Theory of Statistical Learning
Part II

Damien Garreau

Université Côte d'Azur

2022

Outline

1. Linear predictors
    - Linear classification
    - Linear regression
    - Ridge regression
    - Polynomial regression
    - Logistic regression
    - Support vector machines
2. Kernel methods
    - Positive semi-definite kernels
    - Reproducing kernel Hilbert spaces
    - More examples
    - The kernel trick and applications
    - The representer theorem
    - Kernel ridge regression
3. Tree-based classifiers
    - Partition rules
    - Random forests
4. Boosting

1. Linear predictors

1.1. Linear classification

Linear functions

- $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$
- thus $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d})^\top$
- we consider no bias term (otherwise *affine*):

$$\{h : x \mapsto w^\top x, w \in \mathbb{R}^d\} .$$

- **Reminder:** given two vectors $u, v \in \mathbb{R}^d$,

$$\langle u, v \rangle = u^\top v = \sum_{j=1}^{d} u_i v_i .$$

- binary classification: 0-1 loss, $\mathcal{Y} = \{-1, +1\}$
- **Important:** compose h with $\phi : \mathbb{R} \to \mathcal{Y}$ (typically the sign)

The sign function

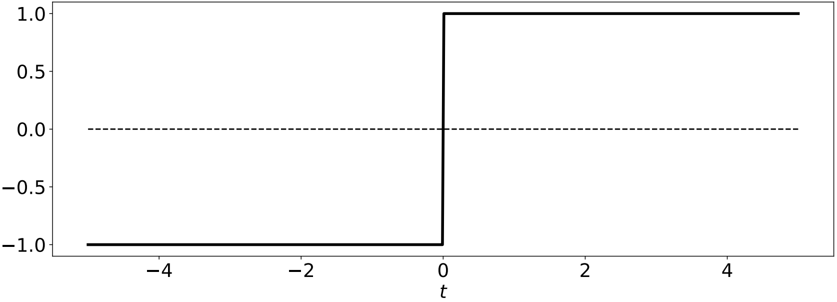


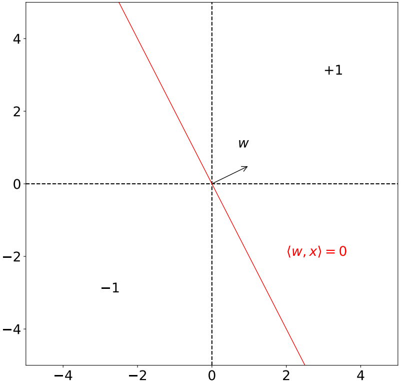
Figure: the sign function

Halfspaces

- thus our function class is

$$\mathcal{H} = \left\{ x \mapsto \operatorname{sign}\left(w^\top x\right), w \in \mathbb{R}^d \right\}.$$

- gives label $+1$ to vector pointing in the same direction as w



VC dimension of halfspaces

Proposition: the VC dimension of halfspaces in dimension d is exactly $d + 1$.

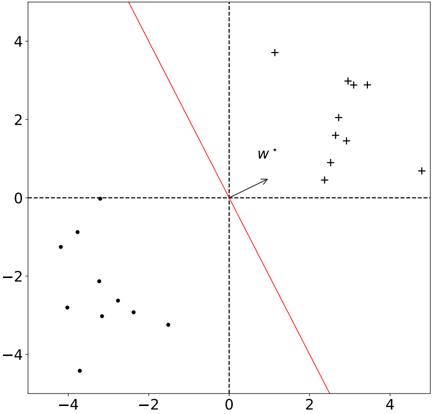
- **Consequence:** $\mathcal{H}$ is PAC learnable with sample complexity

$$\Omega\left(\frac{d + \log(1/\delta)}{\varepsilon}\right) .$$

Linearly separable data

- **Important assumption:** data is linearly separable
- that is, there is a $w^\star \in \mathbb{R}^d$ such that

$$y_i = \operatorname{sign}\left(\langle w^\star, x_i \rangle\right) \quad \forall 1 \leq i \leq n\,.$$



Linear programming

- **Empirical risk minimization:** recall that we are looking for w such that

$$\hat{\mathcal{R}}_S(w) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{y_i \neq \mathrm{sign}(w^\top x_i)}$$

  is minimal
- **Question:** how to solve this?
- we want $y_i = \mathrm{sign}\left(w^\top x_i\right)$ for all $1 \leq i \leq n$
- equivalent formulation: $y_i\langle w, x_i\rangle > 0$
- we know that there is a vector that satisfies this condition $(w^\star)$
- let us set $\gamma = \min_i\{y_i\langle w^\star, x_i\rangle\}$ and $\overline{w} = w^\star/\gamma$
- we have shown that there is a vector such that $y_i\langle \overline{w}, x_i\rangle \geq 1$ for any $1 \leq i \leq n$ (and it is an ERM)

Linear programming, ctd.

- define the matrix $A \in \mathbb{R}^{n \times d}$ such that

$$A_{i,j} = y_i x_{i,j} \,.$$

- **Intuition:** observations × labels
- remember that we have the ± 1 label convention
- define $v = (1, \ldots, 1)^\top \in \mathbb{R}^n$
- then we can rewrite the above problem as

$$\text{maximize} \quad \langle u, w \rangle \text{ subject to } Aw \leq v \,.$$

- we call this sort of problems **linear programs**[1]
- solvers readily available, e.g., `scipy.optimize.linprog` if you use Python

[1] Boyd, Vandenberghe, *Convex optimization*, Cambridge University Press, 2004

The perceptron

- another possibility: the *perceptron*[2]
- **Idea:** iterative algorithm that constructs $w^{(1)}, w^{(2)}, \ldots, w^{(T)}$
- update rule: at each step, find i that is misclassified and set

$$w^{(t+1)} = w^{(t)} + y_i x_i \, .$$

- **Question:** why does it work?
- pushes w in the right direction:

$$y_i \langle w^{(t+1)}, x_i \rangle = y_i \langle w^{(t)} + y_i x_i, x_i \rangle = y_i \langle w^{(t)}, x_i \rangle + \|x_i\|^2$$

- remember, we want $y_i \langle w, x_i \rangle > 0$ for all i

[2] Rosenblatt, *The perceptron, a perceiving and recognizing automaton*, tech report, 1957

1.2. Linear regression

Least squares

- regression ⇒ squared-loss function

$$\ell(y, y') = (y - y')^2 \, .$$

- still looking at linear functions:

$$\mathcal{H} = \{h : x \mapsto \langle w, x \rangle \text{ s.t. } w \in \mathbb{R}^d\} \, .$$

- empirical risk in this context:

$$\hat{\mathcal{R}}_S(h) = \frac{1}{n} \sum_{i=1}^{n} (w^\top x_i - y_i)^2 = F(w) \, .$$

- also called **mean squared error**
- empirical risk minimization: we want to minimize $w \mapsto F(w)$ with respect to $w \in \mathbb{R}^d$
- F is a convex, smooth function

Least squares, ctd.

- let us compute the gradient of F:

$$\begin{aligned}
\frac{\partial F}{\partial w_j}(w) &= \frac{1}{n}\sum_{i=1}^{n} \frac{\partial}{\partial w_j}(w^\top x_i - y_i)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n} 2 \cdot \frac{\partial}{\partial w_j}\left(w^\top x_i - y_i\right) \cdot (w^\top x_i - y_i) \\
&= \frac{1}{n}\sum_{i=1}^{n} 2 \cdot \frac{\partial}{\partial w_j}\left(\cdots + w_j x_{i,j} + \cdots - y_i\right) \cdot (w^\top x_i - y_i) \\
\frac{\partial F}{\partial w_j}(w) &= \frac{2}{n}\sum_{i=1}^{n} x_{i,j} \cdot (w^\top x_i - y_i)\,.
\end{aligned}$$

Least squares, ctd.

- it is more convenient to write $\nabla F(w) = 0$ in matrix notation
- define $X \in \mathbb{R}^{n\times d}$ the matrix such that line i of X is observation x_i
- one can check that, for any $1 \leq j,k \leq d$,

$$\left(X^\top X\right)_{j,k} = \sum_{i=1}^{n} x_{i,j} x_{i,k} \, .$$

- thus

$$\begin{aligned}\left(X^\top X w\right)_j &= \sum_{k=1}^{d} \left(X^\top X\right)_{j,k} w_k \\ &= \sum_{k=1}^{d} \sum_{i=1}^{n} x_{i,j} x_{i,k} w_k \\ &= \sum_{i=1}^{n} x_{i,j} w^\top x_i \, .\end{aligned}$$

Least squares, ctd.

- thus, if we define

$$A = X^\top X = \sum_{i=1}^{n} x_i x_i^\top \in \mathbb{R}^{d\times d} \text{ and } b = X^\top y = \sum_{i=1}^{n} y_i x_i \in \mathbb{R}^d \,,$$

  solving $\nabla F(w) = 0$ is equivalent to solving

$$Aw = b\,.$$

- if A is invertible, straightforward:

$$\boxed{\hat{w} = A^{-1} b}$$

- computational cost: $\mathcal{O}\left(d^3\right)$ (inversion is actually a bit less)
- what happens when A is not invertible?

Singular value decomposition

- since A is symmetric, it has an eigendecomposition

$$A = VDV^\top \,,$$

  with $D \in \mathbb{R}^d$ diagonal and V orthonormal
- define D^+ such that

$$D^+_{i,i} = 0 \text{ if } D_{i,i} = 0 \text{ and } D^+_{i,i} = \frac{1}{D_{i,i}} \text{ otherwise.}$$

- define $A^+ = VD^+V^\top$
- then we set

$$\boxed{\hat{w} = A^+ b \,.}$$

Singular value decomposition, ctd.

- why did we do that?
- let v_i denote the ith column of V, then

$$\begin{aligned} A\hat{w} &= AA^+ b && \text{(definition of } \hat{w}\text{)} \\ &= VDV^\top VD^+ V^\top b && \text{(definition of } A^+\text{)} \\ &= VDD^+ V^\top b && \text{(}V \text{ is orthonormal)} \\ A\hat{w} &= \sum_{i:D_{i,i}\neq 0} v_i v_i^\top b\,. \end{aligned}$$

- in definitive, $A\hat{w}$ is the projection of b onto the span of v_i such that $D_{i,i} \neq 0$
- since the span of these v_i is the span of the x_i and b is in the linear span of the x_i, we have $A\hat{w} = b$
- cost of SVD: $\mathcal{O}\left(dn^2\right)$ if $d > n$ (SVD of X)

Exercise

Exercise: Of course, one does not have to use the squared loss. Instead, we may prefer to use

$$\ell(y, y') = |y - y'| \ .$$

1. show that, for any $v \in \mathbb{R}^d$,

$$\|v\|_1 = \min_z \mathbf{1}^\top z \quad \text{subject to} \quad z \geq |v| \ .$$

2. deduce that ERM with the absolute value loss function is equivalent to minimizing the linear function $\sum_{i=1}^n s_i$, where the s_i satisfy linear constraints
3. write this as a linear program, that is, find $A \in \mathbb{R}^{2n \times (n+d)}$, $v \in \mathbb{R}^{d+n}$, and $b \in \mathbb{R}^{2n}$ such that the problem can be written

$$\text{minimize } c^\top v \quad \text{subject to} \quad Av \leq b \, .$$

Correction of the exercise

1. We have $|v| \geq |v|$ and $\mathbf{1}^\top |v| = \|v\|_1$.
2. In that case, the empirical risk can be written

$$\hat{\mathcal{R}}_S(w) = \frac{1}{n} \sum_{i=1}^{n} \left| y_i - w^\top x_i \right| \ .$$

We deduce the result from question 1.
3. One possibility is to define $v = (w_1, \dots, w_d, s_1, \dots, s_n)^\top \in \mathbb{R}^{n+d}$, $c = (0, \dots, 0, 1, \dots, 1)^\top \in \mathbb{R}^{d+n}$, $b = (y_1, \dots, y_n, -y_1, \dots, -y_n)^\top \in \mathbb{R}^{2n}$, and

$$A = \begin{pmatrix} X & -I_n \\ -X & -I_n \end{pmatrix} \in \mathbb{R}^{2n \times (n+d)} \ ,$$

with $X \in \mathbb{R}^{n \times d}$ the matrix whose lines are the x_is and I_n the identity matrix.

Recap

- **What happens when we invoke** `sklearn.linear_model.LinearRegression` with default parameters?
- `fit_intercept` is True → assumes that the data is not centered (our maths are not totally accurate)
- `normalize` is False → we are responsible for the normalization of our data
- behind the scenes, calls `scipy.linalg.lstsq` when fitting, which itself calls LAPACK (Linear Algebra PACKage)[3]
- LAPACK is coded in Fortran90

[3] `http://www.netlib.org/lapack/`

1.3. Ridge regression

Ridge regression

- same hypothesis class: linear functions

$$\mathcal{H} = \{h : x \mapsto w^\top x, w \in \mathbb{R}^d\}$$

- squared loss:

$$\ell(y, y') = (y - y')^2 \,.$$

- **Idea:** regularization:

$$\text{minimize} \left\{ \frac{1}{n} \sum_{i=1}^{n} (y_i - w^\top x_i)^2 + \lambda \left\| w \right\|^2 \right\},$$

with $\left\| u \right\|^2 = u_1^2 + \cdots + u_d^2$ and $\lambda > 0$ a *regularization parameter*

Exercise

Exercise: Let $(x_1, y_1), \dots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$ be n given training samples. For any $w \in \mathbb{R}^d$, set

$$F(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - w^\top x_i)^2 + \lambda \left\| w \right\|^2 \, .$$

Notice that F is a convex smooth function.

1. show that the minimizer $\hat{w}$ satisfies

$$\left(X^\top X + n\lambda I_d \right) w = X^\top y \, .$$

2. show that $X^\top X + n\lambda I_d$ is an invertible matrix

Correction of the exercise

1. Let $1 \leq j \leq d$ and let us compute $\partial_j F$:

$$\frac{\partial F}{\partial w_j}(w) = \frac{\partial}{\partial w_j}\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - w^\top x_i)^2\right) + \frac{\partial}{\partial w_j}(\lambda(w_1^2 + \cdots w_d^2))$$

$$= \frac{2}{n}\sum_{i=1}^{n} x_{i,j} \cdot (w^\top x_i - y_i) + 2\lambda w_j \,,$$

where we used the derivation for the least squares. We deduce the result by setting to zero and multiplying by n.

Correction of the exercise, ctd.

2. By contradiction, suppose that $X^\top X + n\lambda I_d$ is not invertible. Then

$$\det\left(X^\top X + n\lambda I_d\right) = 0\,.$$

In other words, $-n\lambda$ is an eigenvalue of $X^\top X$. Since $X^\top X$ is a symmetric matrix, its spectrum is $\subseteq \mathbb{R}$. Moreover, it is positive definite, thus all eigenvalues are non-negative. Since $\lambda > 0$, we deduce that $-n\lambda$ cannot be an eigenvalue of $X^\top X$ and we can conclude. □

Recap

- **What happens when we invoke** `sklearn.linear_model.Ridge` with default settings?
- `alpha` $= 1 \rightarrow \lambda = 1/n$ with our notation, barely any regularization if n large
- `fit_intercept` is True → does not consider centered data (so our analysis is not entirely accurate)
- `normalize` is False → we decide whether we normalize our data
- `solver` is auto → `sklearn` will decide how to solve the minimization problem depending on the size of the data: the solution could be not exact!
- `tol` $= 0.001$ → tolerance threshold on the residuals

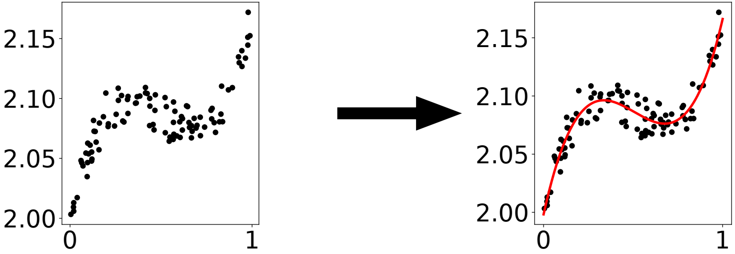
1.4. Polynomial regression

Polynomial regression

- linear regression is a powerful tool, especially because we can transform the inputs in a non-linear fashion
- **Example:** polynomial regression in $\mathbb{R}$
- inputs $x_1, \ldots, x_n \in \mathbb{R}$
- define the mapping $\phi(x) = (1, x, x^2, \ldots, x^p)^\top$
- then
$$\langle w, \phi(x) \rangle = w_0 + w_1 x + w_2 x^2 + \cdots + w_p x^p \,,$$
and we can find the best coefficients by linear regression
- `numpy.polyfit` → very handy when we want to fit univariate data

Polynomial regression, ctd.

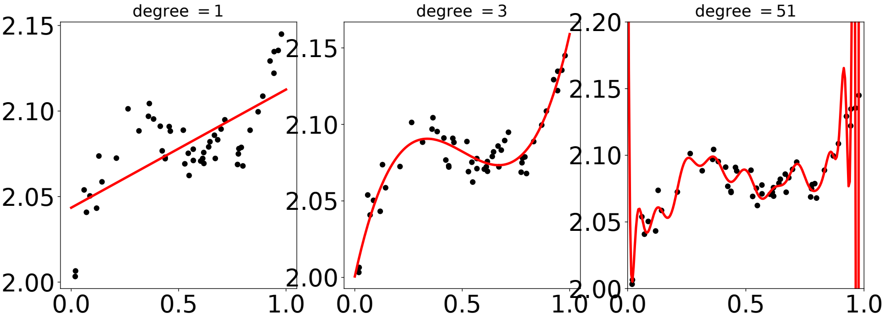
- **Example:** data = degree three polynomial + Gaussian noise with small variance
- fit a degree 3 polynomial:



- **Remark:** in practice, we do not know the degree of the polynomial!

Polynomial regression, ctd.

- typical case of under / overfitting:
  - when degree too low, poor fit
  - when degree too high, wiggly function ($n + 1 \Rightarrow$ interpolation)



1.5. Logistic regression

Logistic regression

- classification with $\mathcal{Y} = \{0, 1\}$
- however, we predict the probability of belonging to class 1
- hypothesis class:
$$\mathcal{H} = \{x \mapsto \phi(\langle w, x \rangle), w \in \mathbb{R}^d\} \, ,$$
with ϕ the *logistic function* (aka *sigmoid* function)
$$\phi(z) = \frac{1}{1 + \mathrm{e}^{-z}} \, .$$
- **Intuition:** squeeze the score between 0 and 1 to transform it into a probability
- $\mathbb{P}\left(y = 1 \,|\, x\right) = \phi(w^\top x)$ and $\mathbb{P}\left(y = 0 \,|\, x\right) = 1 - \phi(w^\top x)$

Logistic function

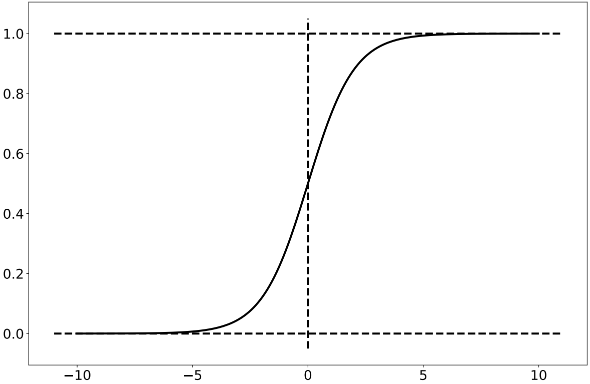
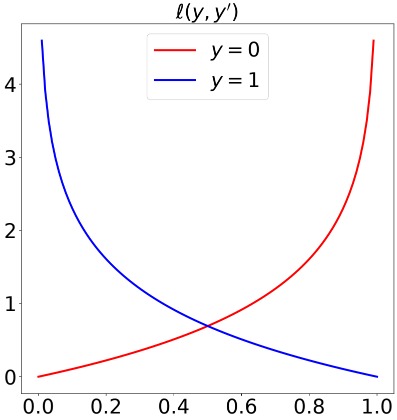


Figure: the logistic function $\phi : t \mapsto 1/(1+\mathrm{e}^{-t})$.

Logistic loss

- **Next:** we need to define a loss function
- for any y, y', we define the *logistic loss*:

$$\ell(y, y') = -(1-y)\log(1-y') - y\log y' \,.$$



Logistic regression

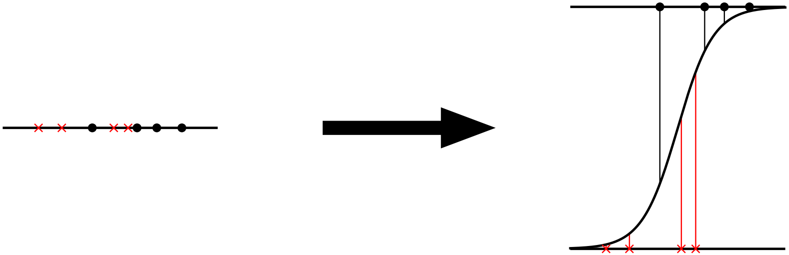
- finally, logistic regression = empirical risk minimization with the logistic loss
- that is, minimize for $w \in \mathbb{R}^d$

$$\hat{\mathcal{R}}_S(w) = \sum_{i=1}^{n} \left\{ -(1 - y_i) \log(1 - \phi(w^\top x_i)) - y_i \log \phi(w^\top x_i) \right\} .$$

- **Remark (i):** we can show that this is equivalent to maximum likelihood for a certain prior distribution
- **Remark (ii):** complicated to optimize (see exercise)

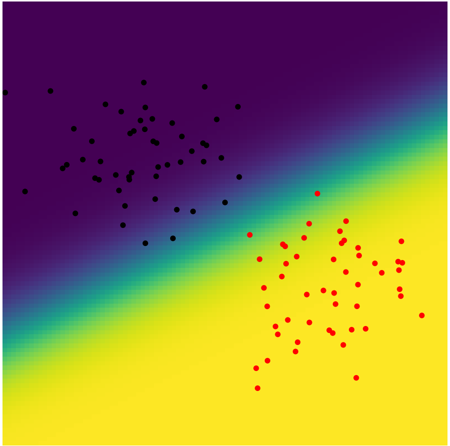
Logistic regression in dimension 1

- **Example:** in dimension one:



Logistic regression in dimension 2

- **Example:** in dimension two:



Exercise

Exercise: Recall that we defined the logistic loss by

$$\ell(y, y') = -(1-y)\log(1-y') - y\log y'\,.$$

1. Show that ERM with the logistic loss is equivalent to minimizing

   $$F(w) = \sum_{i=1}^{n} \log(1 + \exp\left(-\tilde{y}_i\langle w, x_i\rangle\right))\,,$$

   where $\tilde{y}_i = \operatorname{sign}\left(y_i - 0.5\right)$. Deduce that $\hat{\mathcal{R}}$ is a convex function of w.
2. Compute the gradient of $\hat{\mathcal{R}}$ with respect to w. *Hint:* show that $\phi'(z) = \phi(z)(1-\phi(z))$.
3. Can you solve $\nabla\hat{\mathcal{R}}(w) = 0$? If not, propose a strategy for finding a good w.

Correction of the exercise

1. Let us set $1 \leq i \leq n$. We write

$$\begin{aligned}\ell(y_i, \phi(w^\top x_i)) &= -(1-y_i)\log(1-\phi(w^\top x_i)) - y_i \log\phi(w^\top x_i)\\ &= -(1-y_i)\log\frac{\mathrm{e}^{-w^\top x_i}}{1+\mathrm{e}^{-w^\top x_i}} - y_i\log\frac{1}{1+\mathrm{e}^{-w^\top x_i}}\\ &= -(1-y_i)\log\mathrm{e}^{-w^\top x_i} + \log(1+\mathrm{e}^{-w^\top x_i})\,.\end{aligned}$$

If $y_i = 0$, the last display equals

$$\log(1+\exp\left(w^\top x_i\right))\,,$$

if $y_i = 1$, it is

$$\log(1+\exp\left(-w^\top x_i\right))\,.$$

One can check directly that $x \mapsto \log(1+\mathrm{e}^{-x})$ is convex. By composition, F is a sum of convex functions, thus convex.

Correction of the exercise, ctd.

2. Let $1 \leq j \leq d$. We write

$$\begin{aligned}
\frac{\partial \hat{\mathcal{R}}(w)}{\partial w_j} &= -\sum_{i=1}^{n} \frac{\partial}{\partial w_j} \left\{(1-y_i)\log(1-\phi(w^\top x_i)) + y_i \log \phi(w^\top x_i)\right\} \\
&= -\sum_{i=1}^{n} \left\{ \frac{-(1-y_i)}{1-\phi(w^\top x_i)} + \frac{y_i}{\phi(w^\top x_i)} \right\} \frac{\partial}{\partial w_j}\phi(w^\top x_i) \\
&= -\sum_{i=1}^{n} \left\{ \frac{-(1-y_i)}{1-\phi(w^\top x_i)} + \frac{y_i}{\phi(w^\top x_i)} \right\} \phi(w^\top x_i)(1-\phi(w^\top x_i))x_{i,j} \\
\frac{\partial \hat{\mathcal{R}}(w)}{\partial w_j} &= -\sum_{i=1}^{n} \left(y_i - \phi(w^\top x_i)\right) x_{i,j} \, .
\end{aligned}$$

3. It does not seem possible to solve $\nabla F(w) = 0$ in closed-form, one has to use gradient descent. □

Recap

- **What happens when we call** `sklearn.linear_model.LogisticRegression`?
- `penalty` is ℓ_2 → there is regularization by default! (not much though, $C = 1$)
- `fit_intercept` is True → again, our maths are not entirely accurate
- `solver` is liblinear → since there is no closed-form, a solver will be used
- liblinear uses coordinate descent
- will default soon to lbfgs (limited memory Broyden-Fletcher- -Goldfarb-Shanno)
- do not worry too much about the solvers, just change if you see that it is not converging

1.6. Support vector machines

Support vector machines

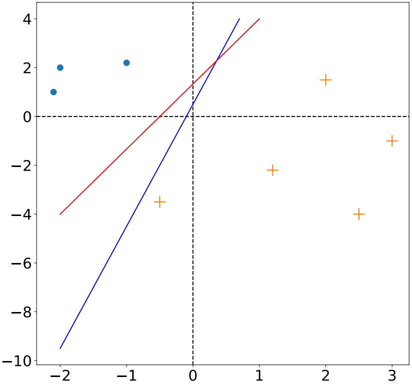
- classification with $x_1, \ldots, x_n \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$
- **Recall:** linearly separable means that there exist (w, b) such that

$$\forall i \in [n], \qquad y_i(w^\top x_i + b) > 0\,.$$

- **Remark:** all halfspaces satisfying this condition are empirical risk minimizers
- **Question:** which one should we pick?

Some intuition

- **Idea:** choose the one with maximum *margin*



- **Intuitively,** we would prefer the red line instead of the blue one

Margins

Definition: The margin of a hyperplane with respect to a training set is defined as the minimal distance between a point in the training set and the hyperplane.

- *Hard-SVM*[4] = minimizing empirical risk and choosing the max margin hyperplane
- **Question:** how to put this in equation?
- first, we need to express the distance between a point and a hyperplane:

Lemma: Assume that $\|w\| = 1$. Then the distance between x and the hyperplane defined by (w, b) is given by $\left|w^\top x + b\right|$.

[4] Boser, Guyon, Vapnik, *A training algorithm for optimal margin classifiers*, 5th workshop on computational learning theory, 1992

Proof of the lemma

- we want to compute
$$\min\{\|x - v\| \quad \text{s.t.} \quad w^\top v + b = 0\}\,.$$
- take $v = x - (w^\top x + b)w$:
$$w^\top v + b = w^\top x - (w^\top x + b)\,\|w\|^2 + b = 0\,,$$
since $\|w\| = 1$.
- moreover,
$$\|x - v\| = \left|w^\top x + b\right| \|w\| = \left|w^\top x + b\right|\,.$$
- for now, we have a point v on the hyperplane with distance $\left|w^\top x + b\right|$
- let us show that any other point has a larger distance

Proof of the lemma, ctd.

- let u such that $w^\top u + b = 0$, then

$$\begin{aligned}
\|x-u\|^2 &= \|x-v+v-u\|^2 \\
&= \|x-v\|^2 + \|v-u\|^2 + 2(x-v)^\top(v-u) \\
&\geq \|x-v\|^2 + 2(x-v)^\top(v-u) \\
&= \|x-v\|^2 + 2(w^\top x + b)w^\top(v-u)
\end{aligned}$$

- notice that $w^\top v = w^\top u = -b$, therefore

$$\|x-u\|^2 \geq \|x-v\|^2 \,.$$

$\square$

Hard-SVM rule

- **Consequence of the lemma:** the closest point in the training set has distance $\min_i \left|w^\top x_i + b\right|$ to the hyperplane
- we can rewrite the hard-SVM rule as

$$(\hat{w}, \widehat{b}) \in \underset{(w,b),\|w\|=1}{\arg\max} \; \min_i \left|w^\top x_i + b\right| \quad \text{s.t.} \quad y_i(w^\top x_i + b) > 0 \quad \forall i \,.$$

- **Intuition:** x_i on the right side of the hyperplane if y_i and $w^\top x_i + b$ have the same sign
- in the separable case, it is possible to show that an equivalent formulation is

$$(\hat{w}, \widehat{b}) \in \underset{(w,b),\|w\|=1}{\arg\max} \; \min_i y_i(w^\top x_i + b) \,.$$

Hard-SVM as quadratic programming

- as in the first linear example, possible to reframe as a standard optimization problem

Lemma: Let (w_0, b_0) be the solution of the following QP:

$$(w_0, b_0) \in \underset{(w,b)}{\arg\min} \left\|w\right\|^2 \qquad \text{s.t.} \qquad y_i(w^\top x_i + b) \geq 1 \quad \forall i \,.$$

Then $\hat{w} = w_0 / \left\|w_0\right\|$ and $\widehat{b} = b_0 / \left\|w_0\right\|$ satisfy the Hard-SVM rule.

- QP $=$ quadratic programming: objective is a quadratic function and the constraints are linear inequalities

Proof of the lemma

- let (w, b) be a solution of the Hard-SVM rule
- define the achieved margin by

$$\gamma = \min_i y_i(w^\top x_i + b)\,.$$

- by definition, for all $1 \le i \le n$, we have

$$y_i(w^\top x_i + b) \ge \gamma\,,$$

  that is

$$y_i\left(\left(\frac{w}{\gamma}\right)^\top x_i + \frac{b}{\gamma}\right) \ge 1\,.$$

- thus $(w/\gamma, b/\gamma)$ satisfies the condition of the QP

Proof of the lemma, ctd.

- in particular,

$$\|w_0\| \leq \left\| \frac{w}{\gamma} \right\| = \frac{1}{\gamma}\,.$$

- as a consequence, for all $1 \leq i \leq n$,

$$y_i(\hat{w}^\top x_i + \widehat{b}) = \frac{1}{\|w_0\|} \cdot y_i(w_0^\top x_i + b_0) \geq \frac{1}{\|w_0\|} \geq \gamma\,.$$

- since $\|\hat{w}\| = 1$, we have shown that $(\hat{w}, \widehat{b})$ is a solution of the Hard-SVM rule

□

The homogeneous case

- if we set $b = 0$, Hard-SVM rule becomes

$$\text{minimize}_w \left\|w\right\|^2 \quad \text{s.t.} \quad \forall 1 \leq i \leq n, \quad y_i w^\top x_i \geq 1\,.$$

- in that case, the solution w_0 is *supported* by the examples exactly at distance $1/\left\|w_0\right\|$ from the hyperplane

Theorem (Fritz John): Let $I := \{i \text{ s.t. } \left|w_0^\top x_i\right| = 1\}$. Then there exists coefficients α_i, $i \in I$, such that

$$w_0 = \sum_{i \in I} \alpha_i x_i\,.$$

- the $\{x_i, i \in I\}$ are called *support vectors*, hence the name

Soft-SVM

- linearly separable assumption is quite restrictive
- condition in the QP:
$$\forall 1 \leq i \leq n, \qquad y_i(w^\top x_i + b) \geq 1\,.$$
- **Natural relaxation:** allow this constraint to be violated for some points in the dataset
- we introduce *slack variables* $\xi_1, \ldots, \xi_n$ and replace the constraint by
$$\forall 1 \leq i \leq n, \qquad y_i(w^\top x_i + b) \geq 1 - \xi_i\,.$$
- **Intuition:** the ξ_i encode by how much the constraint is violated

Soft-SVM, ctd.

- **Key idea:** minimize jointly $\|w\|$ and the average of the ξ_i
- namely, the Soft-SVM rule is

$$\text{minimize}_{w,b,\xi} \left\{ \lambda \|w\|^2 + \frac{1}{n} \sum_{i=1}^{n} \xi_i \right\} \quad \text{s.t.} \quad \forall 1 \leq i \leq n, \quad y_i(w^\top x_i + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0\,,$$

where $\lambda > 0$ is a fixed hyperparameter

Soft-SVM as regularized ERM

- recall the definition of the *hinge loss*:

$$\ell(y, y') = \max\{0, 1 - yy'\} \, .$$

- we have the remarkable result:

Lemma: Consider

$$(\hat{w}, \widehat{b}) \in \operatorname*{arg\,min}_{(w,b)} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, w^\top x_i + b) + \lambda \left\| w \right\|^2 \right\} \, .$$

Then $(\hat{w}, \widehat{b})$ satisfies the Soft-SVM rule.

Proof of the lemma

- recall that we want to minimize

$$\lambda \|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i$$

  subject to

$$\forall 1 \leq i \leq n, \quad y_i(w^\top x_i + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0\,.$$

- fix w, b and let us try and minimize in ξ
- let us fix some i: since $\xi_i \geq 0$, the best assignment would be $\xi_i = 0$ if $y_i(w^\top x_i + b) \geq 1$, and $1 - y_i(w^\top x_i + b)$ otherwise
- in other words,

$$\forall 1 \leq i \leq n, \quad \xi_i = \max\{0, 1 - y_i(w^\top x_i + b)\} = \ell(y_i, w^\top x_i + b)\,.$$

□

Recap

- scikit-learn implementation: `sklearn.svm.LinearSVC`
- then a solver is called, similarly to the logistic regression case
- default is ℓ_2 regularization (as we have seen)
- regularization is given by $C = 1$ ($C = 1/\lambda$, beware!)
- we will see an extension of SVM when we look into kernel methods

2. Kernel methods

2.1. Positive semi-definite kernels

Representation of the data

- **What we have seen so far:** linear classification / linear regression
- works well if the data is linearly separable
- **Problem:** that is not always the case!
- what if we could transport the data to another space where it is well-behaved?
- for instance a very high-dimensional space
- first we define a *kernel*

Positive semi-definite kernels

Definition: a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a positive definite kernel if $k(x, x') = k(x', x)$ for any $x, x' \in \mathcal{X}$, and

$$\forall x_1, \ldots, x_n \in \mathcal{X}, \forall c_1, \ldots, c_n \in \mathbb{R}, \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0 \,.$$

- unlike the name suggests, k has no reason to be *positive*
- in other words, the Gram matrix $K = (k(x_i, x_j))_{i,j=1}^{n}$ is positive definite for any input data $x_1, \ldots, x_n$
- *kernel methods* take this K as input
- **Remark:** this is *costly*, $\mathcal{O}\left(n^2\right)$ whatever we do, with possible dependency in the dimensionality of the data

Fundamental example

- suppose that $\mathcal{X} = \mathbb{R}$
- then $k(x, y) := xy$ is a positive definite kernel
- **Why?** first, we check that $k(x, y) = k(y, x)$
- second, let $n \geq 1$, $x_1, \ldots, x_n \in \mathbb{R}^d$, and $c_1, \ldots, c_n \in \mathbb{R}$, then

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j k(x_i, x_j) &= \sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j x_i x_j \\ &= \left(\sum_{i=1}^{n} c_i x_i\right)^2 \\ &\geq 0\,.\end{aligned}$$

Fundamental example, ctd.

- we can extend this example:
- suppose that $\mathcal{X} = \mathbb{R}^d$
- let $n \geq 1$, $x_1, \ldots, x_n \in \mathbb{R}^d$, and $c_1, \ldots, c_n \in \mathbb{R}$, then

$$\begin{aligned} \sum_{i=1}^n \sum_{j=1}^n c_i c_j k(x_i, x_j) &= \sum_{i=1}^n \sum_{j=1}^n c_i c_j x_i^\top x_j \\ &= \left\| \sum_{i=1}^n c_i x_i \right\|^2 \\ &\geq 0 \,. \end{aligned}$$

- $k(x, y) := x^\top y$ is usually called the **linear kernel**
- **Intuition:** kernels are a generalization of inner product

Other examples

- **Polynomial kernel:**

$$\mathcal{X} = \mathbb{R}^d, \qquad k(x,y) = (x^\top y + c)^k\,.$$

- **min kernel:**

$$\mathcal{X} = \mathbb{R}, \qquad k(x,y) = \min(x,y)\,.$$

- **Gaussian kernel:**

$$\mathcal{X} = \mathbb{R}^d, \qquad k(x,y) = \exp\left(\frac{-\|x-y\|^2}{2\nu^2}\right)\,.$$

- **Exponential kernel:**

$$\mathcal{X} = \mathbb{R}^d, \qquad k(x,y) = \exp\left(\frac{-\|x-y\|}{2\nu}\right)\,.$$

- ...

Choosing the bandwidth

- Gaussian and Laplace kernel: one has to choose the bandwidth parameter ν
- indeed, if ν is *too large* with respect to the typical value of $\|x_i - x_j\|$, then $K \approx I_n$
- in the other direction, if ν is *too small*, then $K \approx \mathbf{1}\mathbf{1}^\top$
- both cases are degenerate: whatever we do with K is not going to work very well
- one possible solution: **median heuristic**[5]

$$\nu = \mathrm{Med}\{\|x_i - x_j\|\,, \quad 1 \leq i,j \leq n\}\,.$$

- preferable to the mean (too sensitive to extreme values)
- we can pick other quantiles

[5] Garreau, Jitkrittum, Kanagawa, *Large sample analysis of the median heuristic*, 2017

Exercise

Exercise: Show that the following functions are positive definite kernels:

1. $\mathcal{X} = \mathbb{N}, \qquad k(x, y) = 2^{x+y}$
2. $\mathcal{X} = \mathbb{R}, \qquad k(x, y) = \cos(x - y)$
3. $\mathcal{X} = \mathbb{R}^d, \qquad k(x, y) = (x^\top y)^2$

Correction of the exercise

1. let $x_1, \dots, x_n \in \mathbb{N}$ and $c_1, \dots, c_n \in \mathbb{R}$, then

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j 2^{x_i + x_j} = \left(\sum_{i=1}^{n} c_i 2^{x_i} \right) \geq 0 \,.$$

2. let $x_1, \dots, x_n \in \mathbb{N}$ and $c_1, \dots, c_n \in \mathbb{R}$, then

$$\begin{aligned}
\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \cos(x_i - x_j) &= \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \left(\cos x_i \cos x_j + \sin x_i \sin x_j \right) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \cos x_i \cos x_j + \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \sin x_i \sin x_j \\
&= \left(\sum_{i=1}^{n} c_i \cos x_i \right)^2 + \left(\sum_{i=1}^{n} c_i \sin x_i \right)^2 \,.
\end{aligned}$$

Correction of the exercise, ctd.

3. let $x, y \in \mathbb{R}^d$:

$$\begin{aligned}(x^\top y)^2 &= \mathrm{trace}\left(x^\top y x^\top y\right) \\ &= \mathrm{trace}\left(y^\top x x^\top y\right) \\ &= \mathrm{trace}\left(x x^\top y y^\top\right) \ .\end{aligned}$$

We recognize the inner product between matrices.

Important remark

- recall the linear kernel: as in the exercise, all we used were properties of inner product
- let $\Phi : \mathcal{X} \to \mathcal{H}$ be some mapping, $\mathcal{H}$ a Hilbert space with scalar product $\langle \cdot, \cdot \rangle$
- then $k(x, y) = \langle \Phi(x), \Phi(y) \rangle$ is positive definite:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \langle \Phi(x), \Phi(y) \rangle = \left\| \sum_{i=1}^{n} c_i \Phi(x_i) \right\|^2 \geq 0\,,$$

by linearity of the inner product.

Kernel as inner products

- **Remarkable fact:** the converse statement is true!

Theorem (Aronszajn, 1950): For any kernel k on $\mathcal{X}$, there exists a Hilbert space $(\mathcal{H}, \langle \cdot, \cdot \rangle)$ and a mapping $\phi : \mathcal{X} \to \mathcal{H}$ such that

$$\forall x, y \in \mathcal{X}, \qquad k(x,y) = \langle \Phi(x), \Phi(y) \rangle \,.$$

- **Reminder:** Hilbert space $=$ inner product $+$ *complete* for the associated norm (Cauchy sequences converge in $\mathcal{H}$)
- not so important for us, needed for good convergence properties
- **Consequence:** we can think of any kernel as a dot product in the *feature space*

Proof in the finite case

- assume that $\mathcal{X} = \{x_1, \dots, x_N\}$ is finite of size N
- any kernel k is entirely defined by the $N \times N$ positive semi-definite matrix $K := (k(x_i, x_j))_{i,j=1}^{N}$
- we can diagonalize K in an orthonormal basis $(u_1, \dots, u_N)$ with associated (non-negative) eigenvalues $\lambda_1, \dots, \lambda_N$
- then we write

$$
\begin{aligned}
k(x_i, x_j) &= \left(\sum_{\ell=1}^{N} \lambda_\ell u_\ell u_\ell^\top \right)_{i,j} \\
&= \sum_{\ell=1}^{N} \lambda_\ell (u_\ell)_i (u_\ell)_j = \langle \Phi(x_i), \Phi(x_j) \rangle \,,
\end{aligned}
$$

with

$$
\Phi(x_i) := \left(\sqrt{\lambda_1}(u_1)_i \cdots \sqrt{\lambda_n}(u_N)_i \right)^\top \,.
$$

2.2. Reproducing kernel Hilbert spaces

Function spaces

- among all spaces in the previous statement, one of them has interesting properties
- in particular, it is a **space of functions**
- *i.e.*, we can map each point $x \in \mathcal{X}$ to a *function* $\Phi(x) = k_x \in \mathcal{H}$
- **Example:** $\mathcal{X} = \mathbb{R}$, we map each x to the function $t \mapsto xt$
- $\rightarrow$ space of linear functions
- more complicated in general...

RKHS

Definition: let $\mathcal{X}$ be a set and $\mathcal{H}$ be a function space forming a Hilbert space with inner product $\langle \cdot, \cdot \rangle$. The function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a *reproducing kernel* of $\mathcal{H}$ if

- $\mathcal{H}$ contains all functions of the form $k_x : t \mapsto k(x, t)$
- for every $x \in \mathcal{X}$ and $f \in \mathcal{H}$, the *reproducing property* holds:

$$f(x) = \langle f, k_x \rangle .$$

- if a reproducing kernel exists, then $\mathcal{H}$ is called a *reproducing kernel Hilbert space* (RKHS)

Equivalent definition

Theorem: the Hilbert space $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{X}}$ is a RKHS if, and only if, for any $x \in \mathcal{X}$, the mapping $f \mapsto f(x)$ is continuous.

- *Proof:* $\Rightarrow$ if a reproducing kernel k exists, then

$$|f(x)| = |\langle f, k_x \rangle| \leq \|f\| \cdot \|k_x\|$$

  by Cauchy-Schwarz.
- we see that $\|k_x\|^2 = \langle k_x, k_x \rangle = k(x, x)$, thus $|f(x)| \leq \|f\| \cdot k(x, x)^{1/2}$
- thus $f \mapsto f(x)$ is continuous.
- $\Leftarrow$ Riesz theorem. □

Important properties

Theorem (uniqueness): if $\mathcal{H}$ is a RKHS, then it has a unique reproducing kernel. Conversely, a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ can be the reproducing kernel of at most one RKHS.

- this is why we talk about *the* RKHS associated to k

Theorem: a function $k : \mathcal{X} \times \mathcal{X}$ is positive definite if, and only if, it is a reproducing kernel.

- showing that a kernel is positive definite is enough to get Φ and $\mathcal{H}$ with the reproducing property "for free"

Example

- **Example:** polynomial kernel of degree 2:

$$k(x, y) = (x^\top y)^2 \,.$$

- according to the exercise,

$$k(x, y) = \langle xx^\top, yy^\top \rangle_F \,,$$

  we have proven that k is positive definite

- **Question:** what is the RKHS?
- we know that $\mathcal{H}$ contains all the functions

$$f(x) = \sum_i a_i k(x_i, x) = \sum_i a_i \langle x_i x_i^\top, xx^\top \rangle = \langle \sum_i a_i x_i x_i^\top, xx^\top \rangle$$

Example, ctd.

- spectral theorem: any symmetric matrix can be decomposed as $\sum_i a_i x_i x_i^\top$
- candidate RKHS: set a quadratic functions

$$f_S(x) = \langle S, xx^\top \rangle = x^\top S x \,,$$

  with S symmetric matrix of size $d \times d$
- inner product on $\mathcal{H}$:

$$\langle f_S, f_{S'} \rangle = \langle S, S' \rangle_F \,.$$

- we can check that $\mathcal{H}$ is a Hilbert space (isomorphic to $\mathcal{S}^{d \times d}$)
- finally, we check the reproducing property

2.3. More examples

Elementary properties

Proposition: Let $k_i : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a (potentially infinite) family of p.d. kernels. Then

- for any $\lambda_1, \ldots, \lambda_p \geq 0$, the sum $\sum_{i=1}^{p} \lambda_i k_i$ is positive definite
- for any $a_1, \ldots, a_p \in \mathbb{N}$, the product $k_1^{a_1} \cdots k_p^{a_p}$ is positive definite
- if it exists, the limit $k = \lim_{p \to +\infty} k_p$ is positive definite

Moreover, let $\mathcal{X}_i$ be a sequence of sets and k_i positive kernels on each $\mathcal{X}_i$. Then

$$k((x_1, \ldots, x_p), (y_1, \ldots, y_p)) := \prod_{i=1}^{p} k_i(x_i, y_i)$$

and

$$k((x_1, \ldots, x_p), (y_1, \ldots, y_p)) := \sum_{i=1}^{p} k_i(x_i, y_i)$$

are positive definite kernels.

Taking the exponential

Theorem: if k is a positive definite kernel, then e^k as well.

- *Proof:* we write

$$\mathrm{e}^{k(x,y)} = \lim_{n \to +\infty} \sum_{p=0}^{n} \frac{k(x,y)^p}{p!} ,$$

  then reason step by step.
- by the product property, $k(x,y)^p$ is a kernel for any $p \geq 0$
- as a positive linear combination of kernels, $\sum_{p=0}^{n} \frac{k(x,y)^p}{p!}$ is a kernel for all $n \geq 1$
- finally, e^k is a kernel as a limit of kernels. □

Exercise

Exercise: show that the Gaussian kernel k is positive definite, that is,

$$k(x, y) := \exp\left(\frac{-\|x - y\|^2}{2\nu^2}\right) ,$$

where $\nu > 0$ is a fixed *bandwidth* parameter. *Hint:* decompose the squared norm and use the propeerties.

Correction of the exercise

- first recall that
$$\|x-y\|^2 = \|x\|^2 - 2x^\top y + \|y\|^2 \, .$$
- we split the kernel in two parts:
$$k(x,y) = \mathrm{e}^{-\|x\|^2-\|y\|^2} \cdot \exp\left(2x^\top y\right) \, .$$
- the first part is a kernel (scalar product of two feature maps)
- the second part as well (exponential of a kernel)
- since the product of kernels is a kernel, we can conclude.

2.4. The kernel trick and applications

The kernel trick

- input data $x_1, \dots, x_n \mathcal{X}$
- $k : \mathcal{X} \times \mathcal{X}$ kernel with associated RKHS $\mathcal{H}$
- we call $\Phi : \mathcal{X} \to \mathcal{H}$ the feature map
- **Idea:** imagine that our algorithm only depends on scalar products $x_i^\top x_j$
- then we can map the x_i to $\mathcal{H}$ and replace the inner products by kernel evaluations, since

$$\langle \Phi(x_i), \Phi(x_j) \rangle = k(x_i, x_j) \, .$$

- simple “trick” with many, many applications

Example

- **Example:** computing distances
- suppose that our algo relies on distance computation
- that is, $\|x - y\|^2$
- we can write

$$\begin{aligned}\|\Phi(x) - \Phi(y)\|^2 &= \langle \Phi(x) - \Phi(y), \Phi(x) - \Phi(y) \rangle \\ &= \langle \Phi(x), \Phi(x) \rangle - 2\langle \Phi(x), \Phi(y) \rangle + \langle \Phi(y), \Phi(y) \rangle \\ \|\Phi(x) - \Phi(y)\|^2 &= k(x,x) - 2k(x,y) + k(y,y)\,.\end{aligned}$$

- in other words,

$$d_{\mathcal{H}}(x,y) = \sqrt{k(x,x) - 2k(x,y) + k(y,y)}\,.$$

- as promised, **we do not need to know Φ!**

Exercise

Exercise: let $S = \{x_1, \ldots, x_n\}$ be a finite set of points in $\mathcal{X}$. Compute the distance to the barycenter of S in the RKHS.

Correction

- first we see that the barycenter is
$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \,.$$
- **Beware:** there is not necessarily a point $m \in \mathcal{X}$ such that $\Phi(m) = \mu$
- then we proceed as before:

$$\begin{aligned} d_{\mathcal{H}}(x, S)^2 &= \left\|x - \mu\right\|^2 \\ &= k(x, x) - \frac{2}{n}\sum_{i=1}^{n} k(x, x_i) + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} k(x_i, x_j) \,. \end{aligned}$$

2.5. The representer theorem

Motivation

- let us imagine that we take $\mathcal{H}$ as hypothesis class
- starting from regularized ERM, our optimization problem will look like

$$\operatorname*{arg\,min}_{f\in\mathcal{H}} \left\{ \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f(x_i)) + \lambda \left\| f \right\|^2 \right\} . \tag{$\star$}$$

- we penalize by the norm because it is an indicator of the *smoothness* of f
- **Why?** Cauchy-Schwarz + exercise:

$$|f(x) - f(y)| = |\langle f, k_x - k_y \rangle| \leq \|f\| \cdot \|k_x - k_y\| = \|f\| \cdot d_{\mathcal{H}}(x, y) .$$

- Eq. $(\star)$ is a complicate problem, potentially *infinite-dimensional*
- **Question:** how to solve it in practice?

The representer theorem

Theorem: let $\mathcal{H}$ be the RKHS associated to k defined on $\mathcal{X}$. Let $S = \{x_1, \ldots, x_n\} \subseteq \mathcal{X}$ be a finite set of points. Let $\Psi : \mathbb{R}^{n+1} \to \mathbb{R}$ be a function, increasing in the last variable. Then any solution to the minimization problem

$$\arg\min_{f \in \mathcal{H}} \Psi(f(x_1), \ldots, f(x_n), \|f\|)$$

admits a representation of the form

$$\forall x \in \mathcal{X}, \qquad f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x) \,.$$

- **Main consequence:** Eq. $(\star)$ is actually a finite-dimensional problem (!)

Practical use

- recall that we defined $K := (k(x_i, x_j))_{i,j=1}^n$
- before turning to concrete examples, we notice that we can simply express the key quantities
- for instance, for any $1 \leq j \leq n$,

$$f(x_j) = \sum_{i=1}^n \alpha_i k(x_i, x_j) = (K\alpha)_j \, .$$

- in the same way,

$$\left\|f\right\|^2 = \left\|\sum_{i=1}^n \alpha_i k(x_i, \cdot)\right\| = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j k(x_i, x_j) = \alpha^\top K\alpha \, .$$

2.6. Kernel ridge regression

Kernel Ridge Regression[6] (KRR)

- regression setting: $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$
- $\mathcal{Y} \subseteq \mathbb{R}$, but $\mathcal{X}$ could be anything
- we have a kernel k on $\mathcal{X}$
- same idea than with ridge regression:

$$\hat{f} \in \operatorname*{arg\,min}_{f \in \mathcal{H}} \left\{ \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \left\| f \right\|^2 \right\} .$$

- here effect of the regularization is to make $\hat{f}$ smoother

[6] Cristianini and Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*, Cambridge University Press, 2000

Solving KRR

- representer theorem $\Rightarrow$

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x)\,,$$

  for some $\alpha \in \mathbb{R}^n$
- as per the previous remark, we know that

$$(\hat{f}(x_1), \ldots, \hat{f}(x_n))^\top = K\alpha\,,$$

  and

$$\left\|\hat{f}\right\|^2 = \alpha^\top K \alpha\,.$$

- thus KRR can be re-written as

$$\hat{\alpha} \in \operatorname*{arg\,min}_{\alpha \in \mathbb{R}} \left\{ \frac{1}{n}(y - K\alpha)^\top (y - K\alpha) + \lambda \alpha^\top K \alpha \right\}\,.$$

Solving KRR, ctd.

- convex, smooth objective ⇒ set the gradient to zero
- $\hat{\alpha}$ has to be solution of

$$0 = \frac{-2}{n} K(y - K\alpha) + 2\lambda K\alpha = \frac{2}{n} K\left[(K + n\lambda I_n)\alpha - y\right]$$

- since $\lambda > 0$, $K + n\lambda I_n$ is invertible
- the solution is given by

$$\hat{\alpha} = (K + n\lambda I_n)^{-1} y\,.$$

- **Question:** what do you think of the uniqueness of the solution?

Exercise

Exercise: show that, when the kernel is linear, the solution given by the kernel ridge regression approach and the vanilla ridge regression coincide.
Hint: show that for any matrices A, B,

$$A(BA + \gamma I)^{-1} = (AB + \gamma I)^{-1} A\,,$$

whenever the inverses are well-defined.

Correction

- for the linear kernel, $K = X^\top X$
- proof of the hint: assume that the inverse exist
- then write

$$A(I + BA) = (I + AB)A\,.$$

- finally, (right) multiply by $(I + BA)^{-1}$ and (left) multiply by $(I + AB)^{-1}$ both sides

3. Tree-based classifiers

3.1. Partition rules

Introduction

- let $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}$
- in this section, we consider partition-based classifiers:

$$\mathcal{H} = \left\{ h : x \mapsto \sum_{j=1}^{p} h_j \mathbb{1}_{x \in A_j} \right\} ,$$

  where $a_j \in \mathbb{R}$ and $A_1, \ldots, A_p$ form a *partition* of the space
- that is,

$$A_1 \cup \cdots \cup A_p = \mathcal{X} \quad \text{and} \quad A_i \cap A_j = \emptyset \forall i \neq j \, .$$

- the A_js are often called *cells*
- generally, for practical reasons the A_js are rectangles

ERM for partition rules

- assume that the partition is fixed and set $A(x) =$ cell containing x
- **Regression:** with squared loss, ERM rule gives

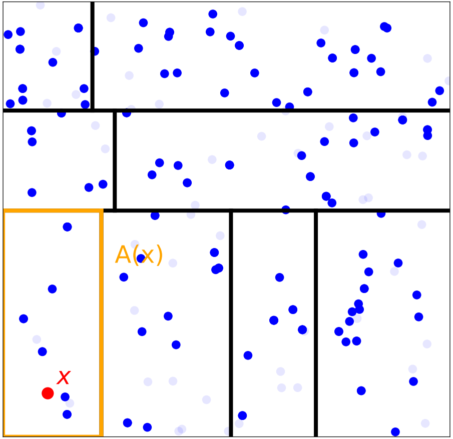
$$f(x) = \frac{1}{|\{j \text{ s.t. } x_j \in A(x)\}|} \sum_{i=1}^{n} x_i \mathbb{1}_{y_i \in A(x)} \, ,$$

  that is the average of the observations on each cell
- **Classification:** majority vote:

$$f(x) = \begin{cases} 1 & \text{if } |\{i \text{ s.t. } x_i \in A(x) \text{ and } y_i = 1\}| \geq \\ & \qquad |\{i \text{ s.t. } x_i \in A(x) \text{ and } y_i = 0\}| \\ 0 & \text{otherwise.} \end{cases}$$

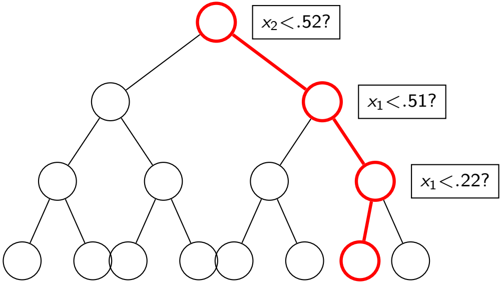
- thus ERM ⇔ finding the best partition (for a fixed p)
- **Problem:** this is computationally very hard! p^n possibilities to compare
- even if we restrict ourselves to rectangles, intractable

Example of a partition-based predictor

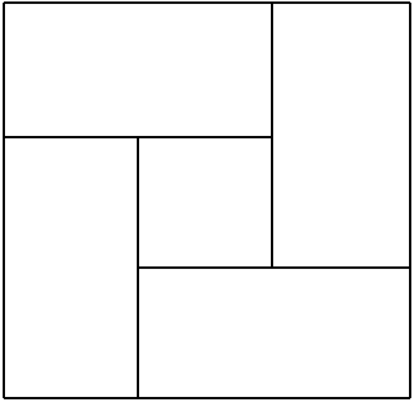


Tree structures

- one possible solution: start from $\mathcal{X}$ and *split* the cells iteratively
- we obtain a tree-like structure
- **Remark:** not necessary to split in two, but very common
- another advantage in doing so: root the new data efficiently



Not a tree



- **Figure:** this partition of $[0,1]^2$ can not be obtained by recursive binary splitting

Growing trees

- **Question:** how do we make the splits?
- **general answer:** take an heuristic that makes sense
- each heuristic yields a different algorithm, completed with stopping criterion (do a split only if gain greater than γ)
- complete reference on such procedures: the *yellow book*[7]
- good splitting rules:
  - create many cells (enough to capture the local variations of the distribution);
  - create cells that are large enough (we want sufficiently training data in the cells to compute a relevant average)
- **Notation:** I current node, I_L (resp. I_R) left (resp. right) node after the split

[7] Devroye, Györfi, Lugosi, *A probabilistic theory of pattern recognition*, 1996

ID3[9] and C4.5

Definition: Let S be a finite set of points. Then we define the *entropy* of S by

$$H(S) = \sum_{y \in \mathcal{Y}} -p(y) \log_2 p(y) \,,$$

where $p(y)$ is the proportion of elements of S classified as y.

- easy to see that $H(S) = 0$ means that the node is *pure* = only one label ($0 \log 0 = 0$)
- C4.5 criterion:[8] find direction and split that maximizes

$$H(I) - H(I_L) - H(I_R) \,.$$

[8] Quinlan, *C4.5: Programs for Machine Learning*, 1993

[9] Quinlan, *Induction of decision trees*, Machine Learning, 1986

CART trees, classification

- later supplanted by CART trees[10]

Definition: Let S be a finite set of points. We define the *Gini impurity* by

$$G(S) = \sum_{y \in \mathcal{Y}} p(y)(1 - p(y))\,.$$

- $G(S) = 0$ if the leaf is pure
- CART trees: find direction and split that maximizes

$$G(I) - G(I_L) - G(I_R)\,.$$

[10] Breiman et al., *Classification and Regression Trees*, 1984

CART trees for regression

- slightly different in the regression setting: look at the *variance*

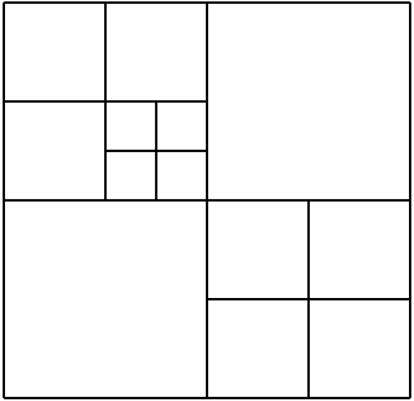
$$V(S) = \frac{1}{|S|}\sum_{i\in S}(y_i - \overline{y}_S)^2 = \frac{1}{|S|^2}\sum_{i,j\in S}\frac{1}{2}(y_i - y_j)^2 \,.$$

- the criterion is the variance reduction due to the split:

$$V(I) - \frac{|I_L|^2}{|I|^2}V(I_L) - \frac{|I_R|^2}{|I|^2}V(I_R) \,.$$

- **Intuition:** maximal if data is homogeneous left and right of the split (then $V(I_L) = V(I_R) = 0$)

Other examples



- **Figure:** quad trees[11]

[11] Finkel, Bentley, *Quad trees: a data structure for retrieval on composite keys*, Acta Informatica, 1974

Other examples, ctd.

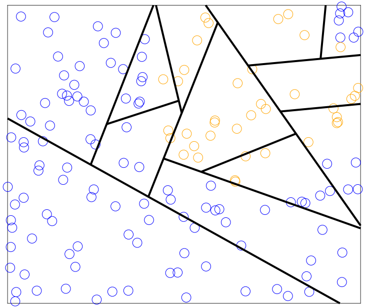


Figure: comparison-based splits[12]

[12] Haghiri et al., *Comparison-based random forests*, ICML, 2018

When to stop?

- usually, many direction to try: CART reduces to a random subset of directions
- also possible to specify T a max height for the tree
- other strategy: grow the trees to the full extent, and then **pruning**
- one possibility = reduced error pruning
- starting at the leaves, each node replaced by its most common class
- if prediction accuracy does not change, ditch the node
- **Remark:** error computations on a *validation set*

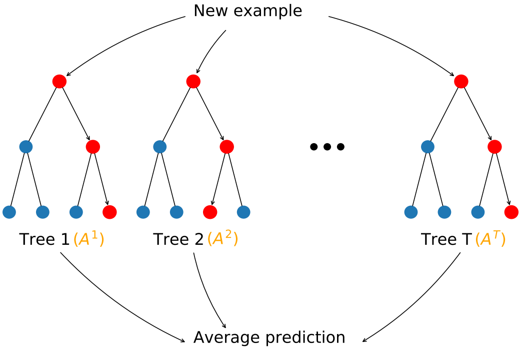
Recap

- **What happens by default when we invoke the function** `sklearn.tree.DecisionTreeClassifier`? let us look at least at the main options
- `criterion` is set to Gini → we are using CART trees
- `splitter` is set to best → looking at the best split at each step
- `max_depth` is None → splitting until leaves are pure or contain less than `min_samples_split`
- `min_samples_split` $= 2$
- `max_features` is None → no max number of features, log could be a reasonable choice if we have many features
- `max_leaf_nodes`: None → many leaves, we could also restrict this
- `min_impurity_decrease` $= 0$ → continues to split even if very small gain

3.2. Random forests

Random forests

- one possible problem with tree classifiers: overfitting
- **Solution:** train many trees and aggregate the prediction
- **Classification:** majority vote
- **Regression:** return the mean



Bagging

- **Additional idea:**[13] train each tree on a random subsample of the data
- usual strategy = bagging
- bagging means bootstrap aggregation
- sample *with replacement* a proportion αn of the training data
- train the tree classifier on this subset
- resample for each tree

[13] Breiman, *Random forests*, Machine Learning, 2001

Recap

What happens by default when we invoke the function `sklearn.ensemble.RandomForestClassifier`?

- `n_estimators` = 100 (T in our notation)
- `criterion` = 'Gini' → we are using CART trees
- `max_depth` = None → trees are grown until leaves are pure
- `max_features` = auto → $\sqrt{d}$ features considered
- `bootstrap` = True → taking subsamples of the data, but since `max_samples` is set to None actually sampling the whole data

4. Boosting

Introduction

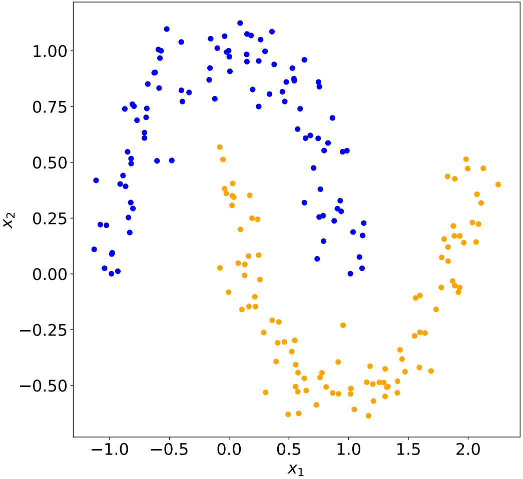
- **Important:** classification setting, $\mathcal{Y} = \{-1, +1\}$
- **Idea:** aggregate many classifiers together, then majority voting:

$$H(x) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) ,$$

  where h_t are classifiers and α_t weights
- weak classifier = barely better than random guessing
- **Examples:** linear classifier, small trees,...
- **Question:** how do we decide which weights to put?
- different strategies, different algorithms[14]

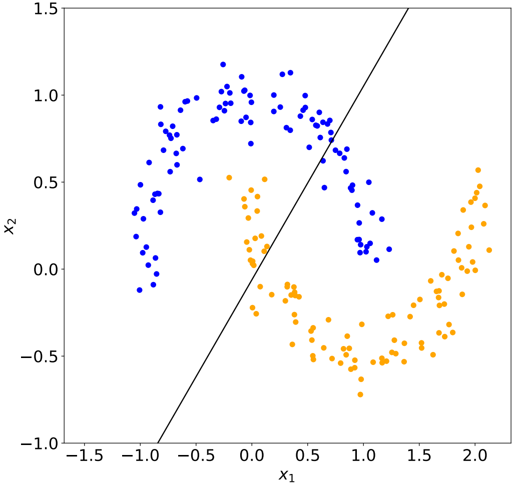
[14] Schapire, Freund, *Boosting: foundations and algorithms*, MIT Press, 2012

Non-linearly separable datasets



- **Figure:** moons datasets from `sklearn`

Weak classifier



- **Figure:** moons datasets from `sklearn`

4.1. Adaboost

Introduction

- we first look at AdaBoost[15] *(short for adaptative boosting)*
- AdaBoost maintains a *distribution* D_t over time
- start with uniform distribution, then increase the weights of misclassified examples
- at each step, we pick a classifier that minimizes the *weighted error*

$$\begin{aligned} \varepsilon_t &:= \mathbb{P}_{i\sim D_t}\left(h_t(x_i) \neq y_i\right) \\ &= \sum_{i=1}^{n} D_t(i) \cdot \mathbb{1}_{h_t(x_i)\neq y_i} \,. \end{aligned}$$

- adjust the weights by multiplying by a quantity depending on ε_t, larger than one if misclassified, smaller if correctly classified

[15] Freund and Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*, Journal of computer and system science, 1997

AdaBoost

```
Algorithm 1: AdaBoost algorithm
Input: $n$ training examples $(x_1, y_1), \dots, (x_n, y_n)$ where $x_i \in \mathcal{X}$ and $y_i \in \{-1, 1\}$
Initialize the distribution to $D_1(i) = \frac{1}{n}$
for $t = 1$ to $T$ do
    Train weak learner using distribution $D_t$
    Get weak hypothesis $h_t : \mathcal{X} \to \{-1, 1\}$
    $h_t$ minimizes the weighed error $\varepsilon_t := \mathbb{P}_{i \sim D_t}\left(h_t(x_i) \neq y_i\right)$ . Set $\alpha_t := \frac{1}{2} \log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
    Update, for $i = 1 \dots n$,
        $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \mathrm{e}^{-\alpha_t} & \text{if } h_t(x_i) = y_i \,, \\ \mathrm{e}^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \,. \end{cases}$
end
Result: final classifier $H(x) := \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$.
```

Exercise

Exercise: With the notation of the previous slide,

1. show that
$$\mathbb{P}_{i \sim D_{t+1}}\left(h_t(x_i) \neq y_i\right) = \frac{\sqrt{\varepsilon_t(1-\varepsilon_t)}}{Z_t}\,.$$
2. show that
$$Z_t = 2\sqrt{\varepsilon_t(1-\varepsilon_t)}\,.$$
   Deduce that
$$\mathbb{P}_{i \sim D_{t+1}}\left(h_t(x_i) \neq y_i\right) = \frac{1}{2}\,.$$

Correction of the exercise

1. We write

$$\begin{aligned}
\mathbb{P}_{i\sim D_{t+1}}\left(h_t(x_i)\neq y_i\right) &= \sum_{i=1}^{n} \mathbb{1}_{h_t(x_i)\neq y_i}\cdot D_{t+1}(i) && \text{(total expectation)}\\
&= \sum_{i=1}^{n} \mathbb{1}_{h_t(x_i)\neq y_i}\cdot \frac{D_t(i)}{Z_t}\cdot \mathrm{e}^{\alpha_t} && \text{(definition of } D_{t+1}\text{)}\\
&= \frac{1}{Z_t}\sum_{i=1}^{n} \mathbb{1}_{h_t(x_i)\neq y_i}\cdot D_t(i)\cdot \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} && \text{(definition of } \alpha_t\text{)}\\
&= \frac{1}{Z_t}\cdot\sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}}\cdot\varepsilon_t = \frac{\sqrt{\varepsilon_t(1-\varepsilon_t)}}{Z_t}\,.
\end{aligned}$$

Correction of the exercise, ctd.

2.

$$
\begin{aligned}
Z_t &= \sum_{\substack{i=1\\ h_t(x_i)\neq y_i}}^{n} D_t(i)\cdot \mathrm{e}^{\alpha_t} + \sum_{\substack{i=1\\ h_t(x_i)=y_i}}^{n} D_t(i)\cdot \mathrm{e}^{-\alpha_t} \\
&= \mathrm{e}^{\alpha_t}\sum_{i=1}^{n} \mathbb{1}_{h_t(x_i)\neq y_i}\cdot D_t(i) + \mathrm{e}^{-\alpha_t}\left(1-\sum_{i=1}^{n} \mathbb{1}_{h_t(x_i)\neq y_i}\cdot D_t(i)\right) \\
&= \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}}\cdot\varepsilon_t + \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}}\cdot(1-\varepsilon_t) \\
Z_t &= 2\sqrt{\varepsilon_t(1-\varepsilon_t)}\,.
\end{aligned}
$$

□