Linear classification

R. Flamary

January 18, 2019

Course overview

Course objective

Introduction

- Binary linear classification.
- Convex optimization.

Linear classification methods

- Logistic regression.
- Rosenbtatt's Perceptron.
- Support Vector Machines.

Convex optimization

- Gradient descent.
- Newton's descent.
- Stochastic gradients.

We will focus on binary classification.

Linear Prediction

Linear function

Function $f : \mathbb{R}^d \to \mathbb{R}$, can be expressed as

$$f(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i + b = \mathbf{x}^\top \mathbf{w} + b = [\mathbf{x}^\top 1]\boldsymbol{\alpha} \tag{1}$$

with $\mathbf{w} \in \mathbb{R}^d$ a vector defining an hyperplane in $\mathbb{R}^d$ et $b \in \mathbb{R}$ a bias term displacing the function along the normal $\mathbf{w}$ of the hyperplane. All parameters can be stored in a unique vector $\boldsymbol{\alpha} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$ of dimensionality $\mathbb{R}^{d+1}$ concatenating $\mathbf{w}$ and b.

Objective of linear prediction

- Regression: $f(\cdot) \in \mathbb{R}$.
- Classification: $\text{sign}(f(\cdot)) \in \{-1, 1\}$.

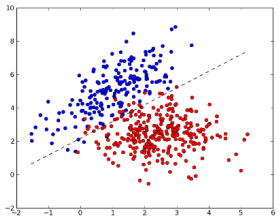
Linear classification

Objective

- Train a linear function $f(\cdot)$ that predicts a binary value $y \in \{-1, 1\}$ from an observation $\mathbf{x} \in \mathbb{R}^d$.
- In practice, we seek to estimate the coefficients $(\mathbf{w}, b)$ of $f(\cdot)$ using the training data $\{\mathbf{x}_i, y_i\}_{i=1,\dots,n}$.
- The predicted class is selected as the sign of function $f(\cdot)$

Exemples

- Optical character recognition.
- Computer Aided Diagnosis.
- Quality inspection.



How do we store training data ?

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top & 1 \\ \mathbf{x}_2^\top & 1 \\ \vdots & \vdots \\ \mathbf{x}_i^\top & 1 \\ \vdots & \vdots \\ \mathbf{x}_n^\top & 1 \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1j} & \dots & x_{1d} & 1 \\ x_{21} & x_{22} & \dots & x_{2j} & \dots & x_{2d} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{i1} & x_{i2} & \dots & x_{ij} & \dots & x_{id} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nj} & \dots & x_{nd} & 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{bmatrix}$$

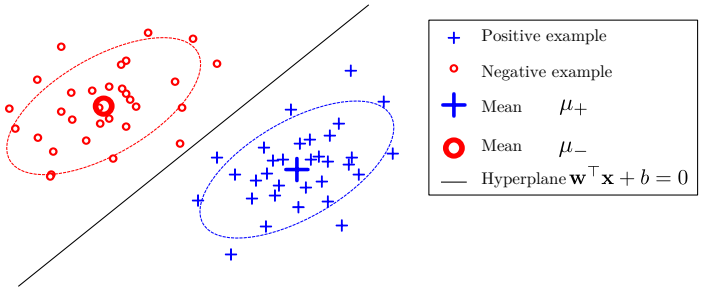
Training data

- $\mathbf{x}_i \in \mathbb{R}^d$ observations for $i = 1, \dots, n$.
- $y_i \in \mathbb{R}$ values to predict for $i = 1, \dots, n$.

Matrix form:

- $\mathbf{X} \in \mathbb{R}^{n \times (d+1)}$ such that $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n, \mathbf{e}]^\top$ with $\mathbf{e} \in \mathbb{R}^d$ and $e_i = 1, \forall i$
- $\mathbf{y} \in \mathbb{R}^n$ such that $\mathbf{y} = [y_1, y_2, \dots, y_n]^\top$.
- $\boldsymbol{\alpha} \in \mathbb{R}^{d+1}$ is a vector such that $\boldsymbol{\alpha} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$

Linear Discriminant Analysis (LDA)



- Bayesian decision method using likelihood ratio for predicting a class.
- We suppose that samples are drawn from Gaussian distributions $\mathcal{N}(\boldsymbol{\mu}_1, \Sigma)$ for class ω_1 and $\mathcal{N}(\boldsymbol{\mu}_2, \Sigma)$ for class ω_2.
- p_1 and p_2 are the probability of a sample being positive and negative.
- The decision function is linear thanks to the shared covariance Σ between the classes.

Likelihood ratio

Decision function

- The conditional probabilities for each class are

$$P(\mathbf{x}|\omega_1) = \det(2\pi\boldsymbol{\Sigma})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_1)\right)$$

- We take as decision function the result of the likelihood ratio.
- If $P(\omega_1|\mathbf{x}) > P(\omega_2|\mathbf{x})$ then choose ω_1 else ω_2:

$$P(\omega_1|\mathbf{x}) \underset{\omega_2}{\overset{\omega_1}{\gtrless}} P(\omega_2|\mathbf{x}) \quad \Leftrightarrow \quad P(\mathbf{x}|\omega_1)P(\omega_1) \underset{\omega_2}{\overset{\omega_1}{\gtrless}} P(\mathbf{x}|\omega_2)P(\omega_2)$$

- The decision is the function f such that :

$$f(\mathbf{x}) = \log\left(\frac{P(\omega_1|\mathbf{x})}{P(\omega_2|\mathbf{x})}\right) = \log\left(\frac{P(\mathbf{x}|\omega_1)P(\omega_1)}{P(\mathbf{x}|\omega_2)P(\omega_2)}\right)$$

The function will be positive if $P(\omega_1|\mathbf{x}) > P(\omega_2|\mathbf{x})$ else negative.
Its sign recovers the likelihood ratio decision.

Decision function

$$
\begin{aligned}
f(\mathbf{x}) &= \log\left(\frac{P(\omega_1|\mathbf{x})}{P(\omega_2|\mathbf{x})}\right) = \log\left(\frac{P(\mathbf{x}|\omega_1)P(\omega_1)}{P(\mathbf{x}|\omega_2)P(\omega_2)}\right) \\
&= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_2) + \log(p_1) - \log(p_2) \\
&= \mathbf{x}^\top \Sigma^{-1}\boldsymbol{\mu}_1 - \frac{1}{2}\boldsymbol{\mu}_1^\top \Sigma^{-1}\boldsymbol{\mu}_1 - \mathbf{x}^\top \Sigma^{-1}\boldsymbol{\mu}_2 + \frac{1}{2}\boldsymbol{\mu}_2^\top \Sigma^{-1}\boldsymbol{\mu}_2 + \log(p_1) - \log(p_2) \\
&= \mathbf{x}^\top \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) - \frac{1}{2}\boldsymbol{\mu}_1^\top \Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^\top \Sigma^{-1}\boldsymbol{\mu}_2 + \log(p_1) - \log(p_2) \\
&= \mathbf{x}^\top \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + \frac{1}{2}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)^\top \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + \log(p_1) - \log(p_2) \\
&= \mathbf{x}^\top \mathbf{w} + b
\end{aligned}
$$

with

$$\mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2), \quad \text{and } b = \frac{1}{2}\mathbf{w}^\top(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2) + \log(p_1) - \log(p_2)$$

Regularized LDA

LDA parameter estimation

$$\mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2), \quad \text{and } b = \frac{1}{2}\mathbf{w}^\top(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2) + \log(p_1) - \log(p_2)$$

- Σ, $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, p_1, p_2$ often unknown, estimated from dataset.
- Σ can be non invertible (not enough samples).
- Risk of overfitting.

Regularized LDA parameter estimation

$$\mathbf{w} = (\Sigma + \lambda \mathbf{I})^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2), \quad \text{and } b = \frac{1}{2}\mathbf{w}^\top(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2) + \log(p_1) - \log(p_2)$$

- $\mathbf{I}$ is the identity matrix and $\lambda \geq 0$ a regularization parameter.
- Makes the matrix invertible and the solution unique.

Fisher Discriminant Analysis

Multiclass LDA

- We suppose that $\forall k$ samples from class k are drawn from $\mathcal{N}(\boldsymbol{\mu}_k, \Sigma)$
- We define matrix Σ_b such that for all classes $1, \ldots, C$

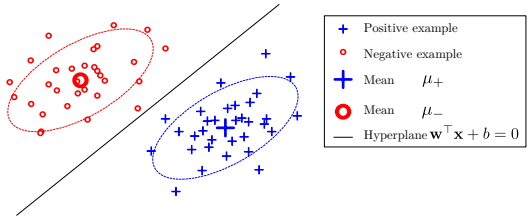
$$\Sigma_b = \frac{1}{C}\sum_{k=1}^{C}(\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^\top \quad \text{where} \quad \boldsymbol{\mu} = \frac{1}{C}\sum_{k=1}^{C}\boldsymbol{\mu}_k$$

Fisher Discriminant Analysis

$$\max_{\mathbf{w}, \|\mathbf{w}\|=1} \frac{\mathbf{w}^\top \Sigma_b \mathbf{w}}{\mathbf{w}^\top \Sigma \mathbf{w}}$$

- We seek for a projection $\mathbf{w}$ that maximize the distance between classes while minimizing the variance of each class.
- Solutions of the problem are the eignevectors of $\Sigma^{-1}\Sigma_b$.
- Special case with two class is exactly the solution of Binary LDA.

Conclusion on LDA



Pros

- Probabilistic model, probabilities can be estimated.
- Closed form solution when the distribution parameters are known
- Regularization helps avoiding over-fitting.
- Extension to multiclass with Fisher Discriminant.

Cons

- Gaussian distribution parameters have to be estimated.
- All classes are supposed to have the same covariance matrix Σ.

Logistic regression

Objective

- Train a linear discriminant function.
- Model directly conditional probabilities (predict probabilities).
- Avoid parameter estimations for distributions such as Gaussians.

Approach

- We suppose a conditional probability $P(\omega_1|\mathbf{x})$ of the form:

$$P(\omega_1|\mathbf{x}) = \frac{\exp(\mathbf{w}^\top \mathbf{x} + b)}{1 + \exp(\mathbf{w}^\top \mathbf{x} + b)} = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x} - b)} \quad (2)$$

and so

$$P(\omega_2|\mathbf{x}) = 1 - P(\omega_1|\mathbf{x}) = \frac{1}{1 + \exp(\mathbf{w}^\top \mathbf{x} + b)} \quad (3)$$

Likelihood ratio

Decision function

- We take as decision function the result of the likelihood ratio.
- If $P(\omega_1|\mathbf{x}) > P(\omega_2|\mathbf{x})$ then choose ω_1 else ω_2:

$$\text{ou} \quad P(\omega_1|\mathbf{x}) \underset{\omega_2}{\overset{\omega_1}{\gtrless}} P(\omega_2|\mathbf{x})$$

- The decision is function f such that :

$$f(\mathbf{x}) = \log\left(\frac{P(\omega_1|\mathbf{x})}{P(\omega_2|\mathbf{x})}\right) = \log(\exp(\mathbf{w}^\top \mathbf{x} + b)) = \mathbf{w}^\top \mathbf{x} + b$$

- Its sign recovers the decision of the likelihood ratio.

Optimization problem

Log-likelihood

We want to maximize the log-likelihood on the data, which means minimizing:

$$J(\mathbf{w}, b) = -\log\left(\prod_i P(y_i|\mathbf{x}_i)\right) = -\sum_{i\in\mathcal{I}_1} \log(P(\omega_1|\mathbf{x}_i)) - \sum_{i\in\mathcal{I}_2} \log(P(\omega_2|\mathbf{x}_i)) \quad (4)$$

where $\mathcal{I}_1$ and $\mathcal{I}_2$ are the set of examples from class ω_1 and ω_2 respectively.

Objective function

We define the following objective function:

$$\begin{aligned} J(\mathbf{w}, b) &= \sum_{i\in\mathcal{I}_1} \log(1 + \exp(-\mathbf{w}^\top \mathbf{x}_i - b)) + \sum_{i\in\mathcal{I}_2} \log(1 + \exp(\mathbf{w}^\top \mathbf{x}_i + b)) \\ &= \sum_i \log(1 + \exp(-y_i(\mathbf{w}^\top \mathbf{x}_i + b))) \end{aligned} \quad (5)$$

Gradient computation

- Function $J(\mathbf{w}, b)$ is convex and differentiable. In order to compute its gradient we reformulate it as:

$$J(\boldsymbol{\alpha}) = \sum_i \log(1 + \exp(-y_i \boldsymbol{\alpha}^\top \tilde{\mathbf{x}}_i)) \quad (6)$$

- Partial derivative of $J(\boldsymbol{\alpha})$ with respect to $\boldsymbol{\alpha}_j$ is

$$\frac{\partial J(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}_j} = \sum_i \frac{-y_i(\tilde{\mathbf{x}}_i)_j \exp(-y_i \boldsymbol{\alpha}^\top \tilde{\mathbf{x}}_i)}{1 + \exp(-y_i \boldsymbol{\alpha}^\top \tilde{\mathbf{x}}_i)} = \sum_i \frac{-y_i(\tilde{\mathbf{x}}_i)_j p_i}{1 + p_i} \quad (7)$$

with $p_i = \exp(-y_i \boldsymbol{\alpha}^\top \tilde{\mathbf{x}}_i)$

- Which leads to this gradient formulation in its matrix form:

$$\nabla_\alpha J(\boldsymbol{\alpha}) = -\mathbf{X}^\top \mathbf{P}\mathbf{y} \quad (8)$$

where $\mathbf{P}$ is a diagonal matrix of diagonal elements $\frac{p_i}{1+p_i}$ that depends on $\boldsymbol{\alpha}$. $\nabla_\alpha J(\boldsymbol{\alpha}) = \mathbf{0}$ defines a non linear equation that cannot be solved with a closed form $\rightarrow$ Iterative optimization method.

Steepest gradient descent (GD)

Iterative optimization method

- Principle : update an approximate solution at each iteration.
- At iteration t the solution is updated with:

$$\boldsymbol{\alpha}^{(t)} = \boldsymbol{\alpha}^{(t-1)} + \mu_t \mathbf{d}_t \tag{9}$$

where $\mathbf{d}_t$ is a descent direction which means that $\mathbf{d}_t^\top \nabla_\alpha J(\boldsymbol{\alpha}^{(t-1)}) < 0$ and $\mu_t > 0$ is the descent step.

Steepest gradient descent (GD)

- We take $\mathbf{d}_t = -\nabla_\alpha J(\boldsymbol{\alpha}^{(t-1)})$, that is obviously a descent direction (the steepest).
- Step μ_t has to be chosen small enough to ensure descent.
- Each iteration decrease the objective value but can converge slowly.

Steepest gradient descent (2)

Algorithm of steepest GD

Initialization of $\boldsymbol{\alpha}$, μ
repeat
  $\mathbf{d} \leftarrow -\nabla J(\boldsymbol{\alpha})$
  $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \mu\mathbf{d}$
until Convergence

Algo. for log. reg.

Initialization of $\boldsymbol{\alpha}$, μ and $\mathbf{P} = \mathbf{I}$
repeat
  Compute $\mathbf{P}$ for current $\boldsymbol{\alpha}$
  $\mathbf{d} \leftarrow \mathbf{X}^\top \mathbf{P}\mathbf{y}$
  $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \mu\mathbf{d}$
until Convergence

Discussion

- Sensitive to initialization of $\boldsymbol{\alpha}$.
- We can ensure de decrease of the objective function at each iteration with a linesearch:

Backtracking method
Initialization of μ of $0 < \rho < 1$.
repeat
  $\mu \leftarrow \rho\mu$
until $J(\boldsymbol{\alpha} + \mu\mathbf{d}) < J(\boldsymbol{\alpha})$

- Convergence conditions (when to stop) is discussed later.

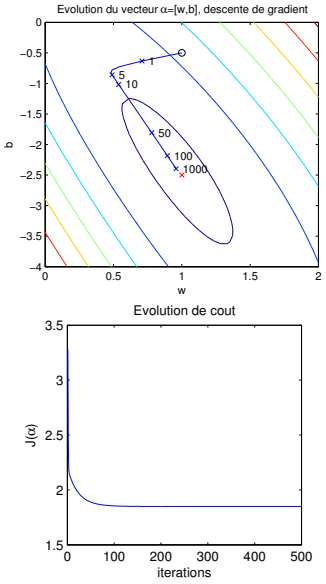
Example of steepest descent

Simulation

- Regularized logistic regression.
- Steepest gradient descent.
- Data (x_i, y_i) with $d = 1$: $(1, -1), (2, -1), (3, 1), (4, 1)$
- $\mu = 0.1, \lambda = 1$
- 1000 iterations
- Initialization $\boldsymbol{\alpha}_0 = [1, -0.5]$
- Problem solution : $\boldsymbol{\alpha}^* = [1, -2.5]$

Discussion

- Slow convergence around the solution.
- After 1000 iterations, still not converged.
- Complexity $\mathcal{O}(nd)$ per iteration.



Hessian matrix

- The Hessian matrix of a differentiable function is the matrix $\mathbf{H} \in \mathbb{R}^{d+1 \times d+1}$ such that

$$H_{u,v} = \frac{\partial^2 J(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}_u \partial \boldsymbol{\alpha}_v} \tag{10}$$

It contains all the second order derivatives of the multivariate function J.

- For the logistic regression we have

$$\frac{\partial^2 J(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}_u \partial \boldsymbol{\alpha}_v} = \sum_i \frac{(\tilde{\mathbf{x}}_i)_u (\tilde{\mathbf{x}}_i)_v \exp(-y_i \boldsymbol{\alpha}^\top \tilde{\mathbf{x}}_i)}{(1 + \exp(-y_i \boldsymbol{\alpha}^\top \tilde{\mathbf{x}}_i))^2} = \sum_i \frac{(\tilde{\mathbf{x}}_i)_u (\tilde{\mathbf{x}}_i)_v p_i}{(1 + p_i)^2} \tag{11}$$

- Which can be expressed in matrix form as :

$$\mathbf{H} = \mathbf{X}^\top \tilde{\mathbf{P}} \mathbf{X} \tag{12}$$

where $\tilde{\mathbf{P}}$ is a diagonal matrix of diagonal element $\frac{p_i}{(1+p_i)^2}$.

Newton's gradient decsent

Principle

- Minimize a quadratic approximation $\tilde{J}(\boldsymbol{\alpha})$ of the function $J(\boldsymbol{\alpha})$ à each iteration.
- Minimizing this approximation is equivalent to taking $\mathbf{d} = -\mathbf{H}^{-1}\nabla_\alpha J(\boldsymbol{\alpha}^{(t-1)})$ as direction.
- If the function J is convex , $\mathbf{H}$ is positive definite and $\mathbf{d}$ is provably a descent direction.

Newtons's gradient descent

Initialization of $\boldsymbol{\alpha}$, μ et $\mathbf{P} = \mathbf{I}$ et $\tilde{\mathbf{P}} = \mathbf{I}$
repeat
  Update$\mathbf{P}$ and $\tilde{\mathbf{P}}$
  $\mathbf{d} \leftarrow (\mathbf{X}^\top \tilde{\mathbf{P}} \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{P}\mathbf{y}$
  $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \mu\mathbf{d}$
until Convergence

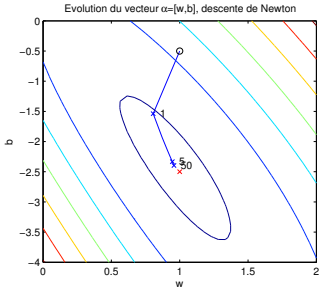
Discussion

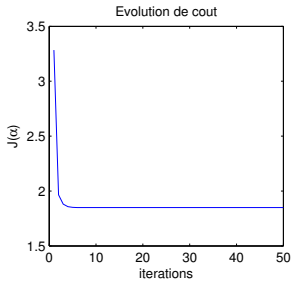
- Better convergence speed.
- Needs computation and inverse of Hessian.
- Iteration much more complex than steepest.
- If the problem is quadratic, algorithm converges in one step.

Example for Newton descent

Simulation

- Regularized logistic regression.
- Newton's descent.
- Data (x_i, y_i) with $d = 1$: $(1, -1), (2, -1), (3, 1), (4, 1)$
- $\mu = 0.1, \lambda = 1$
- 1000 iterations
- Initialization $\boldsymbol{\alpha}_0 = [1, -0.5]$
- Problem solution : $\boldsymbol{\alpha}^* = [1, -2.5]$





Discussion

- Converges quickly to the solution.
- After 5 iterations, same position as 100 with steepest decsnet.
- Complexity $\mathcal{O}(nd^2 + d^3)$ per iteration.
- Quasi-Newton avoid matrix inverse.

Convergence and stopping conditions

Convergence

- For an iterative method, when to stop?.
- Convergence of the algorithm is proved under mild conditions (Nocedal, Convex Optimization)
- A stationary point is reached by the algorithm if:

$$\nabla_\alpha J(\boldsymbol{\alpha}) = \mathbf{0} \tag{13}$$

Stopping conditions

In practice iterations are stopped with those conditions:

- Norm of the gradient below a threshold : $\|\nabla_\alpha J(\boldsymbol{\alpha})\| < \epsilon$
- Relative variation of the objective value below a threshold :

$$\frac{|J(\boldsymbol{\alpha}^t) - J(\boldsymbol{\alpha}^{t-1})|}{J(\boldsymbol{\alpha}^{t-1})|} < \epsilon$$

- Maximum number of iterations t_{max} reached .

Régularization

A priori on w

- With an *a priori* $P(\mathbf{w})$ of the probability distribution of $\mathbf{w}$ can be easily added to the log-likelihood.
- If we suppose that $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ then we have

$$P(\mathbf{w}) = e^{-\frac{\|\mathbf{w}\|^2}{2\sigma^2}} \tag{14}$$

- Maximizing the log-likelihood is then equivalent to minimizing:

$$\begin{aligned} J(\mathbf{w}, b) &= -\log\left(P(\mathbf{w})\prod_i P(y_i|\mathbf{x}_i)\right) = -\log\left(P(\mathbf{w})\right) - \log\left(\prod_i P(y_i|\mathbf{x}_i)\right) \\ &= \sum_i \log(1 + \exp(-y_i \boldsymbol{\alpha}^\top \tilde{\mathbf{x}}_i)) + \frac{1}{2\sigma^2}\|\mathbf{w}\|^2 \end{aligned} \tag{15}$$

  The additionnal term is exactly the ridge regularization with $\lambda = \frac{1}{\sigma^2}$
- Other *a priori* about $\mathbf{w}$ would lead to a different optimization problem.

Conclusions about logistic regression

Pros

- Probabilistic modelisation, conditional probability can be estimated.
- Convex problem, strictly convex with regularization.
- Regularization helps avoiding over-fitting.
- Less parameter to estimate than Bayesian approaches such as LDA

$$d+1 \ll \underbrace{d^2+2d+2}_{\text{LDA}}$$

Cons

- Non-linear problem to optimize and interpret.
- Iterative methods such as gradient descent are necessary.

Method of Perceptron

History

- Perceptron was proposed in 1957 by Frank Rosenblatt.
- First very simple neuron (linear).

$$f(\mathbf{x}) = \sum_k w_k x_k + b$$

- Able to train only on separable data.

Principle

- Seek for an hyperplane defined by $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b = 0$ separating the classes.
- Iterative method with very small complexity per iteration.
- Update $(\mathbf{w}, b)$ on mis-classified examples.
- Stop the iterations when all examples are well classified.

Optimization problem

Objective function

The percetron is equivalent to minimizing:

$$J(\boldsymbol{\alpha}) = J(\mathbf{w}, b) = -\sum_{i \in \mathcal{M}} y_i(\mathbf{x}_i^\top \mathbf{w} + b) = \sum_i \max(0, -y_i(\mathbf{x}_i^\top \mathbf{w} + b)) \qquad (16)$$

where $\mathcal{M}$ is the set of all misclassified examples.

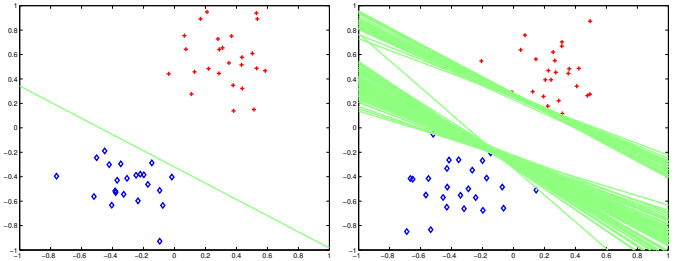
Algorithm of perceptron

Initialization of $\boldsymbol{\alpha}$ and $\mu > 0$
repeat
  for $i \in \mathcal{I}$ **do**
    if $y_i \tilde{\mathbf{x}}_i^\top \boldsymbol{\alpha} < 0$ **then**
      $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \mu y_i \tilde{\mathbf{x}}_i$
    end if
  end for
until $y_i \tilde{\mathbf{x}}_i^\top \boldsymbol{\alpha} \geq 0, \quad \forall i$

Discussion

- $\mathcal{I}$ define the order in which the examples are selected.
- Each iteration compute the gradient for a unique example.
- Algorithm of type "Stochastic Gradient Descent" (SGD).
- Converges in a finite number of iterations if the data is separable.

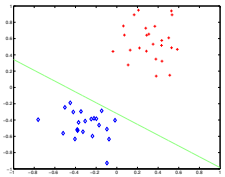
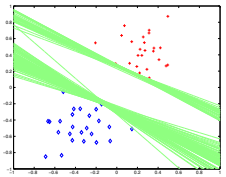
Example of perceptron solutions



Discussion

- Each green line is a solution.
- Final solution depends on initialization and order of sample update.
- Note that the hyperplane are often close to one class or the other.
- Do those solutions generalize well?

Conclusion on the perceptron

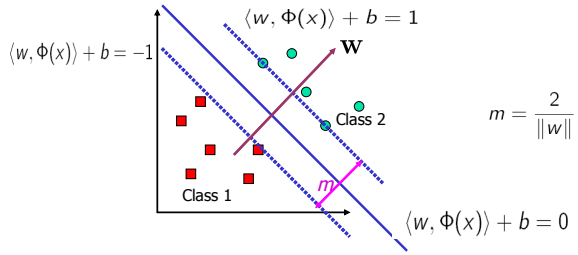
Avantages

- Historical methods.
- Find a solution in a finite number of iteration for separable data.
- Iterations are very cheap (SGD is still used a lot).

Inconvénients

- No unique solution
- No convergence on non separable data.
- Risque of overfitting since no regularization.
- Bad performances proved in multi-class.

Support vector machines



$$m = \frac{2}{\|w\|}$$

Principle

- Find the hyperplane that maximize the margin between the classes
- We want tye samples to be well classified with a margin, leading to the following constraints:

$$y_i(\mathbf{w}^\top \mathbf{x} + b) \geq 1 \quad \forall i$$

Optimization problem

- Distance of a sample to the hyperplane is defined as

$$d(\mathbf{x}) = \frac{|\mathbf{w}^\top \mathbf{x} + b|}{\|\mathbf{w}\|}$$

- Constraints $y_i(\mathbf{w}^\top \mathbf{x} + b) \geq 1$ ensure that the minimal distance of the samples to the hyperplane is equal to $\frac{1}{\|\mathbf{w}\|}$. The margin is then equal to

$$m = \frac{2}{\|\mathbf{w}\|} \tag{17}$$

- Maximizing the margin is then equivalent to minimizing $\|\mathbf{w}\|$ (also $\|\mathbf{w}\|^2$).
- The final support vector machine optimization problem is then :

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \|\mathbf{w}\|^2 \\ & y_i(\mathbf{w}^\top \mathbf{x} + b) \geq 1 \quad \forall i \end{aligned} \tag{18}$$

smaples eaxctly on the margin ($\mathbf{w}^\top \mathbf{x}_k + b = y_k$) are called support vectors.

Optimization methods

Non separable data (primal formulation)

When the margin constraints are relaxed the optimization problem becomes:

$$\min_{\mathbf{w},b} \quad \sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)) + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{19}$$

Direct solver in the primal with a gradient descent approach.
$\rightarrow$ Non-differentiable problem of size $d + 1$.

Dual formulation

$$\max_{\beta} \quad \sum_{i=1}^{n} \beta_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i \beta_i (\mathbf{x}_i^\top \mathbf{x}_j) y_j \beta_j, \tag{20}$$

$$\text{subject to } \sum_{i=1}^{n} \beta_i y_i = 0, \text{ and } 0 \leq \beta_i \leq \frac{1}{2\lambda} \text{ for all } i. \tag{21}$$

The hyperplane can be recovered with $\mathbf{w} = \sum_{i=1}^{n} \beta_i y_i \mathbf{x}_i$
$\rightarrow$ Constrained Quadratic Program (QP) of size n.

Kernel Trick

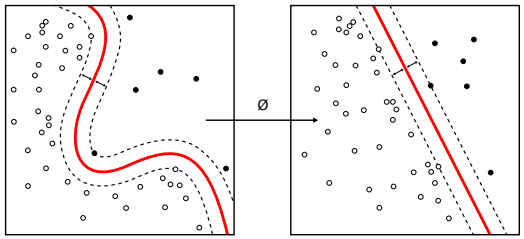
- A kernel is a positive definite function of two samples that can be expressed as a scalar product:
$$k(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1)^\top \phi(\mathbf{x}_2)$$
- The dual formulation of the problem depends only on scalar product and can be expressed with kernels:

$$\max_{\beta} \quad \sum_{i=1}^{n} \beta_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i \beta_i k(\mathbf{x}_i, \mathbf{x}_j) y_j \beta_j, \tag{22}$$

$$\text{subject to } \sum_{i=1}^{n} \beta_i y_i = 0, \text{ and } 0 \leq \beta_i \leq \frac{1}{2n\lambda} \text{ for all } i. \tag{23}$$

- The prediction is then of the form:
$$f(\mathbf{x}) = \sum_{i=1}^{n} \beta_i k(\mathbf{x}, \mathbf{x}_i) + b$$
where $b = y_k - \sum_{i=1}^{n} \beta_i k(\mathbf{x}_k, \mathbf{x}_i)$ can be estimated from a sample k on the margin where $f(\mathbf{x}_k) = y_k$.

Conclusion for the SVM



Avantages

- Optimization problem is strictly convex.
- Consistant method: converges to the Bayes classifier for an infinite number of training samples.
- Can be extended to non-linear classifier thanks to the kernel trick.
- Very good performances in practice even on small datasets.

Cons

- Non differentiable objective function.
- Do not scale well in the dual that is necessary for non-linear kernel formulation.

Regularized linear regression

General problem formulation:

$$\min_{\mathbf{w},b} \quad \sum_{i=1}^{n} L(y_i, \mathbf{w}^\top \mathbf{x}_i + b) + \lambda \Omega(w) \tag{24}$$

With

- $L(\cdots)$ a loss function.
- $\Omega(\cdot)$ a regularization term.

Examples:

Loss function $L(y, \hat{y})$

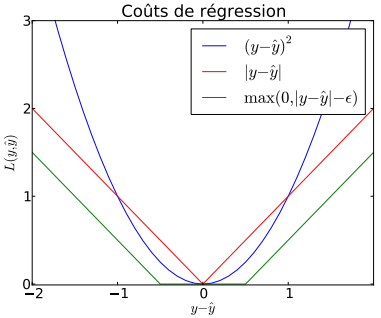
- $(y - \hat{y})^2$, quadratic.
- $|y - \hat{y}|$, absolute value.
- $\min(0, |y - \hat{y}| - \epsilon)$ epsilon insensitive
- $\max(0, 1 - y\hat{y})$, Hinge.
- $\log(1 + e^{-y\hat{y}})$, logistic.

Regularizations $\Omega(\mathbf{w})$

- $\|\mathbf{w}\|_2^2$, quadratic.
- $\|\mathbf{w}\|_1$, ℓ_1 norm.
- $\mathbf{w}^\top \mathbf{\Sigma} \mathbf{w}$, Mahalanobis.

Data fitting for regression

Cost	$L(y, \hat{y})$	Smooth.	Cvx.
Square	$(y - \hat{y})^2$	✓	✓
Absolute value	$\|y - \hat{y}\|$	-	✓
ϵ insensible	$\max(0, \|y - \hat{y}\| - \epsilon)$	-	✓

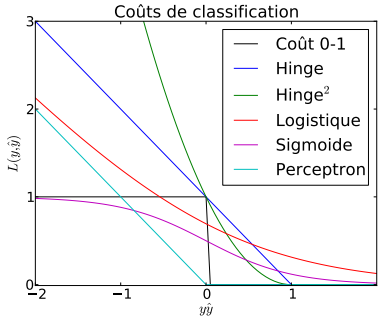


Regression problem

- **Objective**: predict a real value.
- Error if $y \neq \hat{y}$.
- **Error measure**: $|y - \hat{y}|$

Data fitting for classification

Cost	$L(y, \hat{y})$	Smooth.	Cvx.
0-1 loss	$(1 - \text{sgn}(y\hat{y}))/2$	-	-
Hinge	$\max(0, 1 - y\hat{y})$	-	✓
Squared Hinge	$\max(0, 1 - y\hat{y})^2$	✓	✓
Logistic	$\log(1 + \exp(-y\hat{y}))$	✓	✓
Sigmoid	$(1 - \tanh(y\hat{y}))/2$	✓	-
Perceptron	$\max(0, -y\hat{y})$	-	✓



Regression problem

- **Objective**: predict a binary value.
- Error when $y \neq \text{signe}(\hat{y})$ i.e. if y and $\hat{y}$ have a different sign.
- **Error measure**: $y\hat{y}$
- Non symmetric loss.