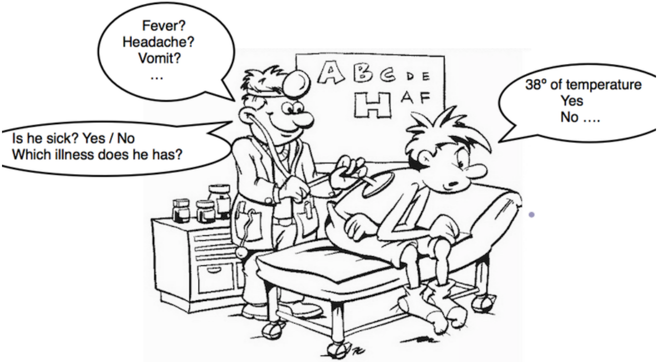
Machine learning tools: learning to classify

Vanessa Gómez Verdejo

http://vanessa.webs.tsc.uc3m.es/blog/

Summary

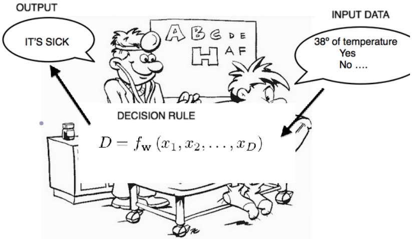
The classification problem



Machines can learn to classify

Machine learning...

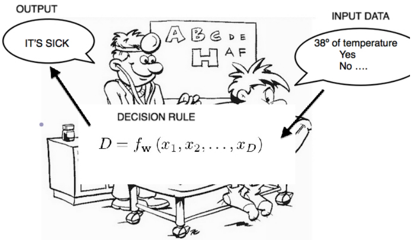
designs algorithms that allow computers to learn tasks based on empirical data



Machines can learn to classify

Machine learning...

designs algorithms that allow computers to learn tasks based on empirical data



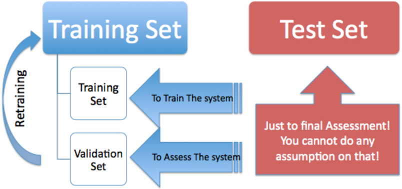
Final goal

Then machine needs to obtain good results in new patterns; this is known as **GENERALIZATION** (**OVERFITTING** problems has to be avoided).

Ingredients of the classification problem

The data

- Training data: input samples $\left\{\mathbf{x}^{(l)}\right\}_{l=1}^{L} \in \Re^{M}$ and their labels $\left\{y^{(l)}\right\}_{l=1}^{L}$ which belong to the set of categories $\{C_1, C_2, ...C_J\}$.
- Validation data: useful to adjust free parameters.
- Test data: to evaluate the final performance of the classifier.



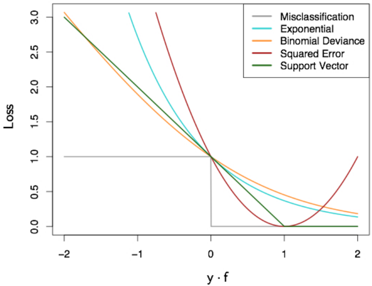
Ingredients of the classification problem

The loss function

Parametric models fix a decision function:

$$D = f_{\mathbf{w}}(\mathbf{x})$$

and need to adjust their free parameters ($\mathbf{w}$) minimizing a loss function. In the ideal case, this cost would be the **classification rate**; however, this cost function is not differentiable, so upper-bounds are preferred.



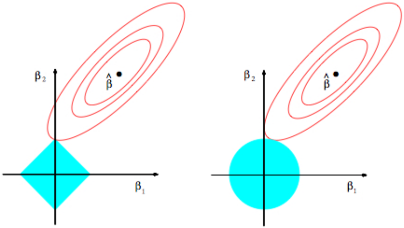
- Exponential: $\exp\left(-yf(\mathbf{x})\right)$
- Binomial Deviance: $\log\left[1 + \exp\left(-yf(\mathbf{x})\right)\right]$
- Squared error: $\left(y - f(\mathbf{x})\right)^2 = \left(1 - yf(\mathbf{x})\right)^2$
- SVM Hinge loss: $\left[1 - yf(\mathbf{x})\right]_+$

Ingredients of the classification problem

The regularization term

It's typical to include a regularization term during the optimization of the cost function to add some properties to the solution:

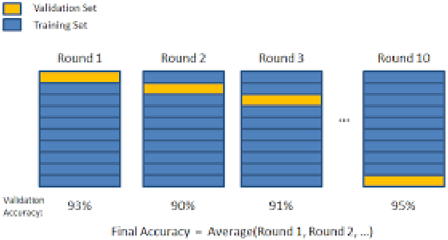
- An L2 term helps to avoid overfitting problems (it maximizes the classifier margin).
- An L1 term provides sparsity to the solution.



Ingredients of the classification problem

Hiperparameter selection

- The training process depends on hyperparameter values.
- These values are critical in the generalization capability.
- **Cross validation** (CV): divides the data set in K subsets without replacement and successively train K models with all the subsets but one which is used to validate.



Criteria to classify classifiers

- Binary vs. multiclass (and multilabel)
- Linear vs. non-linear
- Parametric vs. non-parametric
- Discriminative vs. generative
- Single vs. ensembles

Strategies in multiclass problems

One vs. one

- It trains a binary classifier per pair of classes.
- At prediction time, the class which receives the most votes is selected.
- It requires to train $n_{classes} * (n_{classes} - 1)/2$ classifiers.
- Each individual learning problem involves a small subset of the data.

One vs. all

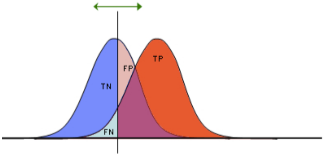
- It defines binary problems fitting each class against the remaining classes.
- Only $n_{classes}$ classifiers are trained (computational efficiency).
- A gain of interpretability (each class is represented by one classifier).
- This is the most commonly used strategy.

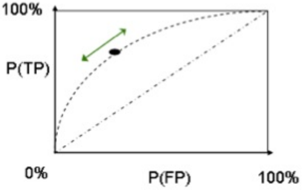
Performance evaluation in binary problems

- Classification error or accuracy
- False positive (FP) rate, True positive (TP) or detection rate, ...

	$D = 1$	$D = 0$
$H = 1$	TP	FP
$H = 0$	TN	FN

- ROC curve (AUC)





Performance evaluation in multiclass problems

- Classification error or accuracy
- Confusion matrix: It analyzes the number of errors by class

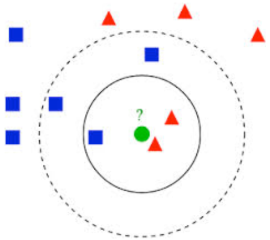
		ACTUAL			Prediction Totals	Prediction Error%
		Setosa	Versicolour	Virginica		
PRED	Setosa	50	0	0	50	0.00%
PRED	Versicolour	0	48	1	49	2.04%
PRED	Virginica	0	2	49	51	3.92%
Actual Totals		50	50	50	150	2.00%
Actual Error%		0.00%	4.00%	2.00%	2.00%	

K-Nearest Neighbours

It is a **non-parametric** classifier, i.e, there are not parameters to be learned.

To classify a test data $\mathbf{x}^*$:

- Select the value of K.
- Search, among the training data, the K nearest neighbours of $\mathbf{x}^*$.
- Decide that $\mathbf{x}^*$ belongs to the majority class of the neighbours.



It's intrinsically a **multiclass** classifier.

The support vector machine

- It is the reference (baseline) **binary** classifier.
- It is characterized for **maximizing the classification margin**.
- It minimizes the *hinge-loss*.
- Its formulation can be reduced to a convex optimization problem (**unique solution**).
- Its linear formulation can provide **non linear** classifiers by means of the *kernel trick*.
- There are multiple extensions: for regression and novelty-detection, with different cost functions, different regularizations....

The linear SVM: separable case

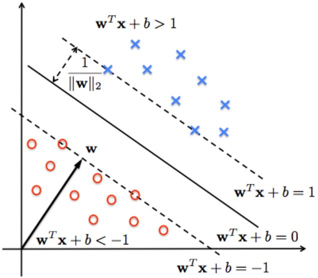
Let's start considering that training data are linearly separable...

- We want a **maximum margin** classifier

$$\rho = \frac{1}{\|\mathbf{w}\|_2}$$

- which is able to classify all training data:

$$\min_{\mathbf{w},b} \quad \|\mathbf{w}\|_2^2$$
$$\text{st.}y^{(l)}\left(\mathbf{w}^T\mathbf{x}^{(l)}+b\right) \geq 1; \quad \forall l$$



The linear SVM: separable case

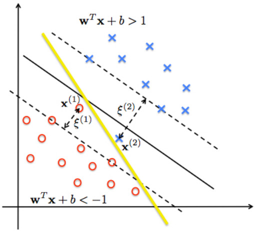
When training data aren't linearly separable or we let some data be misclassified...

- We can add some slack variables in the formulation
- Linear binary classifier

$$\min_{\mathbf{w},b,\xi_{(l)}} \quad \|\mathbf{w}\|_\mathbf{2}^\mathbf{2} + \mathbf{C}\sum_{\mathbf{l=1}}^{\mathbf{L}} \xi^{(\mathbf{l})}$$

$$\text{st.} \quad y^{(l)}\left(\mathbf{w}^T\mathbf{x}^{(l)} + b\right) \geq 1 - \xi^{(l)}; \quad \forall l$$

$$\xi^{(l)} \geq 0; \quad \forall l$$



The linear support vector machine

- The training of the linear SVM relies on solving

$$\min_{\mathbf{w},b,\xi_{(l)}} \quad \|\mathbf{w}\|_2^2 + \mathbf{C}\sum_{l=1}^{L} \xi^{(l)}$$
$$\text{st.} \quad y^{(l)}\left(\mathbf{w}^T\mathbf{x}^{(l)} + b\right) \geq 1 - \xi^{(l)}; \quad \forall l$$
$$\xi^{(l)} \geq 0; \quad \forall l$$

  an optimization problem with a **unique solution**.
- The value of C has to be properly selected.
- The soft-output of the classifier is given by

$$f(\mathbf{x}^*) = \mathbf{w}^T\mathbf{x}^* + b,$$

  if $f(\mathbf{x}^*) > 0$ (< 0) the datum is assigned to class $+1$ (-1).
- This optimization problem can be reformulated by means of the Lagrangian multipliers, providing a dual formulation... (next).

The SVM dual formulation

- Previous SVM optimization problem can be reformulated by means of the Lagrangian multipliers, providing a dual formulation

$$\max_{\alpha^{(1)},\dots,\alpha^{(L)}} \quad \sum_{l=1}^{L} \alpha^{(l)} - \frac{1}{2} \sum_{l=1}^{L} \sum_{l'=1}^{L} y^{(l)} y^{(l')} \alpha^{(l)} \alpha^{(l')} \mathbf{x}^{(l)T} \mathbf{x}^{(l')}$$

$$\text{st.} \quad 0 < \alpha^{(l)} < C \quad \forall l$$

$$\sum_{l=1}^{L} \alpha^{(l)} y_{(l)} = 0; \quad \forall l$$

where now the optimization has to be solved regarding to the dual variables $\alpha^{(l)}$.

- The problem complexity is given by the number of data (L).

The SVM dual formulation

- Once $\left\{\alpha^{(l)}\right\}_{l=1}^{L}$ values are obtain, one can compute the weight vector as:

$$\mathbf{w}^T = \sum_{l=1}^{L} y^{(l)} \alpha^{(l)} \mathbf{x}^{(l)}$$

- Then, the soft-output of the classifier for a new data $\mathbf{x}^*$ is given by

$$f(\mathbf{x}^*) = \mathbf{w}^T \mathbf{x}^* + b = \sum_{l=1}^{L} y^{(l)} \alpha^{(l)} \mathbf{x}^{(l)} \mathbf{x}^* + b$$

- Sparsity of the SVM solution:
    - Most dual variables are zero;
    - The SVM output is given by a linear combination of just few input data
    - These data, which support the solution of the classifier, are call **support vectors**.

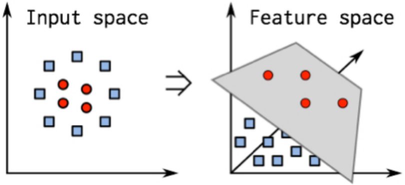
No-linear SVM

Mapping the data

- When we cannot find linear solutions in the input space
- We can map the data to a high dimensional space (even of infinitive dimension)

$$\mathbf{x} \longrightarrow \phi(\mathbf{x})$$

- then, a liner solution of the problem can be found in this high dimensional space



No-linear SVM

Kernel trick

- In most cases, you cannot compute the kernel transformation explicitly.
- But you can compute the dot product of the data in the feature space.

$$K\left(\mathbf{x}, \mathbf{x}'\right) = \phi\left(\mathbf{x}\right)^T \phi\left(\mathbf{x}'\right)$$

- $K\left(\cdot, \cdot\right)$, which is called *kernel function*, measures similarities between the data.

Some examples

- Linear kernel: $K\left(\mathbf{x}, \mathbf{x}'\right) = \mathbf{x}^T \mathbf{x}'$
- Polynomial kernel: $K\left(\mathbf{x}, \mathbf{x}'\right) = \left(\mathbf{x}^T \mathbf{x}' + c\right)^d$
- Gaussian kernel: $K\left(\mathbf{x}, \mathbf{x}'\right) = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2\right)$

No-linear SVM

- We can obtain no-linear classification boundaries by replacing the dot products of the dual formulation by a kernel function.

$$\max_{\alpha^{(1)},\dots,\alpha^{(L)}} \quad \sum_{l=1}^{L} \alpha^{(l)} - \frac{1}{2} \sum_{l=1}^{L} \sum_{l'=1}^{L} y^{(l)} y^{(l')} \alpha^{(l)} \alpha^{(l')} K\left(\mathbf{x}^{(l)}, \mathbf{x}^{(l')}\right)$$

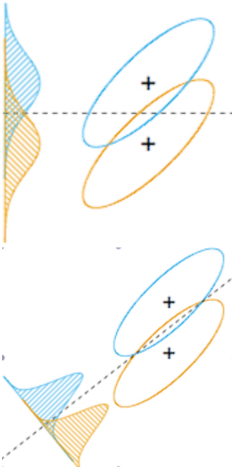
$$\begin{aligned} \text{st.} \quad & 0 < \alpha^{(l)} < C \quad \forall l \\ & \sum_{l=1}^{L} \alpha^{(l)} y_{(l)} = 0; \quad \forall l \end{aligned}$$

- Now, the SVM weight cannot be explicitly computed.
- But, the SVM output can be obtained as:

$$f(\mathbf{x}^*) = \mathbf{w}^T \mathbf{x}^* + b = \sum_{l=1}^{L} y^{(l)} \alpha^{(l)} K\left(\mathbf{x}^{(l)}, \mathbf{x}^*\right) + b$$

Linear Discriminant Analysis

- It is a **generative** classification model.
- It considers that the data follow a gaussian distribution with the same covariance matrix.
- Then, the optimum classifier is **linear**.
- It finds the direction of minimum overlap among the classes. We can project the data over this direction and classify them in this new space.
- Fisher discriminant generalizes it to any data set.



Linear Discriminant Analysis

- Let's consider that the data follow a gaussian distribution with the same covariance matrix.

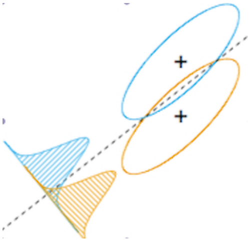
$$p(\mathbf{x}|y=-1) \sim G(\mathbf{m}_0, V) \qquad p(\mathbf{x}|y=1) \sim G(\mathbf{m}_1, V)$$

- Then, the optimum classifier is

$$\hat{y} = \text{sign}\left(\mathbf{w}^T\mathbf{x}\right)$$

where $\mathbf{w} = V^{-1}(\mathbf{m}_1 - \mathbf{m}_0)$.

- To decide, we project the input data over $\mathbf{w}$, i.e., for each multidimensional input data, a unidimensional value is obtained and a threshold is applied.
- In multiclass problems there are as many linear discrimination functions as number of classes minus one.



Logistic Regression

- Define the posterior probabilities (for the binary case) as:

$$P(Y = 1|\mathbf{x}) = \frac{\exp(\mathbf{w}^T\mathbf{x})}{1 + \exp(\mathbf{w}^T\mathbf{x})} = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}$$

$$P(Y = 0|\mathbf{x}) = 1 - P(Y = 1|\mathbf{x}) = \frac{1}{1 + \exp(\mathbf{w}^T\mathbf{x})}$$

- In this way, the MAP classifier is linear:

$$\log \frac{P(Y = 1|\mathbf{x})}{P(Y = 0|\mathbf{x})} = \mathbf{w}^T\mathbf{x}$$

- Taking into account that the probability that Y is 1 is given by $P(Y = 1|\mathbf{x}) = p(\mathbf{x})$ and $1 - p(\mathbf{x})$ is the probability of being 0, the joint likelihood over all training data is:

$$L(\mathbf{w}) = \prod_{l=1}^{L} p(\mathbf{x}_l)^{y_l} \left(1 - p(\mathbf{x}_l)\right)^{y_l}$$

Logistic Regression

- Then, to learn the parameters of the model, we can maximize the log-likelihood for N observations:

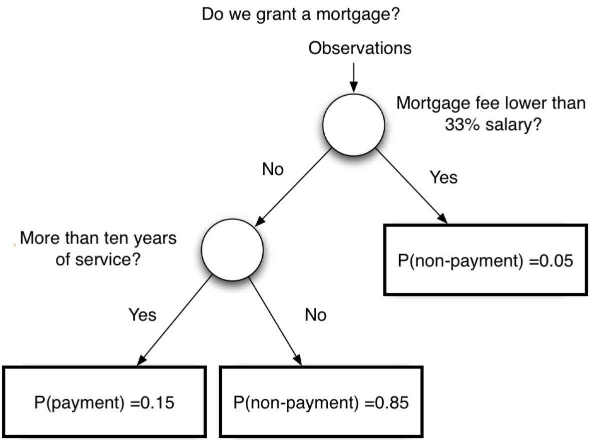
$$l(\mathbf{w}) = \sum_{l=1}^{L} \{y_l \log\left(p(\mathbf{x}_l)\right) + (1 - y_l)\log\left(1 - p(\mathbf{x}_l)\right)\}$$

$$l(\mathbf{w}) = \sum_{l=1}^{L} \left\{y_l(\mathbf{w}^T\mathbf{x}) - \log\left(1 + \exp(\mathbf{w}^T\mathbf{x}_l)\right)\right\}$$

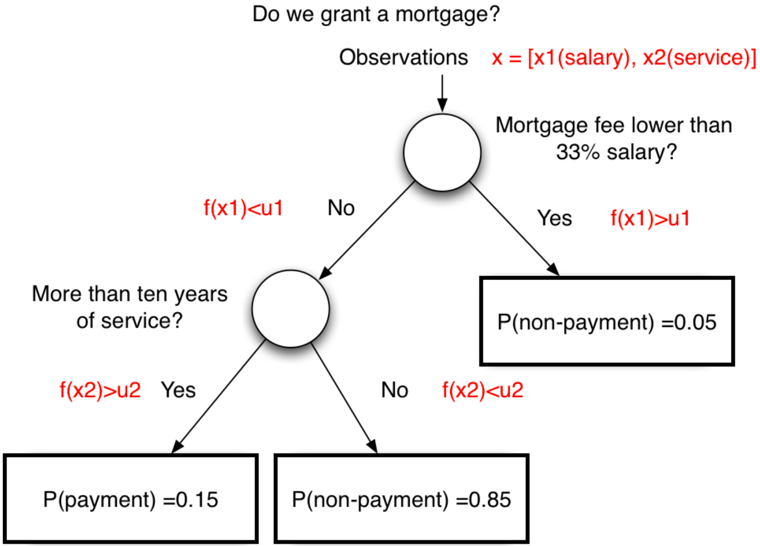
- A Newton method is used to find the maximum of this loss function.
- **Regularized Logistic Regression**: we can minimize the equivalent loss function with a regularization term

$$\min_{\mathbf{w}} - \sum_{l=1}^{L} \left\{y_l(\mathbf{w}^T\mathbf{x}) - \log\left(1 + \exp(\mathbf{w}^T\mathbf{x}_l)\right)\right\} + C\|\mathbf{w}\|_2^2$$

Decision trees: working principles



Decision trees: working principles



Training a decision tree

- Inputs: $\left\{\mathbf{x}^{(l)}\right\}_{l=1}^{L} \in \Re^M$ and $\left\{y^{(l)}\right\}_{l=1}^{L} \in \{C_1, C_2, ...C_J\}$.
- for $d = 1 : DM$ # *For each feature*
    - for $u_{d,l} \in$ all values of x_d # *Exploring thresholds*
        - Evaluate the index Gini :

$$g(u_{d,l}) = \sum_{j=1}^{J} P_j(u_{d,l})\,(1 - P_j(u_{d,l}))$$

being $P_j(u_{d,l})$ the fraction of items classified in the class C_j by the threshold $u_{d,l}$

- Select threshold $(u_{d,l})$ and feature (x_d) minimizing $g(u_{d,l})$
- Split the data according to x_d and threshold $u_{d,l}$
- Apply recursively

The minimization of Gini index aims at getting leaves “pure enough”.

Random Forest

Working principles

- Build many trees (forest) randomizing samples and features
- Key points:
    - Low correlation among trees
    - Strength of each tree

Test data classification

- Then, each test data ($\mathbf{x}^*$) is classified by all the tree
- The forest classification rule is given by:

$$C_j^* = \underset{j}{\operatorname{argmax}} \frac{1}{T} \sum_{t=1}^{T} P_t(C_j|\mathbf{x}^*)$$

where $P_t(C_j|\mathbf{x}^*)$ is the probability output of each tree.

Training a Random Forest

- Inputs: $\left\{\mathbf{x}^{(l)}\right\}_{l=1}^{L} \in \Re^M$ and $\left\{y^{(l)}\right\}_{l=1}^{L} \in \{C_1, C_2, ...C_J\}$.
- For each tree $(1, \ldots, T)$:
  - Sample with replacement from the original data set (boostrap sampling) : $L?(< L)$ data
  - Randomly select $D'(< D)$ features
  - Train a tree optimizing the index Gini with the data matrix $(L? \times D')$.
  - Once the forest is trained, each leaf of the tree has the class probabilities: $P_t(C_j|\mathbf{x})$

Introduction to ensembles

Goal

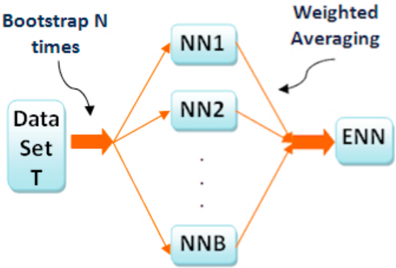
- Combine a set of weak learners to build a strong one
- Exploit the diversity among the base learners

Kind of ensembles

- Bagging, boosting, mixture of experts...
- Random forests is set of bagged trees

Bagging: Boostrap Aggregating

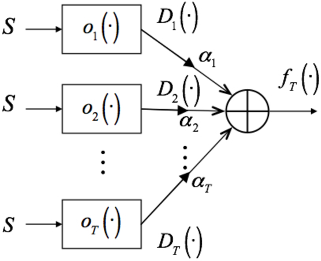
- Generate T data subsets by subsampling the training data with replacement.
- Train T models, one model for each training subset
- Classification: obtain T outputs and majority vote



Boosting

Idea

Iteratively pay more attention to the misclassified data (-¿ emphasis function).



- Emphasis function:

$$D_{t+1}(\mathbf{x}^{(l)}) = \frac{D_t(\mathbf{x}^{(l)}) \exp\left(-\alpha_t o_t(\mathbf{x}^{(l)}) y^{(l)}\right)}{Z_t}$$

- Output weights (α_t) can be analytically computed
- Final output:

$$f(\mathbf{x}^*) = \sum_{t=1}^{T} \alpha_t o_t(\mathbf{x}^*)$$