hyperplane (this is the strict version of the separating hyperplane theorem, which we can use because the two sets are compact.)

$\Longrightarrow$ Conversely, assume that the data is linearly separable, so $\exists a \in \mathbb{R}^n, b \in \mathbb{R}$:

$$\boldsymbol{w}^T\boldsymbol{x}_i < b, \quad \forall i \in C_0 \qquad \text{and} \qquad \boldsymbol{w}^T\boldsymbol{x}_i > b, \quad \forall i \in C_1.$$

Then, by convexity we have $\boldsymbol{w}^T\boldsymbol{x} < b$ for all $\boldsymbol{x} \in \mathbf{conv}\,\{\boldsymbol{x}_i : i \in C_0\}$ and $\boldsymbol{w}^T\boldsymbol{x} > b$ for all $\boldsymbol{x} \in \mathbf{conv}\,\{\boldsymbol{x}_i : i \in C_1\}$, which implies that these convex hulls do not intersect. □

Hard-margin (robust) separation Assume that the data is separable. We want to separate the data in the following manner:

$$\boldsymbol{w}^T\boldsymbol{x}_i \leq b - 1, \quad \forall i \in C_0 \qquad \text{and} \qquad \boldsymbol{w}^T\boldsymbol{x}_i \geq b + 1, \quad \forall i \in C_1.$$

Or equivalently,

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) \geq 1, \qquad \forall i \in [m].$$

To make the classifier robust, we want the stripe $S = \{\boldsymbol{x} : |\boldsymbol{w}^T\boldsymbol{x} - b| \leq 1\}$ to be as wide as possible. The width of the stripe is given by $\delta = \frac{2}{\|\boldsymbol{w}\|}$, so the hard-margin support vector machine (SVM) solves the following problem (which is equivalent to an SOCP):

$$\begin{aligned} \underset{\boldsymbol{w}\in\mathbb{R}^n, b\in\mathbb{R}}{\textbf{minimize}} \quad & \|\boldsymbol{w}\|^2 \\ \textit{s.t.} \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) \geq 1, \qquad \forall i \in [m]. \end{aligned}$$

Soft-margin classifier Since it is not always possible to linearly separate the data, we can also search for a *soft-margin* classifier. Ideally, we would like the stripe S to be very wide, and empty. Instead, the soft-margin SVM minimizes a combination of $\|\boldsymbol{w}\|^2$ (to have a *wide margin*), and a penalty for points lying on the wrong side of the margin:

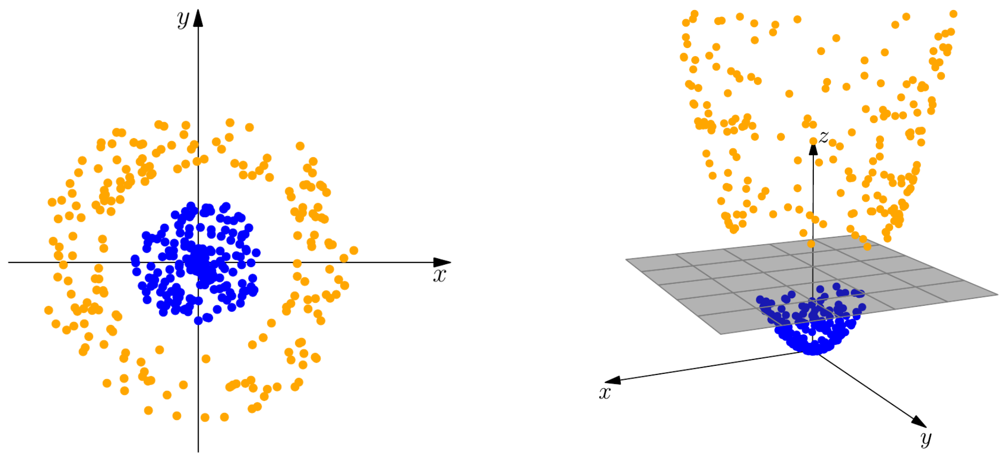
$$\phi(\boldsymbol{w}, \boldsymbol{b}; \boldsymbol{x}_i, y_i) = \begin{cases} 0 & \text{if } y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) \geq 1; \\ 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) & \text{otherwise,} \end{cases}$$

which can be written concisely as $\phi(\boldsymbol{w}, \boldsymbol{b}; \boldsymbol{x}_i, y_i) = \max(0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b))$. Finally, for a parameter $\lambda > 0$ that sets the tradeoff between the two objectives (wide margin, points on the correct side of the margin), the SVM solves

$$\underset{\boldsymbol{w}\in\mathbb{R}^n, b\in\mathbb{R}}{\textbf{minimize}} \quad \sum_{i=1}^m \max(0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b)) + \lambda\|\boldsymbol{w}\|^2$$

This can be rewritten as the following SOCP:

$$\begin{aligned} \underset{\boldsymbol{w},\boldsymbol{\zeta}\in\mathbb{R}^n,\ b,t\in\mathbb{R}}{\textbf{minimize}} \quad & \sum_{i=1}^m \zeta_i + \lambda t \\ \textit{s.t.} \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) \geq 1 - \zeta_i, \qquad \forall i \in [m] \\ & \zeta_i \geq 0, \qquad \forall i \in [m]. \\ & \|\boldsymbol{w}\|^2 \leq t. \end{aligned} \tag{4}$$



Points $\boldsymbol{x}_i$ in original feature space. Points $\boldsymbol{x}_i' = \varphi(\boldsymbol{x}_i)$ in augmented feature space.

Figure 1: The points $\boldsymbol{x}_i$ in the original feature space (left) are not linearly separable (the color of the points indicate their membership to the class C_0 or C_1). However, after a suitable embedding in a space of larger dimensions, the points $\boldsymbol{x}_i' = \varphi(\boldsymbol{x}_i)$ become linearly separable. *Credits: C. Bauckhage.*

4 The kernel trick

In the introduction, we explained that linear regression techniques can be used to learn a nonlinear relation between the features $\boldsymbol{x}_i$ and some observations y_i. For example, we mentioned the case of quadratic regression, which can be handled by working with the vector of modified features

$$\boldsymbol{x}' = [1, \boldsymbol{x}, \text{lower-triangle}(\boldsymbol{x}\boldsymbol{x}^T)] \in \mathbb{R}^{1+n+n(n+1)/2}.$$

The same also works for classification tasks: transforming the initial vector of features $\boldsymbol{x}$ to a vector $\boldsymbol{x}' = \varphi(\boldsymbol{x})$ of larger dimensions may help to perform classification tasks. As an example, consider the data depicted in Figure 1. Here, the initial points cannot be linearly separated, but the augmented vectors

$$\boldsymbol{x}_i' = \varphi(\boldsymbol{x}_i) := [x_{i1}, x_{i2}, x_{i1}^2 + x_{i2}^2]$$

are perfectly separable by a hyperplane!

However, this raises two kinds of issue: First, it is often not clear how to select the function φ. Second, the transformation $\varphi : \mathbb{R}^n \to \mathbb{R}^{n'}$ may drastically improve the dimension of the feature vectors. For example, for quadratic regression we already need $n' = O(n^2)$ features. As a result, the learning algorithm might become inefficient.

One way to overcome these issues is to use the *Kernel trick*. We briefly describe (without proofs) how this works. We first define the notion of positive semidefinite kernels, which are a kind of *infinite positive semidefinite matrix*.

Definition 1. Positive semidefinite kernel. A function $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is called a *positive semidefinite kernel*, if for all $N \in \mathbb{N}$ and for all $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^n$, the matrix

$$K[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N] := \{K(\boldsymbol{x}_i, \boldsymbol{x}_j)\}_{1 \leq i,j \leq N} \in \mathbb{S}_+^N.$$

As for the case of finite matrices, there are equivalent definitions. For example, it can be shown that

$K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a positive semidefinite kernel iff for all square-integrable function $f : \mathbb{R}^n \to \mathbb{R}$, it holds:

$$\int_{\boldsymbol{x}\in\mathbb{R}^n} \int_{\boldsymbol{y}\in\mathbb{R}^n} K(\boldsymbol{x},\boldsymbol{y}) f(\boldsymbol{x}) f(\boldsymbol{y})\, d\boldsymbol{x}\, d\boldsymbol{y} \;\geq\; 0.$$

A simplified version of *Mercer's theorem* follows:

Theorem 3 (Mercer (simplified)). *Let K be a positive semidefinite kernel. Then, there exists a function φ such that*

$$K(\boldsymbol{x},\boldsymbol{y}) = \langle \varphi(\boldsymbol{x}), \varphi(\boldsymbol{y}) \rangle.$$

Intuitively, the idea is that, as for finite positive semidefinite matrices, positive semidefinite kernels have an eigendecomposition of the form

$$K(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{\infty} \lambda_i \psi_i(\boldsymbol{x}) \psi_i(\boldsymbol{y}),$$

and the functions ψ_i are called the *eigenfunctions* of K. Then, the function φ of the theorem is the (possibly infinite) sequence $\varphi(\boldsymbol{x}) = \big(\sqrt{\lambda_i}\psi_i(\boldsymbol{x})\big)_{i\in\mathbb{N}}$.

We now give two examples of kernels which are used a lot in practice (without proof that they are positive semidefinite):

Example:
The function $K(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}^T\boldsymbol{y}+1)^d$ is a positive semidefinite kernel, associated with a feature vector $\varphi(\boldsymbol{x})$ of dimension $\binom{n+d}{d}$. The coordinates of $\varphi(\boldsymbol{x})$ form a basis of the space $\mathbb{R}_d[x_1,\ldots,x_n]$ of polynomials of degree d on n variables. #1

Example:
The function $K(\boldsymbol{x},\boldsymbol{y}) = \exp(-\frac{1}{2\sigma^2}\|\boldsymbol{x}-\boldsymbol{y}\|^2)$ is a positive semidefinite kernel. Its associated feature vector $\varphi(\boldsymbol{x})$ is in fact an infinite sequence. #2

Now, the idea of the *kernel trick* is as follows:

- Assume that we have a regression or classification task, in which the learning algorithm only depends on the scalar products $\boldsymbol{x}_i^T\boldsymbol{x}_j$ between different samples. (*This is the case for many machine learning tasks, such as ridge regression or SVM. We will explain in more details how this works for ridge regression on the next page).*
- Let K be a positive semidefinite kernel associated with a Mercer's function φ (maybe this function φ is unknown, we just need to know that such a function exists).
- If we want to apply the learning algorithm to the vector of modified features $\boldsymbol{x}' = \varphi(\boldsymbol{x})$, our algorithm needs to compute scalar products of the form $\varphi(\boldsymbol{x})^T\varphi(\boldsymbol{y})$.
- Instead, use Mercer theorem and replace all occurences of $\varphi(\boldsymbol{x})^T\varphi(\boldsymbol{y})$ in the algorithm by $K(\boldsymbol{x},\boldsymbol{y})$.

We next show how to derive the kernelized version of the ridge-regression. We point out that it is also possible to make a kernelized version of the SVM, because the dual of the SOCP (4) (we will learn about dual optimization problems in the next chapter) only depends on the scalar products $\boldsymbol{x}_i^T\boldsymbol{x}_j$.

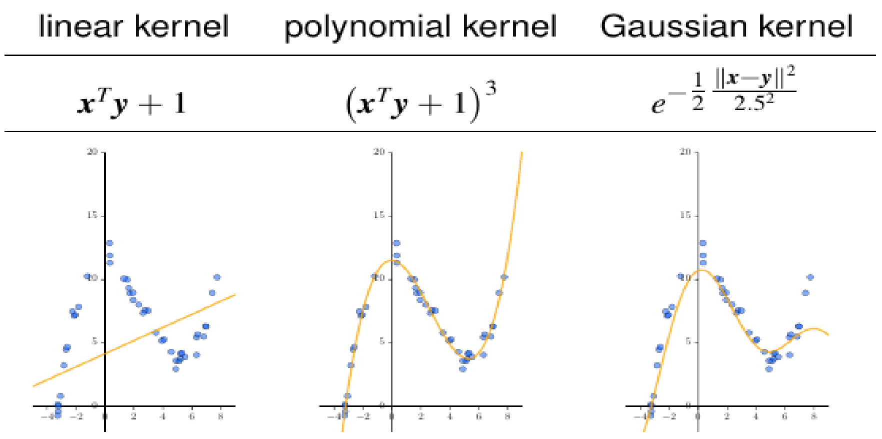


Figure 2: Example of a simple regression wih three different kernels. *Credits: C. Bauckhage*

Example:
We show how to use the Kernel trick for ridge regression.
We recall that the ridge-estimator of the vector of coefficients $\boldsymbol{\theta}$ is

$$\boldsymbol{\theta}^* = (X^T X + \lambda I)^{-1} X^T \boldsymbol{y}.$$

Then, if we want to predict the response y_* corresponding to a new sample $\boldsymbol{x}_*$, we do

$$\hat{y_*} = \boldsymbol{x}_*^T \boldsymbol{\theta}^* = \boldsymbol{x}_*^T (X^T X + \lambda I_n)^{-1} X^T \boldsymbol{y}.$$

After some linear algebra, we can show that[a]

$$\hat{y_*} = \boldsymbol{x}_*^T X^T (X X^T + \lambda I_m)^{-1} \boldsymbol{y}.$$

Using the above formula is not a good idea in general, because we need to compute the inverse a $m \times m$-matrix instead of a $n \times n$ matrix, and we usually have more observations than features (in fact, note that we don't have to invert the matrix, we just have to solve a linear system!). However, the last formula will allow us to use the kernel trick. Indeed, it can be rewritten as $\hat{y_*} = \boldsymbol{k}^T (Q + \lambda I_m)^{-1} \boldsymbol{y}$, where $\boldsymbol{k}$ is the vector with elements $k_i = \boldsymbol{x}_*^T \boldsymbol{x}_i$ and Q is the matrix with elements $\boldsymbol{x}_i^T \boldsymbol{x}_j$. Now, the *kernelized ridge regression* simply consists in replacing the vector $\boldsymbol{k}$ by $\boldsymbol{k}'(\boldsymbol{x}_*) = [K(\boldsymbol{x}_*, \boldsymbol{x}_1), \ldots, K(\boldsymbol{x}_*, \boldsymbol{x}_m)]$ amd the matrix Q by $Q' = \{K(\boldsymbol{x}_i, \boldsymbol{x}_j)\}_{1 \leq i,j \leq m}$. In practice, one need to solve the $m \times m$ linear system #3

$$(Q + \lambda I_m)\boldsymbol{z} = \boldsymbol{y}$$

for $\boldsymbol{z} \in \mathbb{R}^m$ during the training phase. Then, the prediction for a new sample $\boldsymbol{x}_*$ in the testing set is:

$$\hat{y_*} = \boldsymbol{k}'(\boldsymbol{x}_*)^T \boldsymbol{z}.$$

An example of regression in the case $n = 1$ with three different kernels is shown in Figure 2.

[a] This is not trivial! It is a consequence of the *Woodburry matrix identity*.

5 Design of Experiments

In many applications, we have to run an experiment to obtain samples $(\boldsymbol{x}_i, y_i)$. When the budget for experimentation is limited, it is important to carefully select the vector of features $\boldsymbol{x}_i$ for which we want to obtain a measurement y_i. For simplicity, assume that experimental trials can be conducted at any of the points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \in \mathbb{R}^n$. We assume that measurements are linear, as in the model of Section 1: $y = \boldsymbol{x}^T\boldsymbol{\theta} + \epsilon$, where ϵ is centered at $\mathbf{0}$ and has variance σ^2.

Denote by n_i the number of trials to perform at $\boldsymbol{x}_i$ (experiments can be replicated, which yields several independent observations of the random variable Y_i). A total of N experimental trials has to be selected, i.e., $\sum_{i=1}^m n_i = N$. Then, we obtain a big vector with all collected observations, of the form

$$\boldsymbol{y} = X^T\boldsymbol{\theta} + \boldsymbol{\epsilon},$$

where

$$X = [\underbrace{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_1}_{n_1\text{times}}, \underbrace{\boldsymbol{x}_2, \ldots, \boldsymbol{x}_2}_{n_2\text{times}}, \ldots, \underbrace{\boldsymbol{x}_m, \ldots, \boldsymbol{x}_m}_{n_m\text{times}}]^T \in \mathbb{R}^{N\times n}.$$

The Gauss-Markov theorem tells us that the best linear unbiased estimator for $\boldsymbol{\theta}$ coincides with the OLS estimate, and its variance is proportional to

$$N(X^TX)^{-1} = \big(\sum_{i=1}^m \frac{n_i}{N}\boldsymbol{x}_i\boldsymbol{x}_i^T\big)^{-1}.$$

We make the change of variable $w_i = \frac{n_i}{N}$, so the *continuous* variable w_i can be interpreted as the fraction of experimental effort to spend at $\boldsymbol{x}_i$. Note that the weights w_i satisfy $\sum_{i=1}^m w_i = 1$.

Now, the optimal design of experiment (DoE) is to select the weights that "minimize" the above variance-covariance matrix, or equivalently, that "maximize" the information matrix

$$M(\boldsymbol{w}) = \sum_{i=1}^m w_i\ \boldsymbol{x}_i\boldsymbol{x}_i^T.$$

The natural criterion would be to maximize the above matrix for the $\preceq_{\mathbb{S}_+^n}$-ordering. However, in general this problem is ill-posed, because $\preceq_{\mathbb{S}_+^n}$ is not a total order. Instead it has been proposed to maximize some scalar criterion of the information matrix. If we assume that the errors ϵ_i are normally distributed, it can be seen that the error of estimation $\boldsymbol{\delta} = \hat{\boldsymbol{\theta}}_{\text{LS}} - \boldsymbol{\theta}$ lies with high probability in a confidence ellipsoid of the form

$$\{\boldsymbol{\delta} \in \mathbb{R}^n : \boldsymbol{\delta}^T M(\boldsymbol{w})^{-1}\boldsymbol{\delta} \leq \kappa\},$$

where κ is a constant depending on the confidence level. We recall that this ellipsoid has semi-axis of lengths proportional to $\frac{1}{\sqrt{\lambda_i}}$, where the λ_i's are the eigenvalues of $M(\boldsymbol{w})$. Two popular criteria for optimal DoE are:

- The $D-$optimality criterion. This criterion aims at minimizing the volume of the confidence ellipsoids, which is proportional to $\prod_i \lambda_i^{-1/2}$. This is equivalent to maximize the (nth root of the) determinant of the information matrix:
$$\Phi_D(\boldsymbol{w}) = (\det M(\boldsymbol{w}))^{1/n}.$$
We know that this criterion is SDP-representable. The problem of maximizing Φ_D over the set of weights $\{\boldsymbol{w} \in \mathbb{R}_+^m : \sum_{i=1}^m w_i = 1\}$ is hence an SDP.

- The $A-$optimality criterion aims at minimizing the diagonal of the bounding box of the confidence ellipsoids:
$$\Phi_A(\boldsymbol{w}) = \sum_{i=1}^n \left(\frac{1}{\sqrt{\lambda_i}}\right)^2 = \sum_{i=1}^m \frac{1}{\lambda_i} = \text{trace }\ M(\boldsymbol{w})^{-1}$$

By using the Schur-complement lemma, the A-optimality DoE problem can be formulated as the following SDP:

$$\begin{aligned} \underset{\boldsymbol{w}\in\mathbb{R}^m, Y\in\mathbb{S}^n}{\textbf{minimize}} \quad & \operatorname{trace} Y \\ s.t. \quad & \begin{bmatrix} M(\boldsymbol{w}) & I_n \\ I_n & Y \end{bmatrix} \succeq 0 \\ & \sum_{i=1}^m w_i = 1 \\ & \boldsymbol{w} \geq \boldsymbol{0}. \end{aligned}$$

Proof. This problem is an SDP indeed, because $M(\boldsymbol{w})$ depends linearly on the decision variable $\boldsymbol{w}$.

To derive this SDP formulation, we show that Φ_A is SDP-representable:

$$\Phi_A(\boldsymbol{w}) \leq t \iff \exists Y \in \mathbb{S}^n : \begin{cases} \operatorname{trace} Y \leq t \\ \begin{bmatrix} M(\boldsymbol{w}) & I_n \\ I_n & Y \end{bmatrix} \succeq 0. \end{cases}$$

$\Longleftarrow$: The SC lemma tells us that $Y \succeq M(\boldsymbol{w})^{-1}$ (By using the extended SC lemma, we see that $M(\boldsymbol{w})$ cannot be singular, because we must have $\mathbf{Im}\, I_n = \mathbb{R}^n \subseteq \mathbf{Im}\, M(\boldsymbol{w})$). Then, this implies $\operatorname{trace} Y - \operatorname{trace} M(\boldsymbol{w})^{-1} \geq 0 \implies t \geq \operatorname{trace} Y \geq \Phi_A(\boldsymbol{w})$.

$\Longrightarrow$ The inequality $\Phi_A(\boldsymbol{w}) \leq t$ implies that the matrix $M(\boldsymbol{w})$ is non-singular, because otherwise $\boldsymbol{w}$ is not in the domain of Φ_A, hence $\Phi_A(\boldsymbol{w}) = \infty$. Then, setting $Y = M(\boldsymbol{w})^{-1}$ yields a feasible solution for the LMI on the right-hand side. $\square$

Note: In fact, there is also an SOCP formulation for the $A-$ and $D-$optimal DoE problems, but they are more complicated.