Computational Statistics
Linear Classification

Luca Bortolussi

Department of Mathematics and Geosciences
University of Trieste

Office 238, third floor, H2bis
`luca@dmi.units.it`

Trieste, Winter Semester 2016/2017

Outline

INTRODUCTION

- Data: $\mathbf{x_i}, t_i$. Output are discrete, either binary or multiclass (K classes), and are also denoted by y_i. Classes are denoted by $\mathcal{C}_1, \ldots, \mathcal{C}_K$.
- Discriminant function: we construct a function $f(\mathbf{x}) \in \{1, \ldots, K\}$ associating with each input a class.
- Generative approach: We consider a prior over classes, $p(\mathcal{C}_k)$, and the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$, from a parametric family. We learn class-conditional densities from data, and then compute the class posterior.

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}$$

- Discriminative approach: we learn directly a model for the class posteriori $p(\mathcal{C}_k|\mathbf{x})$, typically as $p(\mathcal{C}_k|\mathbf{x}) = f(\mathbf{w}\boldsymbol{\phi}(\mathbf{x}))$. f is called an activation function (and f^{-1} a link function).

ENCODING OF THE OUTPUT

- For a binary classification problem, usually we choose $t_n \in \{0, 1\}$. The interpretation is that of a "probability" to belong to class C_1.
- In some circumstances (perceptron, SVM), we will prefer the encoding $t_n \in \{-1, 1\}$.
- For a multiclass problem, we usually stick to a boolean encoding: $\mathbf{t_n} = (t_{n,1}, \ldots, t_{n,K})$, with $t_{n,j} \in \{0, 1\}$, and t_n is in class k if and only if $t_{n,k} = 1$ and $t_{n,j} = 0$, for $j \neq k$.

LINEAR DISCRIMINANT CLASSIFIER

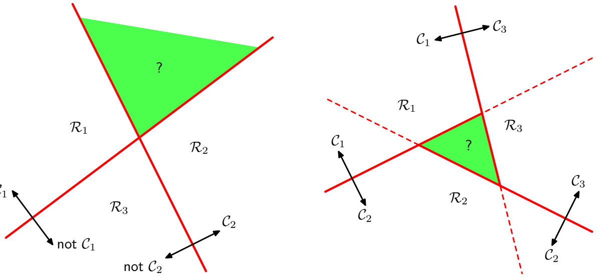
- $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$, decode to class 1 iff $y(\mathbf{x}) > 0$, and to class 0 if $y(\mathbf{x}) < 0$.
- Typically here we use the encoding scheme $t_n \in \{0, 1\}$, but also $t_n \in \{-1, 1\}$ works (different solutions, though).
- Maximum likelihood training like in regression: minimise the sum-of-squares error function

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_i (\mathbf{w}^T\mathbf{x_i} - t_i)^2.$$

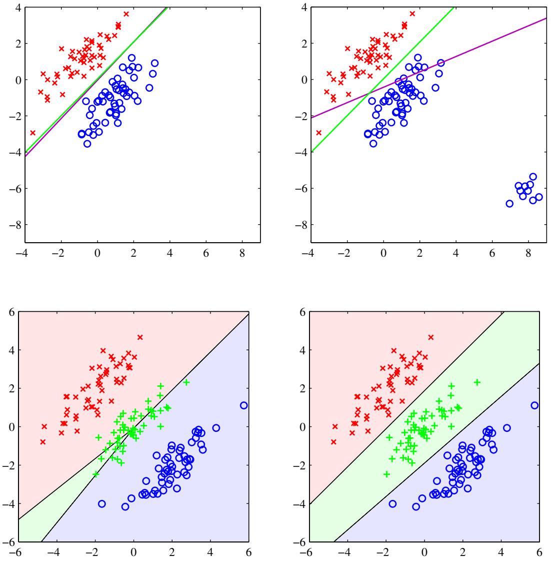
- Solution is $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{t}$.
- The method can be extended to k classes (see next slide), but performs poorly in general, because it tries to approximate a probability in $[0, 1]$ with a real number.

MULTI-CLASS STRATEGIES

- Assume we have a binary classifier. We can train K classifiers, one-versus-the-rest strategy, class C_k versus all other points (unbalanced).
- Alternatively, there is the one-versus-one classifier, trains $K(K-1)/2$ for each pair of classes, decode by majority voting. Both are ambiguous.
- One can train K linear discriminants $y_k(\mathbf{x}) = \mathbf{w_k}^T\mathbf{x} + b_k$ and decode to j such that $y_j(\mathbf{x}) > y_i(\mathbf{x})$ for each $i \neq j$.



LINEAR DISCRIMINANT - EXAMPLE



Comparing linear discriminant with logistic regression, for 2 and 3 classes problems.

FISHER'S DISCRIMINANT

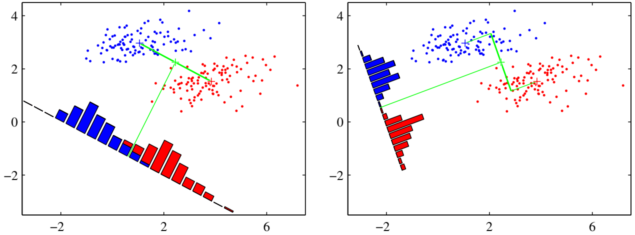
- Idea: project data linearly in one dimension, so to separate as much as possible the two classes. The projection is $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$.
- Choose the projection that (a) maximises the separation between the two classes, either by maximising the projected class means distance, or by maximising the ratio between between-class and within-class variances.
- $\mathbf{m_i} = 1/N \sum_{j \in C_i} \mathbf{x_i}$, $m_i = \mathbf{w}^t\mathbf{m_i}$, class means.
- Between class variance $(m_2 - m_1)^2 = \mathbf{w}^T\mathbf{S_B}\mathbf{w}$, $\mathbf{S_B} = (\mathbf{m_2} - \mathbf{m_1})(\mathbf{m_2} - \mathbf{m_1})^T$
- Within-class variance $\mathbf{w}^T\mathbf{S_W}\mathbf{w}$, $\mathbf{S_W} = \sum_{j \in C_1}(\mathbf{x_j} - \mathbf{m_1})(\mathbf{x_j} - \mathbf{m_1})^T + \sum_{j \in C_2}(\mathbf{x_j} - \mathbf{m_2})(\mathbf{x_j} - \mathbf{m_2})^T$.

FISHER'S DISCRIMINANT

- Maximise the ratio

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S_B} \mathbf{w}}{\mathbf{w}^T \mathbf{S_W} \mathbf{w}}$$

- Deriving and setting the derivative to zero, we get $\mathbf{w} \propto \mathbf{S_W}^{-1}(\mathbf{m_2} - \mathbf{m_1})$.
- Choose the best y_0 that separates the projected data. Classify to $\mathcal{C}_1$ if $y(\mathbf{x}) \geq y_0$. Idea: approximate the projected class distributions $p(y|\mathcal{C}_k)$ as Gaussians and then find y_0 s.t. $p(\mathcal{C}_1|y_0) = p(\mathcal{C}_2|y_0)$, i.e. $p(y_0|\mathcal{C}_1)p(\mathcal{C}_1) = p(y_0|\mathcal{C}_2)p(\mathcal{C}_2)$.



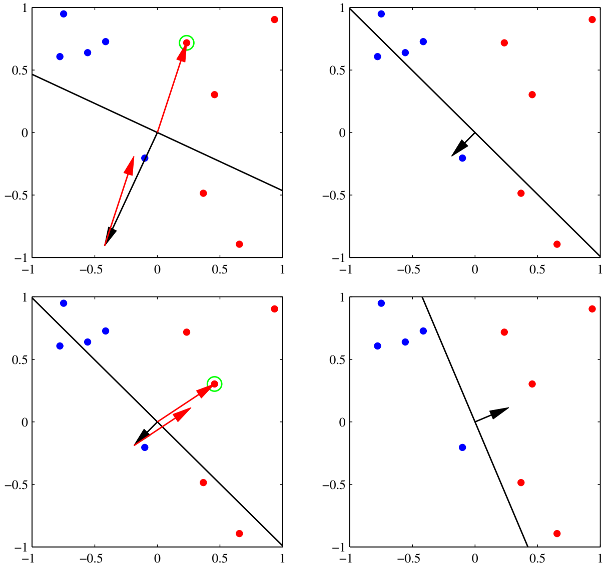
THE PERCEPTRON ALGORITHM

- For binary classes, proposed by Rosenblatt in 62. Typically one maps the input data in a higher dimensional space $\boldsymbol{\phi}(\mathbf{x_i})$, chooses the coding $t_i \in \{-1, 1\}$, and decodes to $\mathcal{C}_1$ if $y(\mathbf{x}) = f(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x_i})) \geq 0$, where the activation function is the step function $f(a) = 1$, if $a \geq 0$ and $f(a) = -1$ if $a < 0$.
- A correctly classified pattern satisfies $\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x_i}) t_i \geq 0$. A misclassified pattern instead $\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x_i}) t_i < 0$.
- We pick as error function $E_P(\mathbf{w}) = -\sum_{i \in \mathcal{M}} \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x_i}) t_i$, which generalises the idea of minimising the number of misclassified patterns $\mathcal{M}$.
- Optimise it by stochastic gradient ascend:

$$\mathbf{w}^{n+1} = \mathbf{w}^n + \eta \boldsymbol{\phi}(\mathbf{x_n}) t_n \mathbb{I}(\mathbf{w}^n \boldsymbol{\phi}(\mathbf{x_n}) t_n < 0)$$

  (typically, $\eta = 1$)
- If the data is linearly separable (in the feature space $\boldsymbol{\phi}$), then the algorithm converges.

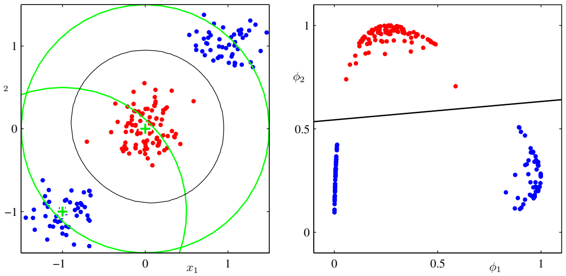
THE PERCEPTRON ALGORITHM



Outline

LOGIT AND PROBIT REGRESSION (BINARY CASE)

- We model directly the conditional class probabilities $p(\mathcal{C}_1|\mathbf{x}) = f(\mathbf{w}^T\phi(\mathbf{x}))$, after a (nonlinear) mapping of the features $\phi(\mathbf{x}) = \phi_1(\mathbf{x}), \ldots, \phi_m(\mathbf{x})$.
- Common choices for f are the logistic or logit function $\sigma(a) = \frac{1}{1+e^{-a}}$ and the probit function $\psi(a) = \int_{-\infty}^{a} \mathcal{N}(\theta|0,1)d\theta$.
- We will focus on logistic regression.
- The non-linear embedding is an important step



LOGISTIC REGRESSION

- We assume $p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(\mathbf{w}^T\phi)$ where $\phi = \phi(\mathbf{x})$ and $\phi_i = \phi(\mathbf{x_i})$.
- As $y = y(\phi(\mathbf{x})) \in [0, 1]$ we interpret is as the probability of assigning input $\mathbf{x}$ to class 1, so that the likelihood is

$$p(\mathbf{t}|\mathbf{w}) = \prod_{i=1}^{N} y_i^{t_i}(1 - y_i)^{1-t_i}$$

  where $y_i = \sigma(\mathbf{w}^T\phi_i)$.
- We need to minimise minus the log-likelihood, i.e.

$$E(\mathbf{w}) = -\log p(\mathbf{t}|\mathbf{w}) = -\sum_{i=1}^{N} t_i \log y_i + (1 - t_i)\log(1 - y_i)$$

NUMERICAL OPTIMISATION

- The gradient of $E(\mathbf{w})$ is $\nabla E(\mathbf{w}) = \sum_{i=1}^{N}(y_i - t_i)\phi_i$. The equation $\nabla E(\mathbf{w}) = 0$ has no closed form solution, so we need to solve it numerically.
- One possibility is gradient descend. We initialise $\mathbf{w}^0$ to any value and then update it by

$$\mathbf{w}^{n+1} = \mathbf{w}^n - \eta \nabla E(\mathbf{w}^n)$$

  where the method converges for η small.
- We can also use stochastic gradient descent for online training, using the update rule for $\mathbf{w}$:

$$\mathbf{w}^{n+1} = \mathbf{w}^n - \eta \nabla_{n+1} E(\mathbf{w}^n),$$

  with $\nabla_n E(\mathbf{w}) = (y_n - t_n)\phi_n$

Logistic regression: overfitting

- If we allocate each point $\mathbf{x}$ to the class with highest probability, i.e. maximising $\sigma(\mathbf{w}^T\phi(\mathbf{x}))$, then the separating surface is an hyperplane in the feature space and is given by the equation $\mathbf{w}^T\phi(\mathbf{x}) = 0$.
- If the data is linearly separable in the feature space, then any separable hyperplane is a solution, and the magnitude of $\mathbf{w}$ tends to go to infinity during optimisation. In this case, the logistic function converges to the Heaviside function.
- To avoid this issue, we can add a regularisation term to $E(\mathbf{w})$, thus minimising $E(\mathbf{w}) + \alpha\mathbf{w}^T\mathbf{w}$.

NEWTON-RAPSON METHOD

- As an alternative optimisation, we can use the Newton-Rapson method, which has better convergence properties.
- The update rule reads:

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta \mathbf{H}^{-1} \nabla E(\mathbf{w}^{old})$$

  where $\mathbf{H}$ is the Hessian of $E(\mathbf{w})$, and η the learning rate.
- For logistic regression, we have $\nabla E(\mathbf{w}) = \Phi^T(\mathbf{y} - \mathbf{t})$ and $\mathbf{H} = \Phi^T \mathbf{R} \Phi$, with R diagonal matrix with elements $R_{nn} = y_n(1 - y_n)$.
- It is easy to check that the Hessian is positive definite, hence the function $E(\mathbf{w})$ is convex and has a unique minimum.

MULTI-CLASS LOGISTIC REGRESSION

- We can model directly the multiclass conditional probability, using the soft-max function:

$$p(C_k|\mathbf{x}) = y_k(\mathbf{x}) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

with $a_k = \mathbf{w_k}\phi(\mathbf{x})$. It holds $\frac{\partial y_k(\mathbf{x})}{\partial a_j} = y_k(\delta_{kj} - y_j)$

- Using the boolean encoding of the outputs, the likelihood is

$$p(\mathbf{T}|\mathbf{w_1}, \ldots, \mathbf{w_K}) = \prod_{n=1}^{N}\prod_{k=1}^{K} p(C_k|\phi_n)^{t_{nk}} = \prod_{n=1}^{N}\prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

- Hence we need to minimise

$$E(\mathbf{w_1}, \ldots, \mathbf{w_K}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \log y_{nk}$$

MULTI-CLASS LOGISTIC REGRESSION

- $E(\mathbf{w_1}, \ldots, \mathbf{w_K})$ has gradient

$$\nabla_{\mathbf{w_j}} E(\mathbf{w_1}, \ldots, \mathbf{w_K}) = \sum_{n=1}^{N} (y_{nj} - t_{nj}) \phi_n$$

- and Hessian with blocks given by

$$\nabla_{\mathbf{w_k}} \nabla_{\mathbf{w_j}} E(\mathbf{w_1}, \ldots, \mathbf{w_K}) = - \sum_{n=1}^{N} y_{nk} (I_{kj} - y_{nj}) \phi_n \phi_n^T$$

- Also in this case the Hessian is positive definite, and we can use the Newton-Rapson algorithm for optimisation

Outline

LAPLACE APPROXIMATION - 1 DIMENSION

- It is a general technique to locally approximate a general distribution around a mode with a Gaussian.
- Consider a 1d distribution $p(z) = \frac{1}{Z} f(z)$ where $Z = \int f(z) dz$ is the normalisation constant.
- Pick a mode z_0 of $f(z)$, i.e. a point such that $\frac{d}{dz} f(z_0) = 0$.
- As the logarithm of the Gaussian density is quadratic, we consider a Taylor expansion of $\log f(z)$ around z_0:

$$\log f(z) \approx \log f(z_0) - \frac{1}{2} A (z - z_0)^2$$

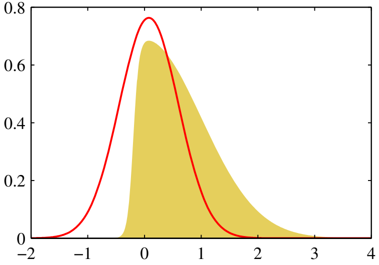
with $A = -\frac{d^2}{dz^2} \log f(z_0)$

LAPLACE APPROXIMATION - 1 DIMENSION

- Hence we have $f(z) \approx f(z_0)\exp(-\frac{1}{2}A(z-z_0)^2)$. Now, we seek the best Gaussian $q(z)$ approximating $p(z)$ around the model z_0, requiring $A > 0$. This is clearly given by

$$q(z) = \left(\frac{A}{2\pi}\right)^{\frac{1}{2}} \exp(-\frac{1}{2}A(z-z_0)^2)$$

- We also have that $Z \approx f(z_0)\left(\frac{A}{2\pi}\right)^{-\frac{1}{2}}$



LAPLACE APPROXIMATION - N DIMENSION

- In n dimensions, we proceed in the same way. Given a density $p(\mathbf{z}) = \frac{1}{Z} f(\mathbf{z})$, we find a mode $\mathbf{z_0}$ (so that $\nabla \log f(\mathbf{z_0}) = \mathbf{0}$, and approximate $\log f(\mathbf{z})$ around $\mathbf{z_0}$ by Taylor expansion, obtaining

$$\log f(\mathbf{z}) = \log f(\mathbf{z_0}) - \frac{1}{2}(\mathbf{z} - \mathbf{z_0})^T \mathbf{A}(\mathbf{z} - \mathbf{z_0})$$

  where $\mathbf{A} = -\nabla\nabla \log f(\mathbf{z_0})$.
- This gives a Gaussian approximation around $\mathbf{z_0}$ by

$$q(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{z_0}, \mathbf{A}^{-1})$$

- Furthermore $Z \approx \frac{(2\pi)^{n/2}}{|\mathbf{A}|^{1/2}} f(\mathbf{z_0})$

MODEL COMPARISON

- We can use Laplace approximation for the marginal likelihood in a model comparison framework.
- Consider data $\mathcal{D}$ and a model $\mathcal{M}$ depending on parameters θ. We fix a prior $\mathcal{P}(\theta)$ over θ and compute the posterior by Bayes theorem:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$$

- Here $p(\mathcal{D}) = \int p(\mathcal{D}|\theta)p(\theta)d\theta$ is the marginal likelihood. It fits in the previous framework by setting $Z = p(\mathcal{D})$, and $f = p(\mathcal{D}|\theta)p(\theta)$.

BIC

- By Laplace approximation around the maximum a-posteriori estimate θ_{MAP}:

  $$\log p(\mathcal{D}) \approx \log p(\mathcal{D}|\theta_{MAP}) + \log p(\theta_{MAP}) + \frac{M}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{A}|$$

  where $\mathbf{A} = -\nabla\nabla p(\mathcal{D}|\theta_{MAP})p(\theta_{MAP})$. The last three terms in the sum penalise the log likelihood in terms of model complexity.

- A crude approximation of them is

  $$logp(\mathcal{D}) \approx \log p(\mathcal{D}|\theta_{MAP}) - \frac{1}{2}M\log N$$

  which is known as Bayesian Information Content, and can be used to penalise log likelihood w.r.t. model complexity, to compare different models.