Theory of Statistical Learning
Part II

Damien Garreau

Université Côte d'Azur

2022-2023

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

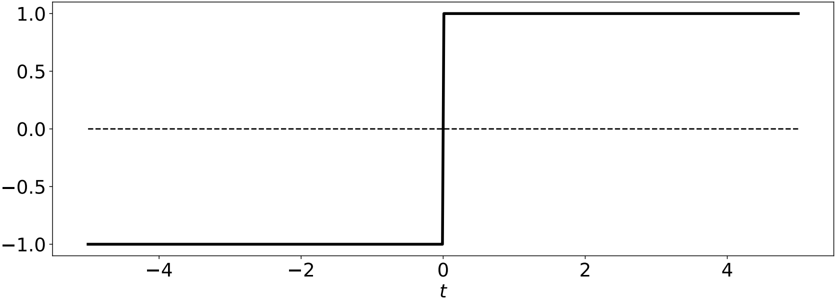


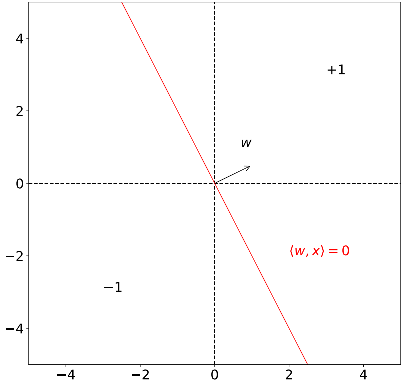
Figure: the sign function

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 $d+1$.

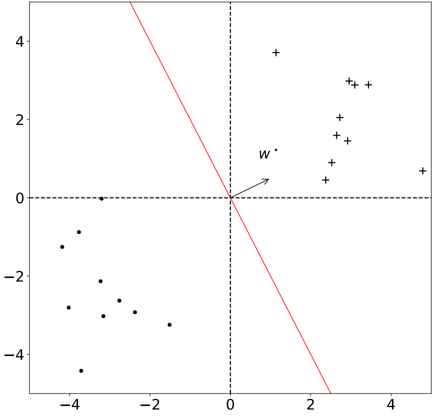
- **Consequence:** $\mathcal{H}$ is PAC learnable with sample complexity

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

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 $\times$ 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 $w^{(1)}, w^{(2)}, \dots, w^{(T)}$
- 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 $\Rightarrow$ 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 $\rightarrow$ assumes that the data is not centered (our maths are not totally accurate)
- `normalize` is False $\rightarrow$ 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. $\square$

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 $\rightarrow$ does not consider centered data (so our analysis is not entirely accurate)
- `normalize` is False $\rightarrow$ we decide whether we normalize our data
- `solver` is auto $\rightarrow$ `sklearn` will decide how to solve the minimization problem depending on the size of the data: the solution could be not exact!
- `tol` $= 0.001 \rightarrow$ tolerance threshold on the residuals

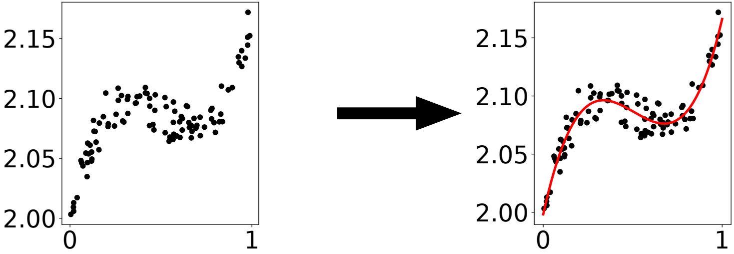
1.4. Polynomial regression

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` $\to$ very handy when we want to fit univariate data

Polynomial regression, ctd.

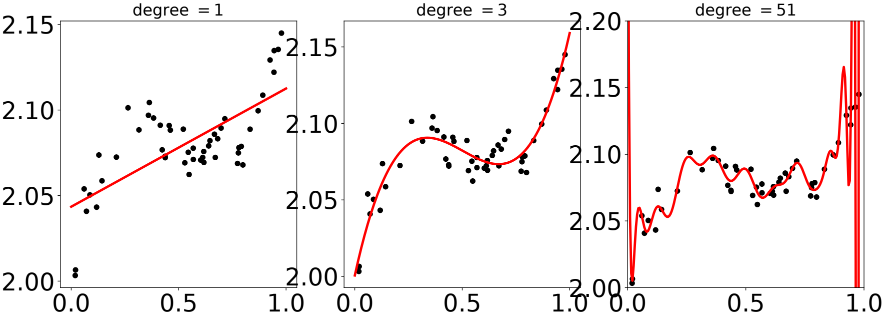
- **Example:** data = degree three polynomial + Gaussian noise with small variance
- fit a degree 3 polynomial:



- **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

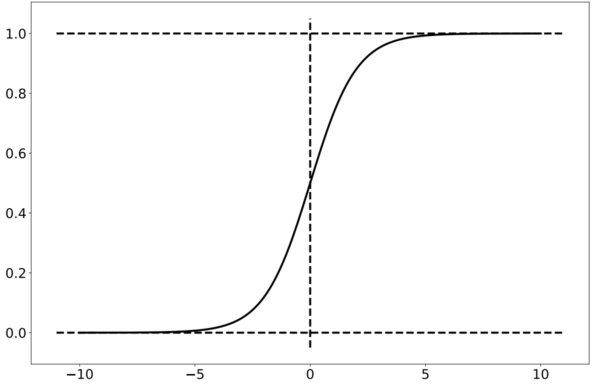
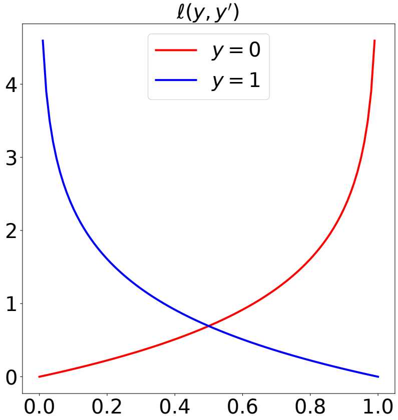


Figure: the logistic function $\phi : t \mapsto 1/(1 + \mathrm{e}^{-t})$.

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

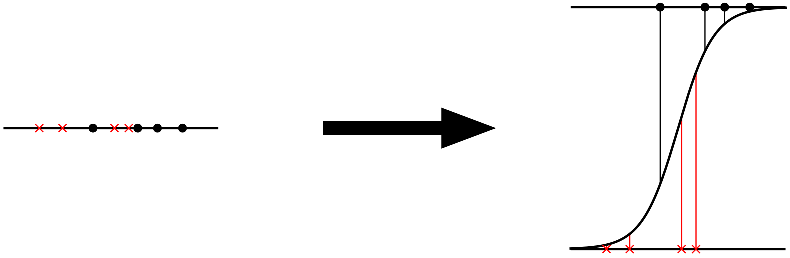
- finally, logistic regression $=$ empirical risk minimization with the logistic loss
- that is, minimize for $w \in \mathbb{R}^d$

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

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

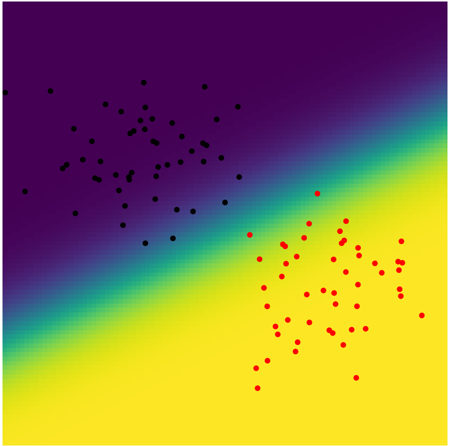
Logistic regression in dimension 1

- **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

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

   where $\tilde{y}_i = \operatorname{sign}(y_i - 0.5)$. 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. $\square$

Recap

- **What happens when we call** `sklearn.linear_model.LogisticRegression`?
- `penalty` is ℓ_2 $\rightarrow$ there is regularization by default! (not much though, $C = 1$)
- `fit_intercept` is True $\rightarrow$ again, our maths are not entirely accurate
- `solver` is liblinear $\rightarrow$ 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

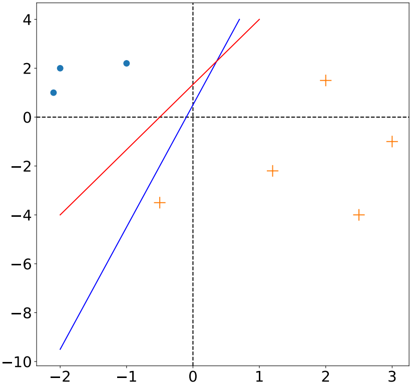
- classification with $x_1, \dots, x_n \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$
- **Recall:** linearly separable means that there exist (w, b) such that

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

- **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

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

- **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

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

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:

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

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

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