Optimization Techniques for Big Data Analysis

Chapter 2. Machine Learning Contextualization

Master of Science in Signal Theory and Communications
Dpto. de Señales, Sistemas y Radiocomunicaciones
E.T.S. Ingenieros de Telecomunicación
Universidad Politécnica de Madrid

2023

1. Basic problems in supervised learning
2. Linear Regression
3. Classification
4. Basic introduction to Neural Networks

Basic problems in supervised learning

We will be focused on two basic problems:

- Linear regression
- Linear classification

Considering a sample $\mathbf{x}_i = [x_{i1}, x_{i2}, \cdots, x_{id}]$, the model will be a form of: $f(\mathbf{x}_i) = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id} + w_0 = \mathbf{w}^T \mathbf{x}_i + w_0$.

Basic problems in supervised learning

We will be focused on two basic problems:

- Linear regression
- Linear classification

Considering a sample $\mathbf{x}_i = [x_{i1}, x_{i2}, \cdots, x_{id}]$, the model will be a form of: $f(\mathbf{x}_i) = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id} + w_0 = \mathbf{w}^T \mathbf{x}_i + w_0$.

For compactness, we can define an extended vector $\bar{\mathbf{x}}_i = [\mathbf{x}_i, 1]^T$ and a parameter vector $\bar{\mathbf{w}} = [\mathbf{w}, w_0]^T$.

Basic problems in supervised learning

We will be focused on two basic problems:

- Linear regression
- Linear classification

Considering a sample $\mathbf{x}_i = [x_{i1}, x_{i2}, \cdots, x_{id}]$, the model will be a form of: $f(\mathbf{x}_i) = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id} + w_0 = \mathbf{w}^T \mathbf{x}_i + w_0$.

For compactness, we can define an extended vector $\bar{\mathbf{x}}_i = [\mathbf{x}_i, 1]^T$ and a parameter vector $\bar{\mathbf{w}} = [\mathbf{w}, w_0]^T$.

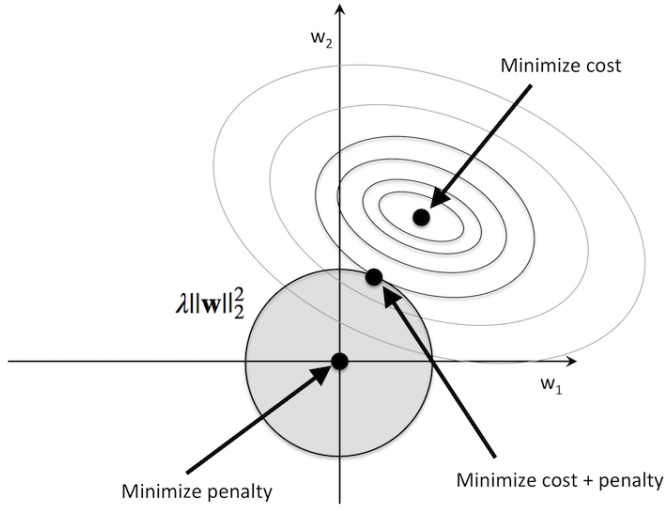
The optimization problem in those cases takes the form:

$$\arg\min_{\bar{\mathbf{w}}} \frac{1}{n} \sum_{i=1}^{n} \ell(\bar{\mathbf{w}}^T \bar{\mathbf{x}}_i, y_i) + r(\bar{\mathbf{w}}) \tag{1}$$

Regularizer

The regularizer adds an extra term (usually a norm) that penalizes/enforces certain characteristics of $\mathbf{w}$.

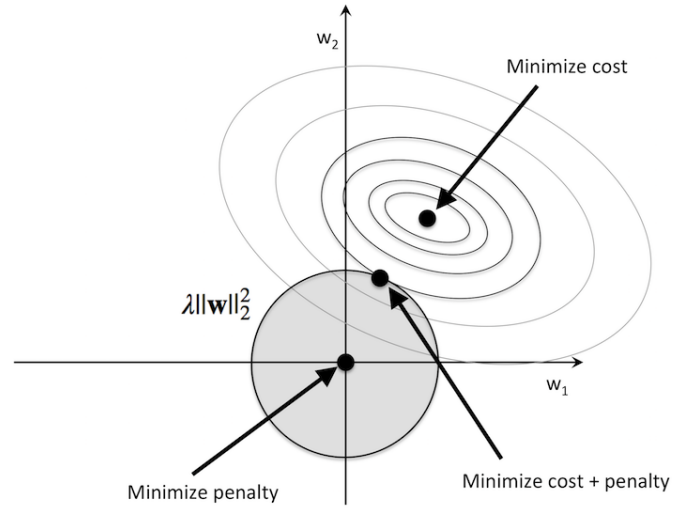
$$r(\mathbf{w}) = \frac{\lambda}{2} \|\mathbf{w}\|_2^2 = \frac{\lambda}{2} \sum_{j=1}^{d} w_j^2$$



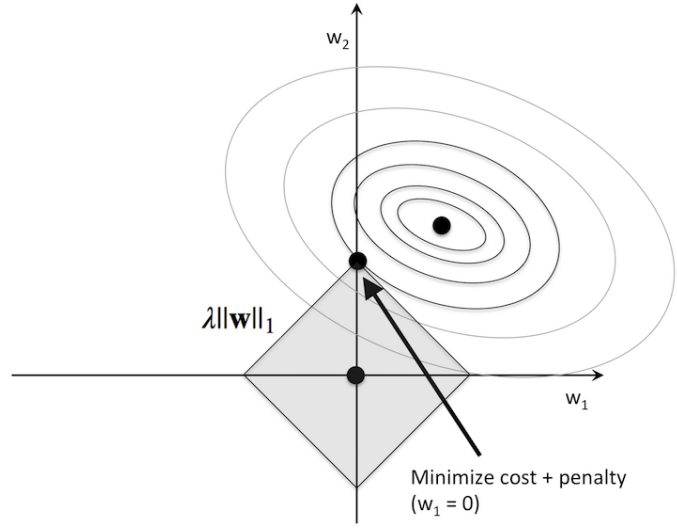
Regularizer

The regularizer adds an extra term (usually a norm) that penalizes/enforces certain characteristics of $\mathbf{w}$.

$$r(\mathbf{w}) = \frac{\lambda}{2} \|\mathbf{w}\|_2^2 = \frac{\lambda}{2} \sum_{j=1}^{d} w_j^2$$



$$r(\mathbf{w}) = \lambda \|\mathbf{w}\|_1 = \lambda \sum_{i=j}^{d} |w_j|$$



Linear regression from a Statistical Signal Processing view

In the general case of an arbitrary observable distribution $\hat{y} = g(\mathbf{x}) = \mathbb{E}(Y|X = \mathbf{x})$. If we assume that variables (Y, X) are jointly Gaussian, $p(y|\mathbf{x}) \sim \mathcal{N}(y|\mu_{y|\mathbf{x}}, \Sigma_{y|\mathbf{x}})$, where:

$$\mu_{y|\mathbf{x}} = \mu_Y + \Sigma_{X,Y}^T \Sigma_Y^{-1}(\mathbf{x} - \mu_X)$$

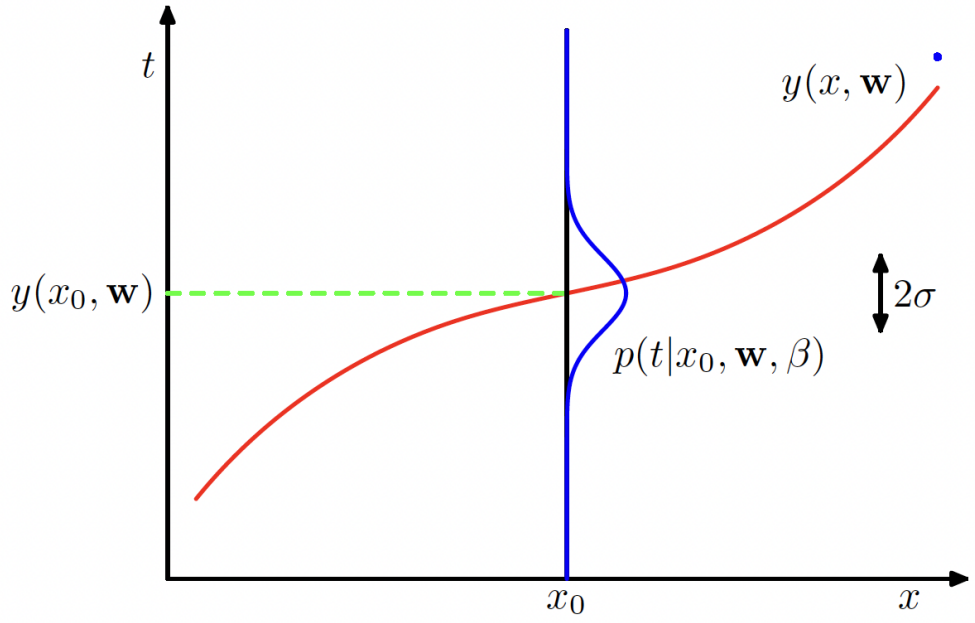
Linear regression from a Statistical Signal Processing view

In the general case of an arbitrary observable distribution $\hat{y} = g(\mathbf{x}) = \mathbb{E}(Y|X = \mathbf{x})$. If we assume that variables (Y, X) are jointly Gaussian, $p(y|\mathbf{x}) \sim \mathcal{N}(y|\mu_{y|\mathbf{x}}, \Sigma_{y|\mathbf{x}})$, where:

$$\mu_{y|\mathbf{x}} = \mu_Y + \Sigma_{X,Y}^T \Sigma_Y^{-1}(\mathbf{x} - \mu_X)$$

Equivalently:

$$\begin{aligned} p(y|\mathbf{x}, \mathbf{w}, \beta) &= \mathcal{N}\left(f(\mathbf{x}, \mathbf{w}), \beta^{-1}\right) \\ &= \mathcal{N}\left(\mathbf{w}^T\mathbf{x}, \beta^{-1}\right) \end{aligned}$$



Least Square Error

Given a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1}^n\}$, of $i.i.d$ samples, the **likelihood** function is [1]:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{i=1}^{n} \mathcal{N}\left(y_i|\mathbf{w}^T\mathbf{x}_i, \beta^{-1}\right)$$

Least Square Error

Given a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1}^n\}$, of $i.i.d$ samples, the **likelihood** function is [1]:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{i=1}^{n} \mathcal{N}\left(y_i|\mathbf{w}^T\mathbf{x}_i, \beta^{-1}\right)$$

For numerical stability, it is convenient to maximize the logarithm of the likelihood function:

$$\ln p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta) = -\frac{\beta}{2}\sum_{i=1}^{n}\left(\mathbf{w}^T\mathbf{x}_i - y_i\right)^2 + \frac{n}{2}\ln\beta - \frac{n}{2}\ln(2\pi)$$

Least Square Error

Given a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1}^n\}$, of $i.i.d$ samples, the **likelihood** function is [1]:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{i=1}^{n} \mathcal{N}\left(y_i|\mathbf{w}^T\mathbf{x}_i, \beta^{-1}\right)$$

For numerical stability, it is convenient to maximize the logarithm of the likelihood function:

$$\ln p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta) = -\frac{\beta}{2}\sum_{i=1}^{n}\left(\mathbf{w}^T\mathbf{x}_i - y_i\right)^2 + \frac{n}{2}\ln\beta - \frac{n}{2}\ln(2\pi)$$

This is equivalent to minimizing:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}\left(\mathbf{w}^T\mathbf{x}_i - y_i\right)^2$$

Linear Regression as a Least Square problem

$$\arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}^T\mathbf{x}_i, y_i) = \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{n}\sum_{i=1}^{n} (\mathbf{w}^T\mathbf{x}_i - y_i)^2$$

Linear Regression as a Least Square problem

$$\arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}^T\mathbf{x}_i, y_i) = \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{n}\sum_{i=1}^{n} (\mathbf{w}^T\mathbf{x}_i - y_i)^2$$

In matrix terms, by defining the matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]^T$ and the vector $\mathbf{y} = [y_1, y_2, \cdots, y_n]^T$,

$$\arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{n}\sum_{i=1}^{n} (\mathbf{w}^T\mathbf{x}_i - y_i)^2 = \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{n}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$$

Linear Regression as a Least Square problem

$$\arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}^T\mathbf{x}_i, y_i) = \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{n}\sum_{i=1}^{n} (\mathbf{w}^T\mathbf{x}_i - y_i)^2$$

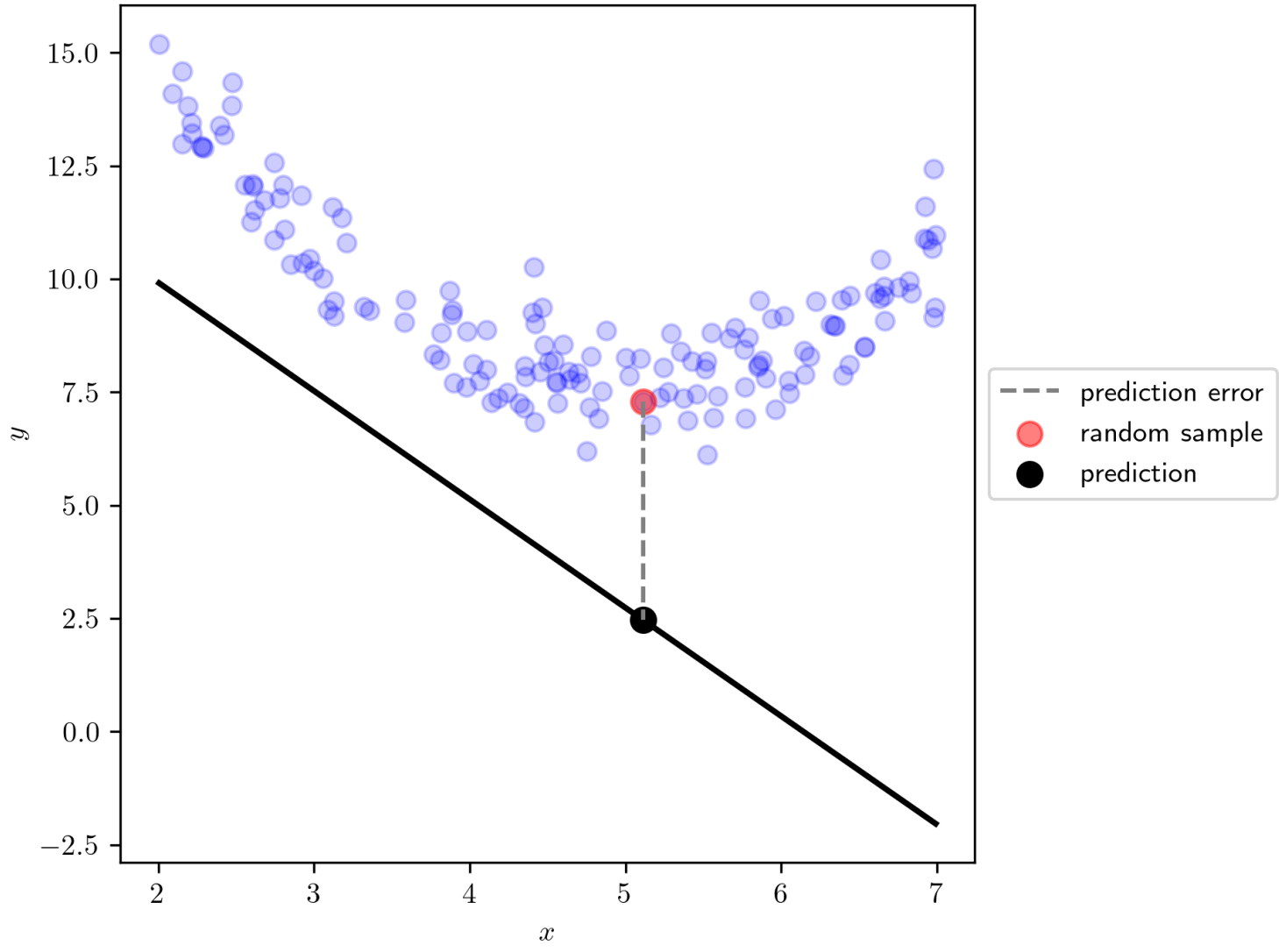
In matrix terms, by defining the matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]^T$ and the vector $\mathbf{y} = [y_1, y_2, \cdots, y_n]^T$,

$$\arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{n}\sum_{i=1}^{n} (\mathbf{w}^T\mathbf{x}_i - y_i)^2 = \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{n}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$$

Whose analytical solution is:

$$\frac{1}{n}\mathbf{X}^T(\mathbf{X}\mathbf{w} - \mathbf{y}) = 0;\ \mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

Error in Least Square



Ridge regression

$$\mathcal{L}(\mathbf{w}) = \underset{\mathbf{w}\in\mathbb{R}^{d+1}}{\arg\min} \frac{1}{n}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

Ridge regression

$$\mathcal{L}(\mathbf{w}) = \underset{\mathbf{w}\in\mathbb{R}^{d+1}}{\arg\min} \frac{1}{n}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

Whose analytical solution turns into:

$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X} + \frac{\lambda n}{2}\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

Ridge regression

$$\mathcal{L}(\mathbf{w}) = \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{n}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

Whose analytical solution turns into:

$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X} + \frac{\lambda n}{2}\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

Multiple interpretations:

- Reduce overfitting (promotes smoothness)
- Reduce the variance of the estimator
- Makes the matrix invertible

Ridge regression

$$\mathcal{L}(\mathbf{w}) = \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{n}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

Whose analytical solution turns into:

$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X} + \frac{\lambda n}{2}\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

Multiple interpretations:

- Reduce overfitting (promotes smoothness)
- Reduce the variance of the estimator
- Makes the matrix invertible

Probabilistic derivation:

$p(\mathbf{w}|\lambda) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda^{-1}\mathbf{I})$

Ridge regression

$$\mathcal{L}(\mathbf{w}) = \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{n}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

Whose analytical solution turns into:

$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X} + \frac{\lambda n}{2}\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

Multiple interpretations:

- Reduce overfitting (promotes smoothness)
- Reduce the variance of the estimator
- Makes the matrix invertible

Probabilistic derivation:

$p(\mathbf{w}|\lambda) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda^{-1}\mathbf{I}) \rightarrow$

Ridge regression

$$\mathcal{L}(\mathbf{w}) = \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{n}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

Whose analytical solution turns into:

$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X} + \frac{\lambda n}{2}\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

Multiple interpretations:

- Reduce overfitting (promotes smoothness)
- Reduce the variance of the estimator
- Makes the matrix invertible

Probabilistic derivation:

$$p(\mathbf{w}|\lambda) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda^{-1}\mathbf{I}) \rightarrow \mathcal{L}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}\left(\mathbf{w}^T\mathbf{x}_i - y_i\right)^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

LASSO regression

LASSO (Least Absolute Shrinkage and Selection Operator) looks similar but uses a L_1 regularizer instead:

$$\mathcal{L}(\mathbf{w}) = \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{n}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_1$$

LASSO regression

LASSO (Least Absolute Shrinkage and Selection Operator) looks similar but uses a L_1 regularizer instead:

$$\mathcal{L}(\mathbf{w}) = \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{n}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_1$$

Considerations:

- Promotes sparseness
- The gradient is not a smooth function (optimizing it requires subgradient or proximal methods)
- Corresponds to imposing a zero-mean Laplacean prior over $\mathbf{w}$.

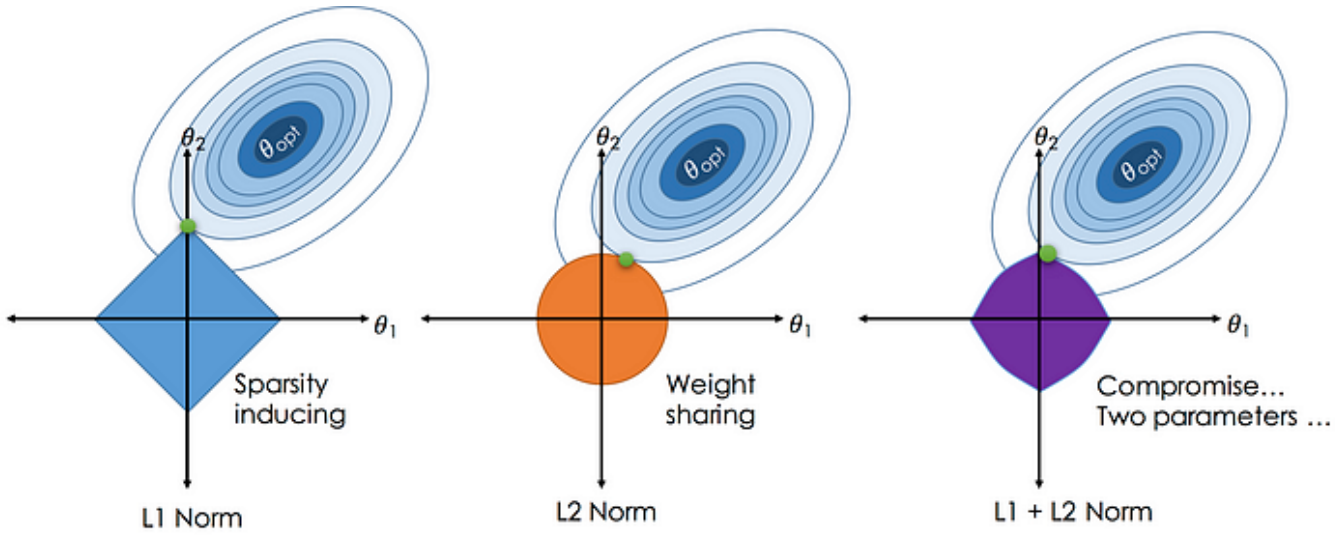
LASSO regression

LASSO (Least Absolute Shrinkage and Selection Operator) looks similar but uses a L_1 regularizer instead:

$$\mathcal{L}(\mathbf{w}) = \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{n}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_1$$

Considerations:

- Promotes sparseness
- The gradient is not a smooth function (optimizing it requires subgradient or proximal methods)
- Corresponds to imposing a zero-mean Laplacean prior over $\mathbf{w}$.



Basis pursuit

1. Basis pursuit is a similar problem to linear regression but with a different goal: the idea now is to find a good fit for the given data as a linear combination of a small number of the basis functions.
2. In this context, the basis family use to be referred to as a dictionary.
3. The goal now is that we seek a function ϕ that fits the data well:

$$\Phi\left(\mathbf{x}_i\right) \approx y_i \qquad \forall i$$

such that this function can be expressed as a linear combination of a particular basis:

$$\Phi\left(\mathbf{x}\right) = \sum_{j=0}^{d} \mathbf{w}_j \phi_j\left(\mathbf{x}\right)$$

Basis Pursuit

The formulation is well known to us (typically an L1-norm is added):

$$\arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \left(\frac{1}{n}\sum_{i=1}^{n} \left(\underbrace{\sum_{j=0}^{d} w_j \phi_j\left(\mathbf{x}_i\right)}_{\Phi(\mathbf{x}_i)} - y_i \right)^2 + \lambda \left\|\mathbf{w}\right\|_1 \right)$$

Basis Pursuit

The formulation is well known to us (typically an L1-norm is added):

$$\underset{\mathbf{w}\in\mathbb{R}^{d+1}}{\arg\min}\left(\frac{1}{n}\sum_{i=1}^{n}\left(\underbrace{\sum_{j=0}^{d}w_j\phi_j\left(\mathbf{x}_i\right)}_{\Phi(\mathbf{x}_i)}-y_i\right)^2+\lambda\left\|\mathbf{w}\right\|_1\right)$$

In matrix form, we define:

$$\mathbf{X}=\begin{bmatrix}\phi_0\left(\mathbf{x}_1\right) & \cdots & \phi_d\left(\mathbf{x}_1\right)\\ \vdots & \ddots & \vdots\\ \phi_0\left(\mathbf{x}_n\right) & & \phi_d\left(\mathbf{x}_n\right)\end{bmatrix}$$

and we reach the standard LASSO-like expression:

$$\mathcal{L}(\mathbf{w})=\underset{\mathbf{w}\in\mathbb{R}^{d+1}}{\arg\min}\frac{1}{n}\|\mathbf{X}\mathbf{w}-\mathbf{y}\|_2^2+\lambda\|\mathbf{w}\|_1$$

Basis Pursuit. Example 2_1

Let us suppose that our observable has the following structure

$$y = w_1\phi_1(x) + w_2\phi_2(x) + \varepsilon$$

where $x \in (0, 10)$ and the two arbitrary basis are

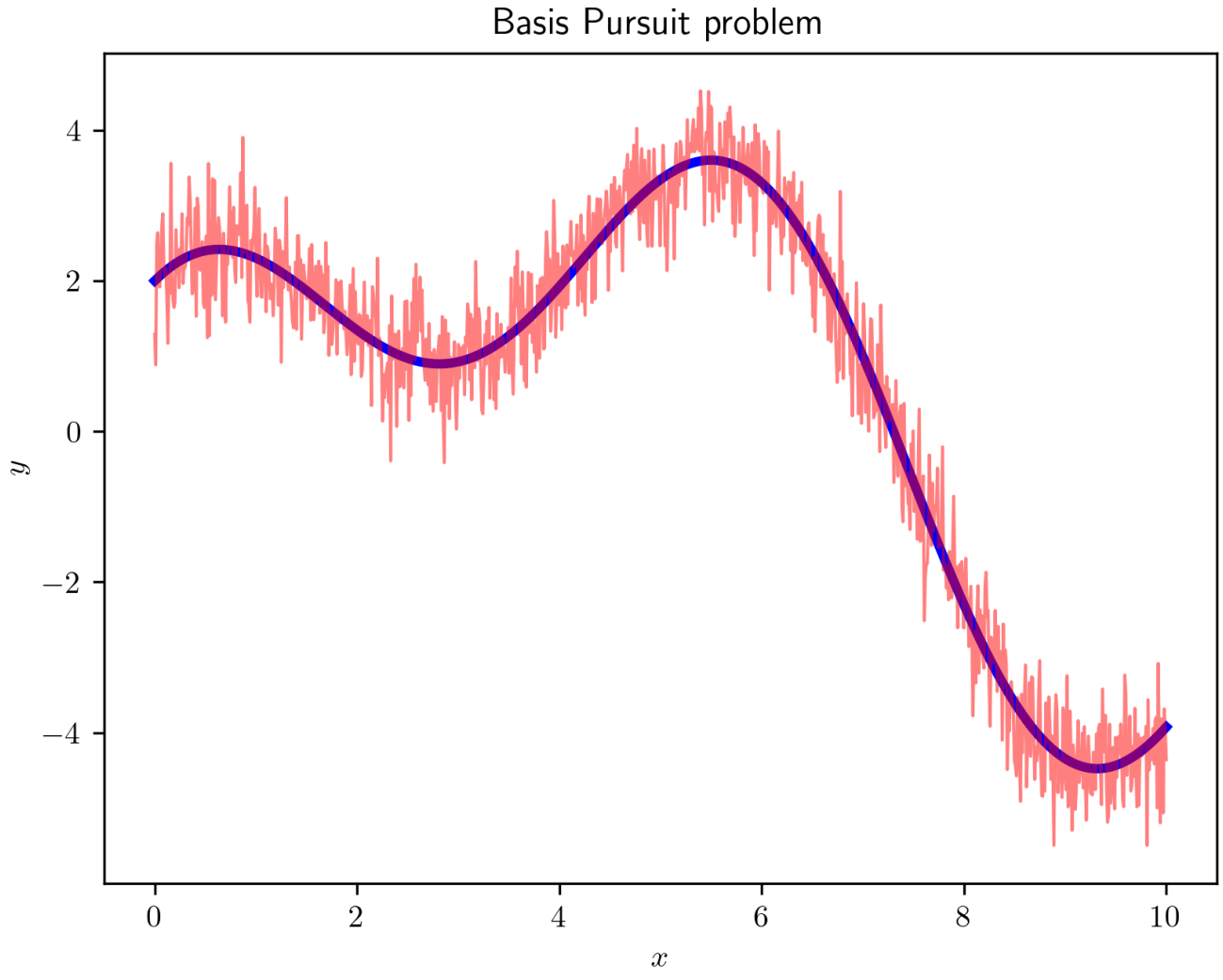
$$\phi_1(x) = \cos\frac{\pi}{3}x, \phi_2(x) = \sin\frac{\pi}{7}x$$

and ε a white Gaussian noise with power σ_n^2. The objective is to write a Python code to calculate coefficients w_1, w_2 from y according to

$$\min_{\mathbf{x}\in\mathbb{R}^d}\left(\frac{1}{N}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2\right)$$

Next figure shows the case where $w_1 = 2$, $w_2 = 3$ and σ_n^2=0.25.

Example 2_1 (Denoising)



Classification. Basic ideas

The classification problem is just like the regression problem, except that the values y_i that we want to predict take on only a small number of discrete values.

We will show two very popular approaches:

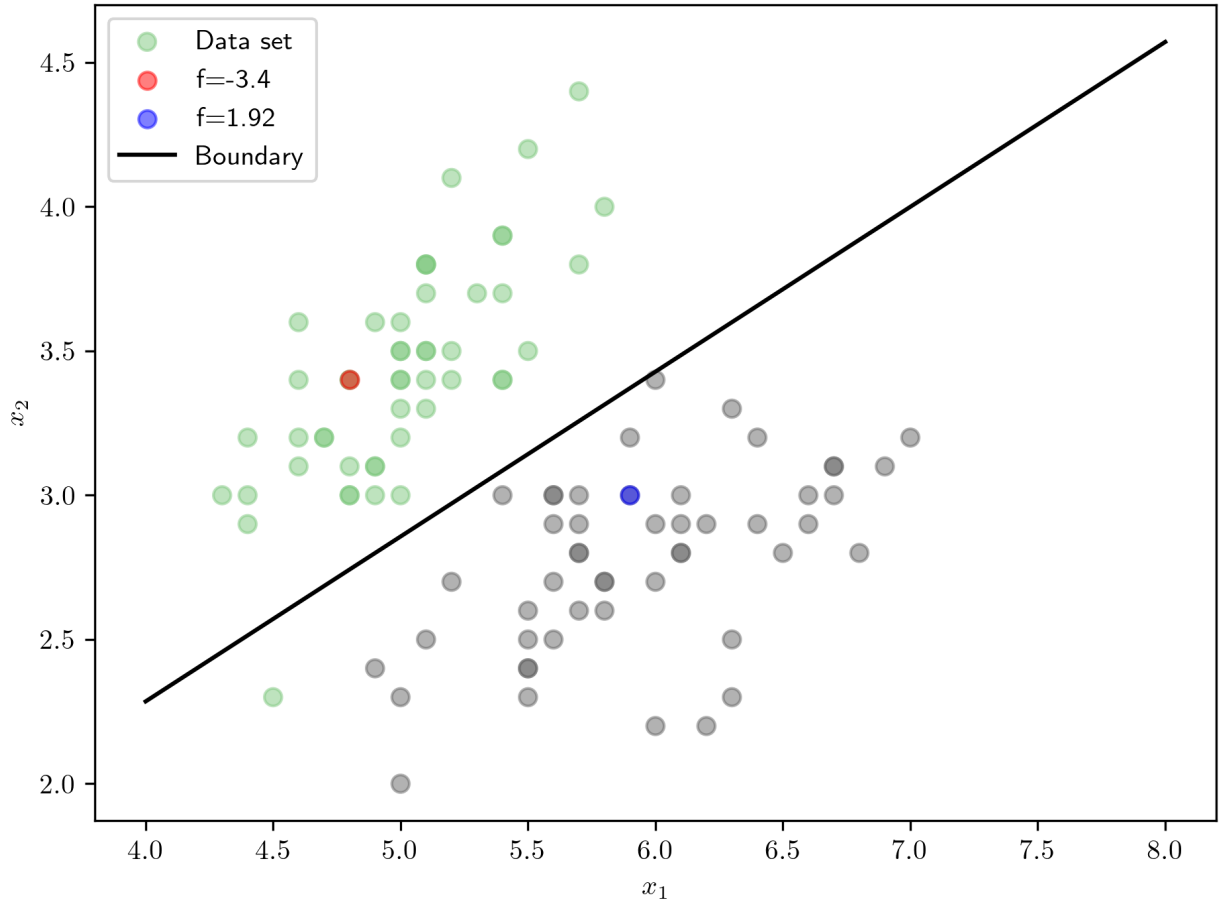
- Logistic Regression
- Support Vector Machines (SVM)

We will be just focused on the binary case, $y_i \in \{+1, -1\}$, in order to simplify the interpretations.

Extensions to more classes are straightforward.

Classification. Basic ideas

In this case, we talk about *discriminative functions* as those that represent the borders of decision regions.



Optimum Bayesian boundary

Defining the probability of a certain hypothesis conditioned to a certain observable

$$P(H_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid H_i)P(H_i)}{p(\mathbf{x})}$$

after the Bayes rule and, since $p(\mathbf{x}) \geq 0$ and it does not depend on i, to maximize the likelihood *a posteriori* is equivalent to maximize the numerator resulting in the rule based on the likelihood functions

$$\text{Accept } H_i \text{ iff } p(X \mid H_i)P(H_i) > p(X \mid H_j)P(H_j),\ \forall j \neq i$$

or, taking logarithms

$$\text{Accept } H_i \text{ iff } \ln p(X \mid H_i) + \ln P(H_i) > \ln p(X \mid H_j) + \ln P(H_j), \forall j \neq i$$

Optimum Bayesian classification in the Gaussian case

Therefore, in general, a Bayesian classifier will use a decision rule type

$$\text{Accept } H_i \text{ iff } g_i(X) > g_j(X),\ \forall j \neq i$$

where $g_i(\mathbf{x})$, $i = 0, 1, \ldots, M-1$ ($M = 2$ for the binary case) are called *discriminant functions*. For a two-class, we can define a single discriminant function

$$g(\mathbf{x}) \equiv g_1(\mathbf{x}) - g_2(\mathbf{x})$$

which decides H_1 if $g(\mathbf{x}) > 0$; otherwise decide H_2. The *borders* between the decision regions of the hypotheses is the set of points $\mathbf{x} \in \mathbb{R}^d$ where $g(\mathbf{x}) = 0$.

Linear classifier

Suppose that the observation vector follows a multivariate Gaussian distribution: $X \sim \mathcal{N}(\mu, \Sigma)$.

The discriminative function for the i-th class will be [2]:

$$g_i(\mathbf{x}) = -\frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| - \frac{1}{2}\mathbf{x}^T\Sigma_i^{-1}\mathbf{x} + \mu_i^T\Sigma_i^{-1}\mathbf{x} - \frac{1}{2}\mu_i^T\Sigma_i^{-1}\mu_i + \ln P(H_i)$$

Linear classifier

Suppose that the observation vector follows a multivariate Gaussian distribution: $X \sim \mathcal{N}(\mu, \Sigma)$.

The discriminative function for the i-th class will be [2]:

$$g_i(\mathbf{x}) = -\frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| - \frac{1}{2}\mathbf{x}^T\Sigma_i^{-1}\mathbf{x} + \mu_i^T\Sigma_i^{-1}\mathbf{x} - \frac{1}{2}\mu_i^T\Sigma_i^{-1}\mu_i + \ln P(H_i)$$

- Case 1: $\Sigma_i = \sigma^2\mathbf{I}$

$$g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + w_{i0} \rightarrow \mathbf{w}_i = \frac{1}{\sigma^2}\mu_i; \quad w_{i0} = \frac{-1}{2\sigma^2}\mu_i^T\mu_i + \ln P(H_i)$$

Linear classifier

Suppose that the observation vector follows a multivariate Gaussian distribution: $X \sim \mathcal{N}(\mu, \Sigma)$.

The discriminative function for the i-th class will be [2]:

$$g_i(\mathbf{x}) = -\frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| - \frac{1}{2}\mathbf{x}^T\Sigma_i^{-1}\mathbf{x} + \mu_i^T\Sigma_i^{-1}\mathbf{x} - \frac{1}{2}\mu_i^T\Sigma_i^{-1}\mu_i + \ln P(H_i)$$

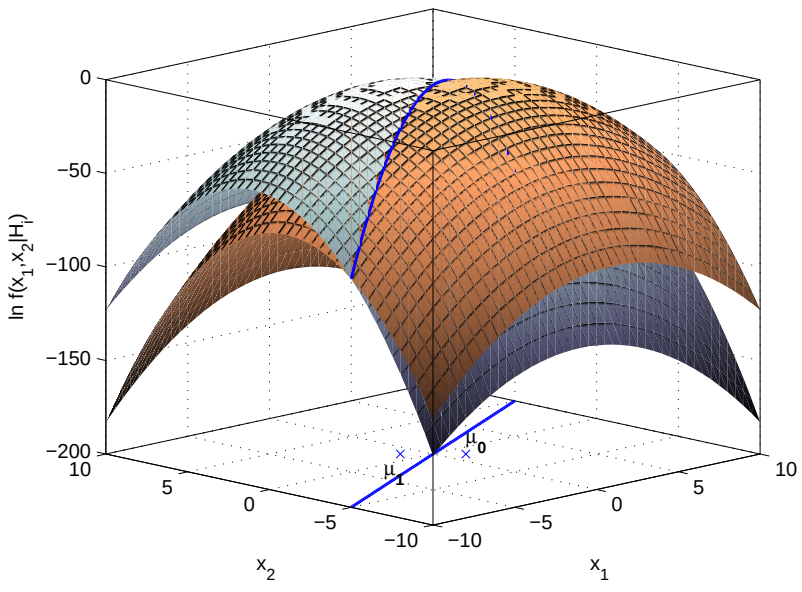
- Case 1: $\Sigma_i = \sigma^2\mathbf{I}$

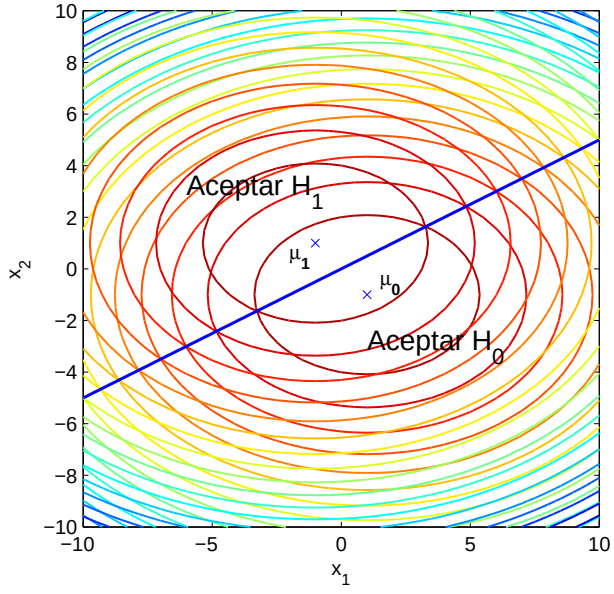
$$g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + w_{i0} \rightarrow \mathbf{w}_i = \frac{1}{\sigma^2}\mu_i; \quad w_{i0} = \frac{-1}{2\sigma^2}\mu_i^T\mu_i + \ln P(H_i)$$

- Case 2: $\Sigma_i = \Sigma$

$$g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + w_{i0} \rightarrow \mathbf{w}_i = \sigma^{-1}\mu_i; \quad w_{i0} = \frac{-1}{2}\mu_i^T\Sigma^{-1}\mu_i + \ln P(H_i)$$

Linear classifier





$P(H_0) = P(H_1) = 1/2$. Border regions decision

Quadratic Classifier

- Case 3: $\Sigma_i = \text{arbitrary}$

$$g_i(\mathbf{x}) = \mathbf{x}^T \mathbf{W}_i \mathbf{x} + \mathbf{w}_i^T \mathbf{x} + w_{i0}$$

$$\mathbf{W}_i = -\frac{1}{2}\Sigma_i^{-1}$$

$$\mathbf{w}_i = \Sigma_i^{-1}\mu_i$$

$$w_{i0} = \frac{-1}{2}\mu_i^T \Sigma_i^{-1}\mu_i - \frac{1}{2}\ln|\Sigma_i| + \ln P(H_i)$$

Quadratic Classifier

- Case 3: $\Sigma_i = \text{arbitrary}$

$$g_i(\mathbf{x}) = \mathbf{x}^T \mathbf{W}_i \mathbf{x} + \mathbf{w}_i^T \mathbf{x} + w_{i0}$$

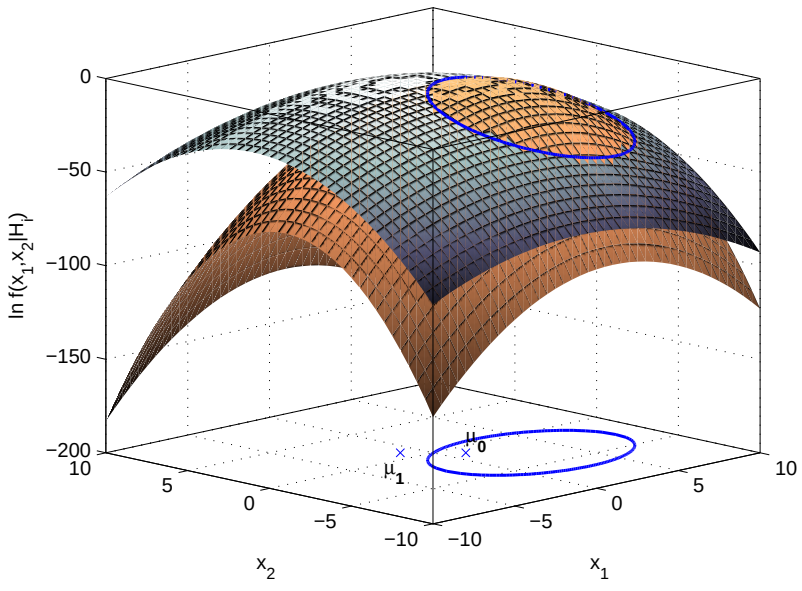
$$\mathbf{W}_i = -\frac{1}{2}\Sigma_i^{-1}$$

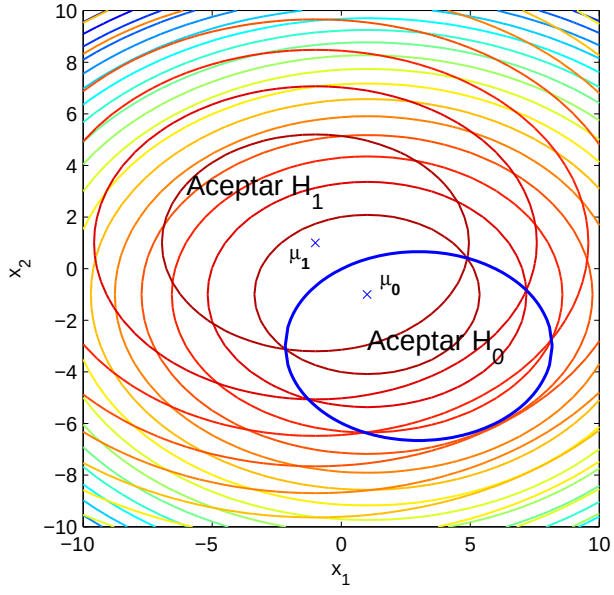
$$\mathbf{w}_i = \Sigma_i^{-1}\mu_i$$

$$w_{i0} = \frac{-1}{2}\mu_i^T \Sigma_i^{-1}\mu_i - \frac{1}{2}\ln|\Sigma_i| + \ln P(H_i)$$

There is an additional case where you assume that Σ_i is arbitrary but diagonal, which is call Naïve Bayes Classifier.

Quadratic classifier

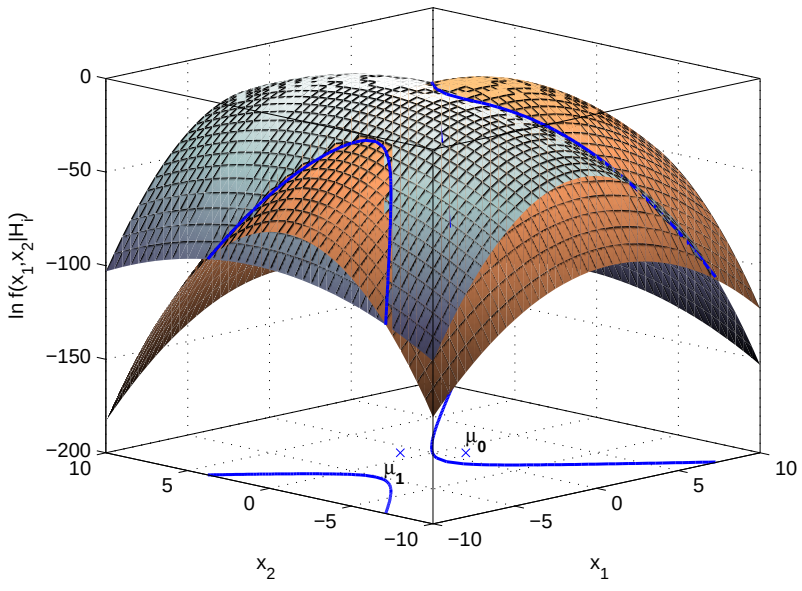


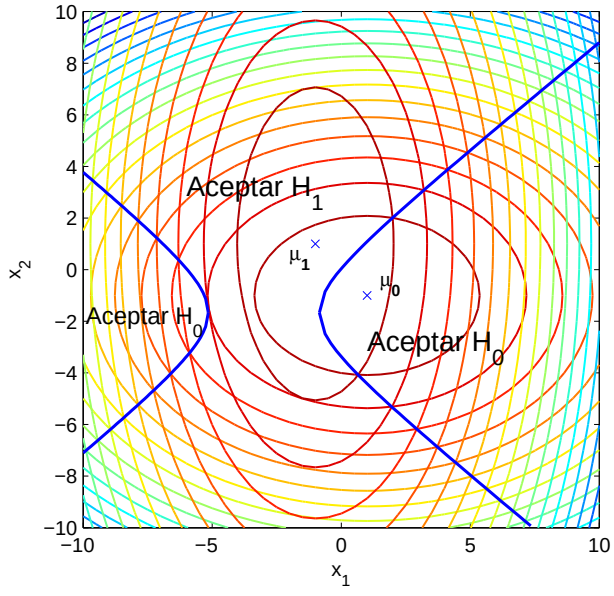


$$P(H_0) = P(H_1) = 1/2,\ \mu_0 = [1,-1]^T,\ \mu_1 = [-1,1]^T,$$
$$\Sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{bmatrix},\ \Sigma_1 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

The border is an ellipse

Quadratic classifier

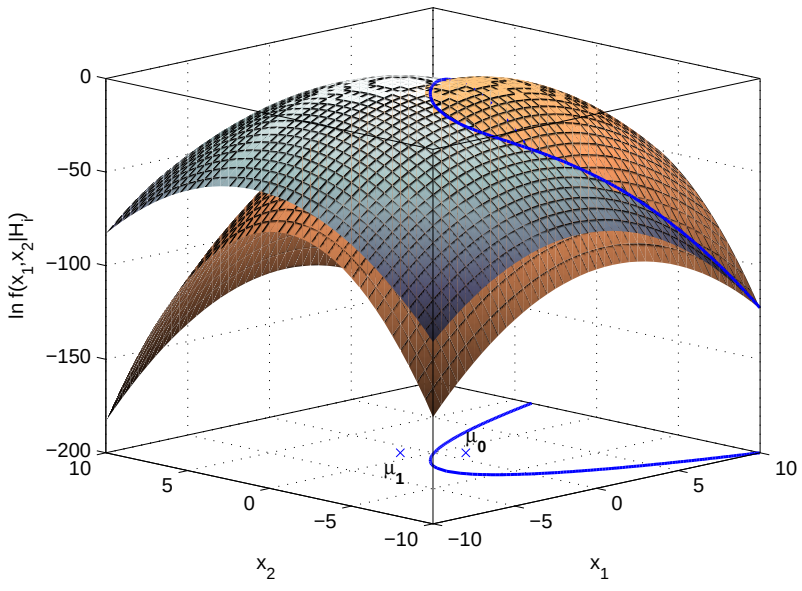


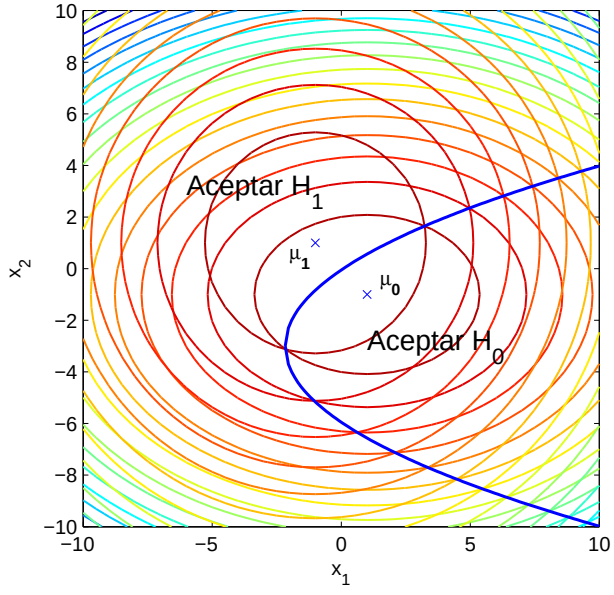


$$P(H_0) = P(H_1) = 1/2,\ \mu_0 = [1, -1]^T,\ \mu_1 = [-1, 1]^T,$$
$$\Sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{bmatrix},\ \Sigma_1 = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & 2 \end{bmatrix}$$

The border is a hyperbola

Quadratic classifier





$$P(H_0) = P(H_1) = 1/2,\ \mu_0 = [1,-1]^T,\ \mu_1 = [-1,1]^T,$$
$$\Sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{bmatrix},\ \Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The border is a parabola

Logistic Regression

Like Linear regression, we can apply the maximum likelihood criterion to a classification problem. Assuming a Bernoulli distribution (2-class problem):

$$\max \mathcal{L} = \log \left(\prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{(1-y_i)} \right) ; \quad y_i \in \{0, 1\}$$

Logistic Regression

Like Linear regression, we can apply the maximum likelihood criterion to a classification problem. Assuming a Bernoulli distribution (2-class problem):

$$\max \mathcal{L} = \log \left(\prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{(1-y_i)} \right); \quad y_i \in \{0, 1\}$$

To model $p_i = p(\mathbf{x}_i; \mathbf{w}, w_0) = P(Y = 1 | X = \mathbf{x}_i; \mathbf{w}, w_0)$, logistic regression uses the inverse of a *logit* function to map the output of a linear function to the interval $(0, 1)$.

$$p(\mathbf{x}_i; \mathbf{w}, w_0) = \frac{1}{1 + \exp\left(-(\mathbf{w}^T \mathbf{x}_i + w_0)\right)}$$

Logistic Regression

Like Linear regression, we can apply the maximum likelihood criterion to a classification problem. Assuming a Bernoulli distribution (2-class problem):

$$\max \mathcal{L} = \log \left(\prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{(1-y_i)} \right); \quad y_i \in \{0, 1\}$$

To model $p_i = p(\mathbf{x}_i; \mathbf{w}, w_0) = P(Y = 1 | X = \mathbf{x}_i; \mathbf{w}, w_0)$, logistic regression uses the inverse of a *logit* function to map the output of a linear function to the interval $(0, 1)$.

$$p(\mathbf{x}_i; \mathbf{w}, w_0) = \frac{1}{1 + \exp\left(-(\mathbf{w}^T \mathbf{x}_i + w_0)\right)} = g(\bar{\mathbf{w}}^T \bar{\mathbf{x}}_i)$$

Logistic loss

$$\begin{aligned}
\arg\max_{\mathbf{w}} \mathcal{L} &= \sum_{i=1}^{n} \log\left((g(\mathbf{w}^T\mathbf{x}_i))^{y_i}\right) + \log\left((1-g(\mathbf{w}^T\mathbf{x}_i))^{(1-y_i)}\right) \\
&= \sum_{i=1}^{n} y_i \log\left(g(\mathbf{w}^T\mathbf{x}_i)\right) + (1-y_i)\log\left(1-g(\mathbf{w}^T\mathbf{x}_i)\right)
\end{aligned}$$

Logistic loss

$$\begin{aligned}
\arg\max_{\mathbf{w}} \mathcal{L} &= \sum_{i=1}^{n} \log\left((g(\mathbf{w}^T\mathbf{x}_i))^{y_i}\right) + \log\left((1-g(\mathbf{w}^T\mathbf{x}_i))^{(1-y_i)}\right) \\
&= \sum_{i=1}^{n} y_i \log\left(g(\mathbf{w}^T\mathbf{x}_i)\right) + (1-y_i)\underbrace{\log\left(1-g(\mathbf{w}^T\mathbf{x}_i)\right)}_{\log(g(-\mathbf{w}^T\mathbf{x}_i))}
\end{aligned}$$

Logistic loss

$$\begin{aligned}\arg\max_{\mathbf{w}} \mathcal{L} &= \sum_{i=1}^{n} \log\left((g(\mathbf{w}^T\mathbf{x}_i))^{y_i}\right) + \log\left((1-g(\mathbf{w}^T\mathbf{x}_i))^{(1-y_i)}\right) \\ &= \sum_{i=1}^{n} y_i \log\left(g(\mathbf{w}^T\mathbf{x}_i)\right) + (1-y_i)\underbrace{\log\left(1-g(\mathbf{w}^T\mathbf{x}_i)\right)}_{\log(g(-\mathbf{w}^T\mathbf{x}_i))}\end{aligned}$$

By resetting $y_i \in \{+1, -1\}$ and taking numerical stability into consideration,

Logistic loss

$$\begin{aligned}
\arg\max_{\mathbf{w}} \mathcal{L} &= \sum_{i=1}^{n} \log\left((g(\mathbf{w}^T\mathbf{x}_i))^{y_i}\right) + \log\left((1-g(\mathbf{w}^T\mathbf{x}_i))^{(1-y_i)}\right) \\
&= \sum_{i=1}^{n} y_i \log\left(g(\mathbf{w}^T\mathbf{x}_i)\right) + (1-y_i)\underbrace{\log\left(1-g(\mathbf{w}^T\mathbf{x}_i)\right)}_{\log(g(-\mathbf{w}^T\mathbf{x}_i))}
\end{aligned}$$

By resetting $y_i \in \{+1, -1\}$ and taking numerical stability into consideration,

$$\arg\min_{\mathbf{w}} \mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n} \log\left(g(y_i\mathbf{w}^T\mathbf{x}_i)\right)$$

Logistic loss

$$\begin{aligned}\arg\max_{\mathbf{w}} \mathcal{L} &= \sum_{i=1}^{n} \log\left((g(\mathbf{w}^T\mathbf{x}_i))^{y_i}\right) + \log\left((1-g(\mathbf{w}^T\mathbf{x}_i))^{(1-y_i)}\right) \\ &= \sum_{i=1}^{n} y_i \log\left(g(\mathbf{w}^T\mathbf{x}_i)\right) + (1-y_i)\underbrace{\log\left(1-g(\mathbf{w}^T\mathbf{x}_i)\right)}_{\log(g(-\mathbf{w}^T\mathbf{x}_i))}\end{aligned}$$

By resetting $y_i \in \{+1, -1\}$ and taking numerical stability into consideration,

$$\begin{aligned}\arg\min_{\mathbf{w}} \mathcal{L} &= -\frac{1}{n}\sum_{i=1}^{n} \log\left(g(y_i\mathbf{w}^T\mathbf{x}_i)\right) \\ &= \frac{1}{n}\sum_{i=1}^{n} \log(1+\exp\left(-y_i\mathbf{w}^T\mathbf{x}_i\right))\end{aligned}$$

Logistic loss

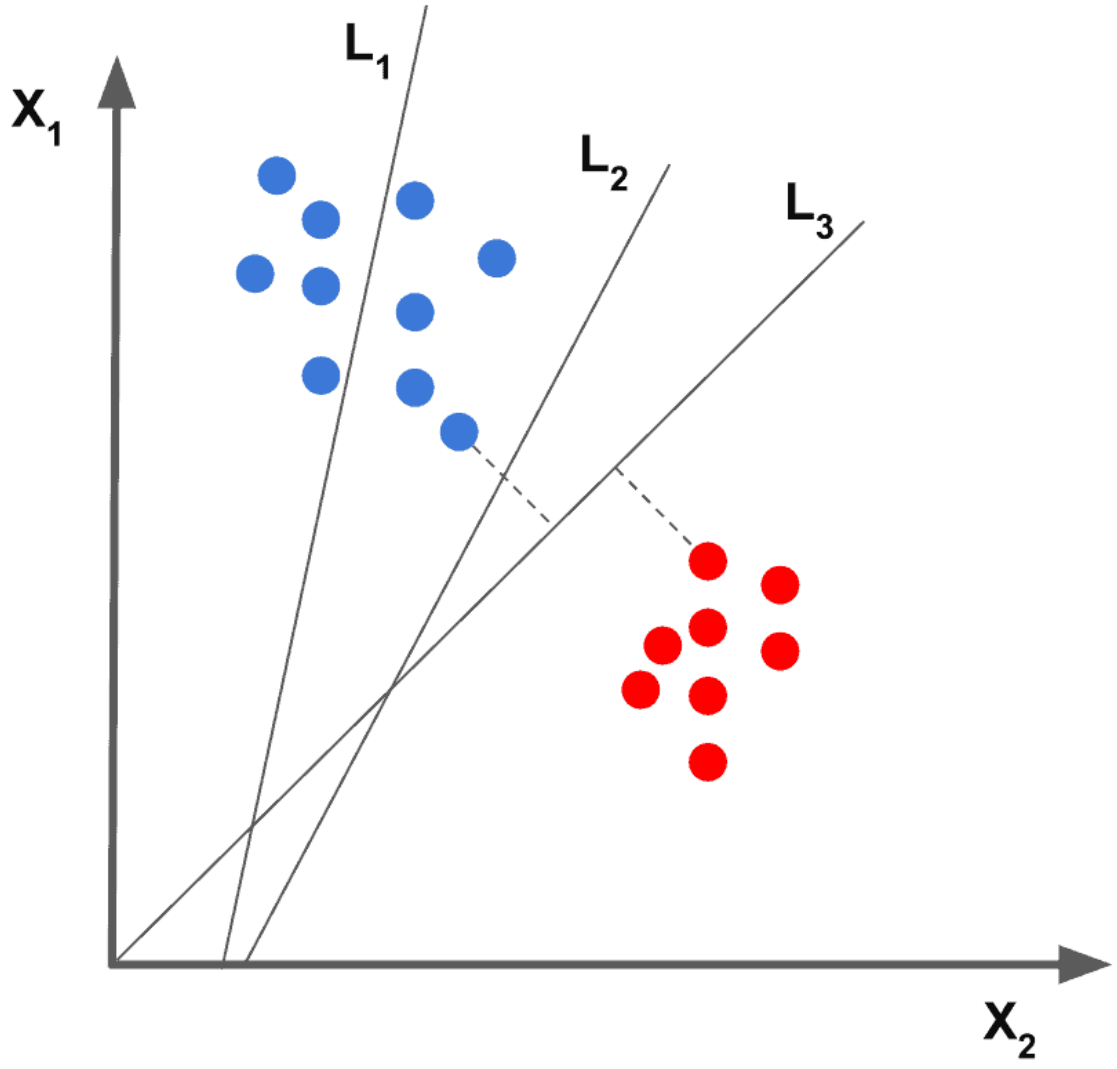
$$\begin{aligned}\arg\max_{\mathbf{w}} \mathcal{L} &= \sum_{i=1}^{n} \log\left((g(\mathbf{w}^T\mathbf{x}_i))^{y_i}\right) + \log\left((1-g(\mathbf{w}^T\mathbf{x}_i))^{(1-y_i)}\right) \\ &= \sum_{i=1}^{n} y_i \log\left(g(\mathbf{w}^T\mathbf{x}_i)\right) + (1-y_i)\underbrace{\log\left(1-g(\mathbf{w}^T\mathbf{x}_i)\right)}_{\log(g(-\mathbf{w}^T\mathbf{x}_i))}\end{aligned}$$

By resetting $y_i \in \{+1, -1\}$ and taking numerical stability into consideration,

$$\begin{aligned}\arg\min_{\mathbf{w}} \mathcal{L} &= -\frac{1}{n}\sum_{i=1}^{n} \log\left(g(y_i\mathbf{w}^T\mathbf{x}_i)\right) \\ &= \frac{1}{n}\sum_{i=1}^{n} \log(1+\exp\left(-y_i\mathbf{w}^T\mathbf{x}_i\right))\end{aligned}$$

It is straightforward to add a regularisation term $r(\mathbf{w})$.

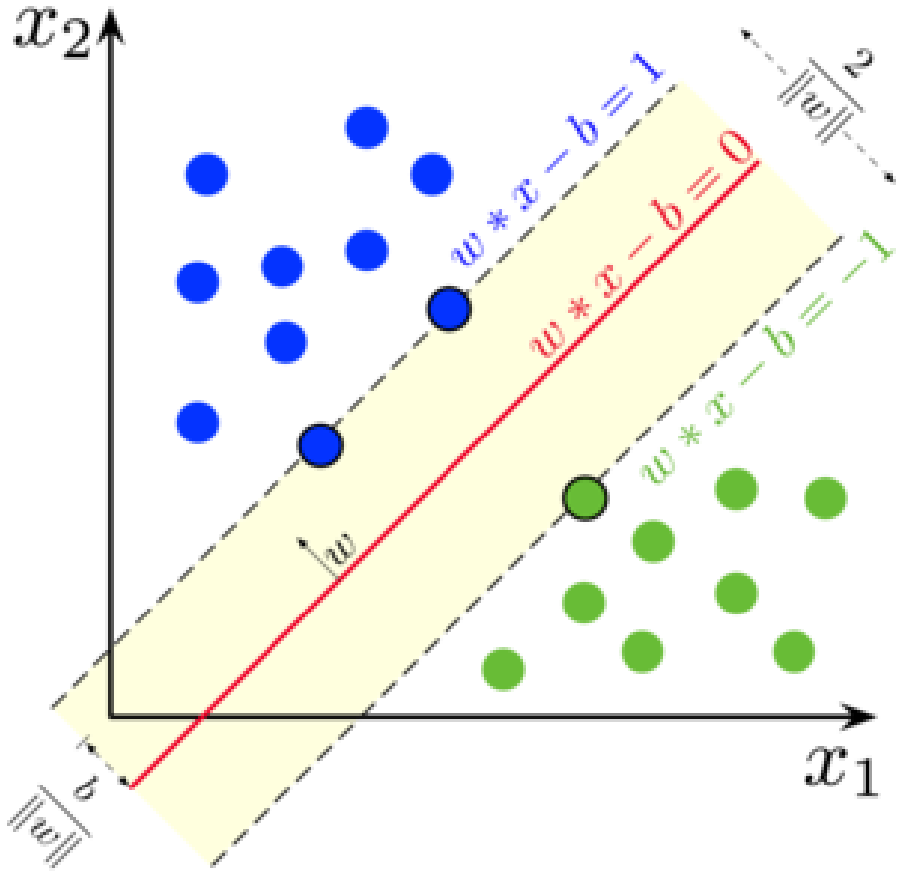
Support Vector Machines



Which one do you prefer?

Support Vector Machines

SVM provides a solution based on the idea of maximising the **margin** between the closest points of the classes.



Support Vector Machines

- Suppose that we find among all the points of the two classes those that are the most critical because they are the closest.
- We draw two hyperplanes over these points and define the discriminant function as the hyperplane in between.
- The equation of this hyperplane is

$$\mathbf{w}^T\mathbf{x} + b = 0$$

where $\mathbf{w}$ is a vector orthogonal to the hyperplane and b is an offset parameter.

- The other two hyperplanes parallel to the first one are denoted by

$$\mathbf{w}^T\mathbf{x} + b = \gamma$$

and

$$\mathbf{w}^T\mathbf{x} + b = -\gamma$$

Support Vector Machines

However, we can normalize just the hyperplane equation:

$$c\left(\mathbf{w}^T\mathbf{x}+b\right)=0$$

where c is an arbitrary constant.
Let us choose this constant $c=\gamma$, so the two parallel hyperplanes become

$$\mathbf{w}^T\mathbf{x}+b=\pm 1$$

Clearly, the intention is to design $\mathbf{w}^T, b$ so that

$$\mathbf{w}^T\mathbf{x}+b\geq 1\Rightarrow y_i=1 \qquad \mathbf{w}^T\mathbf{x}+b\leq -1\Rightarrow y_i=-1$$

Support Vector Machines

The aim is then to maximise the margin, which corresponds to the distance between the points and the decision hyperplane:

$$\frac{y_i f(\mathbf{x}_i)}{\|\mathbf{w}\|} = \frac{y_i(\mathbf{w}^T\mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

which can be rewrite as [3]:

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \text{ Subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$

Support Vector Machines

The aim is then to maximise the margin, which corresponds to the distance between the points and the decision hyperplane:

$$\frac{y_i f(\mathbf{x}_i)}{\|\mathbf{w}\|} = \frac{y_i(\mathbf{w}^T\mathbf{x}_i + b)}{\|\mathbf{w}\|}$$

which can be rewrite as [3]:

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \text{ Subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$

In practice, we must relax the restrictions because the problem cannot be linearly separable.

$$\begin{gathered}\arg\min_{\mathbf{w},b,\zeta_i \geq 0} \frac{1}{2}\|\mathbf{w}\|_2^2 + \alpha \sum_{i=1}^{n} \zeta_i \\ s.t. : y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \zeta_i \\ \zeta_i \geq 0\end{gathered}$$

Hinge loss

If we transform the inequality constraints in an approximate unconstrained problem, we get:

$$y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)=1-\zeta_i \rightarrow \zeta_i=\max\left\{0,1-y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)\right\}$$

so, we have:

$$\underset{\mathbf{w}\in\mathbb{R}^d,b\in\mathbb{R}}{\arg\min}\left(\frac{1}{2}\|\mathbf{w}\|_2^2+\alpha\sum_{i=1}^{n}\max\left(1-y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right),0\right)\right)$$

Hinge loss

If we transform the inequality constraints in an approximate unconstrained problem, we get:

$$y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)=1-\zeta_i \rightarrow \zeta_i=\max\left\{0,1-y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right)\right\}$$

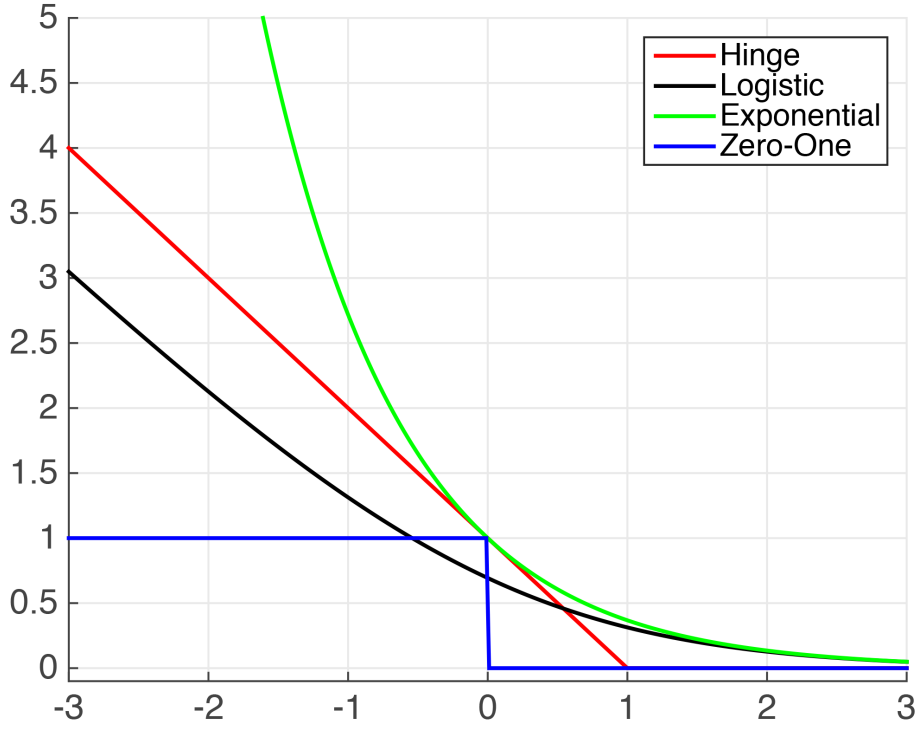
so, we have:

$$\underset{\mathbf{w}\in\mathbb{R}^d,b\in\mathbb{R}}{\arg\min}\left(\frac{1}{2}\|\mathbf{w}\|_2^2+\alpha\sum_{i=1}^{n}\max\left(1-y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right),0\right)\right)$$

where α intends to penalize deviations from the feasibility region. It could also be rewritten as:

$$\underset{\mathbf{w}\in\mathbb{R}^d,b\in\mathbb{R}}{\arg\min}\left(\frac{1}{n}\sum_{i=1}^{n}\max\left(1-y_i\left(\mathbf{w}^T\mathbf{x}_i+b\right),0\right)+\frac{\lambda}{2}\|\mathbf{w}\|_2^2\right)$$

SVM loss function



As we have already mentioned, in practice, we will use an equivalent definition by compacting model parameters:

$$\underset{\mathbf{w}\in\mathbb{R}^{d+1}}{\arg\min}\left(\frac{1}{n}\sum_{i=1}^{n}\max\left(1-y_i\left(\mathbf{w}^T\mathbf{x}_i\right),0\right)^p+\frac{\lambda}{2}\left\|\mathbf{w}\right\|_2^2\right)$$

Code repository

Some functions must be reviewed in detail:

- **Generation**: `Get_data_reg`, `Scenarios_regression`, `Get_data_class`, `Scenarios_classification`. (Have a look at Examples 2_3 and 2_5)

Code repository

Some functions must be reviewed in detail:

- **Generation**: `Get_data_reg`, `Scenarios_regression`, `Get_data_class`, `Scenarios_classification`. (Have a look at Examples 2_3 and 2_5)
- **utils**: `solver_cvx`, `plot_surface`, `test_phase_reg`, `test_phase_class`. (Have a look at Examples 2_6, 2_7, 2_8, 2_9)

Code repository

Some functions must be reviewed in detail:

- **Generation**: `Get_data_reg`, `Scenarios_regression`, `Get_data_class`, `Scenarios_classification`. (Have a look at Examples 2_3 and 2_5)
- **utils**: `solver_cvx`, `plot_surface`, `test_phase_reg`, `test_phase_class`. (Have a look at Examples 2_6, 2_7, 2_8, 2_9)
- **Case_studies**: Compilation of topics.

Code repository

Some functions must be reviewed in detail:

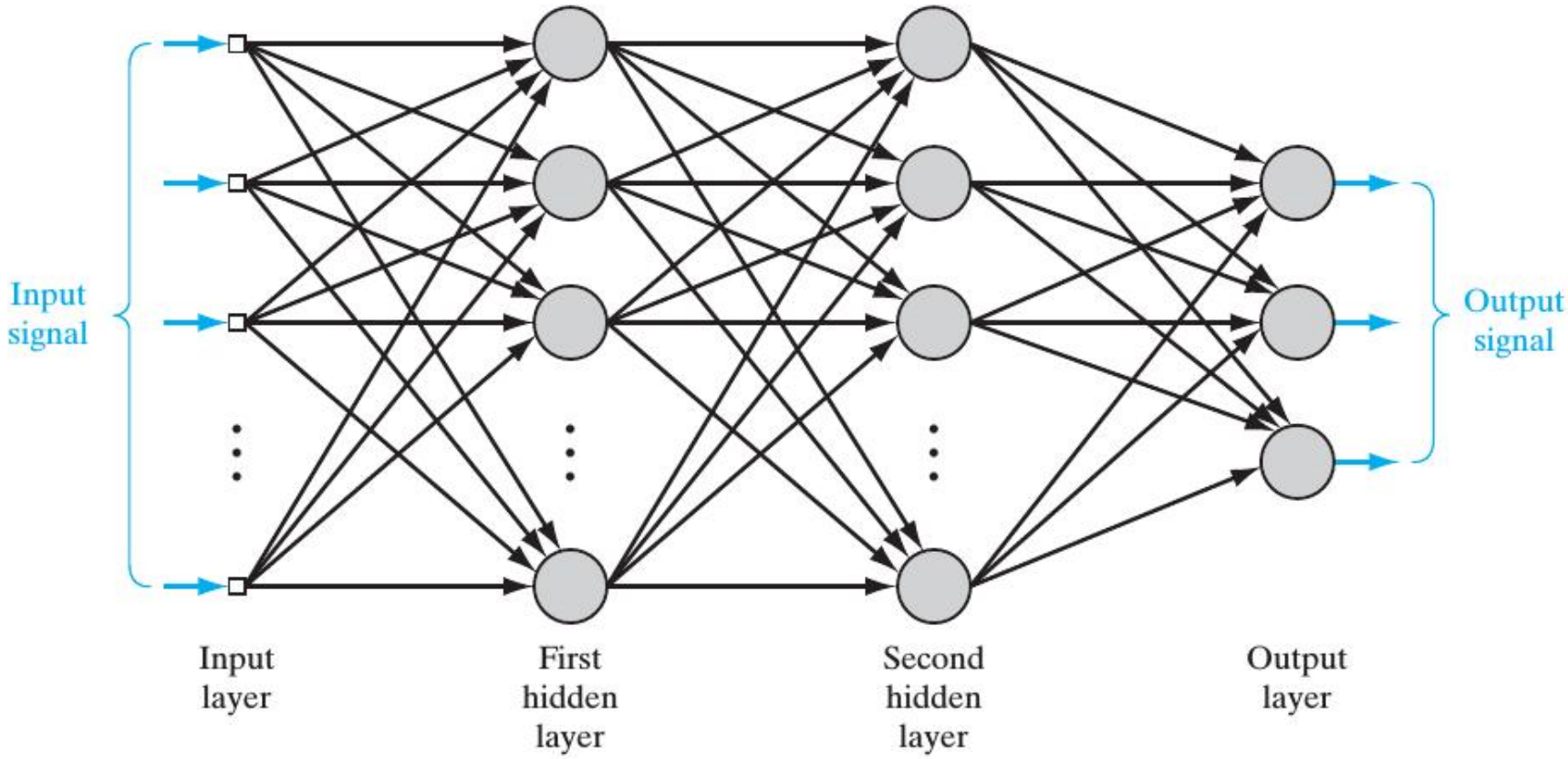
- **Generation**: `Get_data_reg`, `Scenarios_regression`, `Get_data_class`, `Scenarios_classification`. (Have a look at Examples 2_3 and 2_5)
- **utils**: `solver_cvx`, `plot_surface`, `test_phase_reg`, `test_phase_class`. (Have a look at Examples 2_6, 2_7, 2_8, 2_9)
- **Case_studies**: Compilation of topics.
  - `case_study_2_1` (Regression): Understand how the data is generated, training and testing datasets, and the effect of regularization in the error surface.

Code repository

Some functions must be reviewed in detail:

- **Generation**: `Get_data_reg`, `Scenarios_regression`, `Get_data_class`, `Scenarios_classification`. (Have a look at Examples 2_3 and 2_5)
- **utils**: `solver_cvx`, `plot_surface`, `test_phase_reg`, `test_phase_class`. (Have a look at Examples 2_6, 2_7, 2_8, 2_9)
- **Case_studies**: Compilation of topics.
  - `case_study_2_1` (Regression): Understand how the data is generated, training and testing datasets, and the effect of regularization in the error surface.
  - `case_study_2_2` (Classification): What can we expect if the class means are asymmetric? See the effect of regularization in the loss function.

Neural networks. General architecture



Questions?

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Richard O Duda, Peter E Hart, et al. *Pattern classification*. John Wiley & Sons, 2000.

[3] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

Thank You

Julián D. Arias-Londoño
julian.arias@upm.es