Inteligencia Artificial

UTN – FRVM

5º Año Ing. en Sistemas de Información

Agenda

- Resampling Methods.
- K-Nearest Neighbours.
- Logistic Regression
- Support Vector Machines
  - *maximal margin classifier*
  - *support vector classifier*
  - *support vector machines*

Resampling Methods

- ***Resampling methods*** involve repeatedly drawing samples from a training set and refitting a model of interest on each sample in order to obtain additional information about the fitted model.

- In order to estimate the variability of a fit (e.g. regression, classification), we can repeatedly draw different samples from the training data, train a model regards to each new sample, and then examine the extent to which the resulting fits differ.

- Such an approach may allow us to obtain information that would not be available from fitting the model only once using the original training sample.

- Resampling approaches can be computationally expensive, because they involve fitting the same statistical method multiple times using different subsets of the training data.

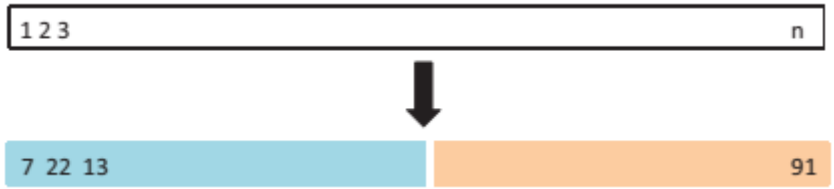
- ***Cross-validation*** and the ***bootstrap***.

- ***Cross-validation*** can be used to estimate the test error associated with a given statistical learning method in order to evaluate its performance, or to select the appropriate level of flexibility.
- The process of evaluating a **model's performance is known as *model assessment***, whereas the process of selecting the proper **level of flexibility for a model is known as *model selection***.
- The ***bootstrap*** is used in several contexts, most commonly to provide a measure of accuracy of a parameter estimate or of a given selection statistical learning method.

Cross-Validation

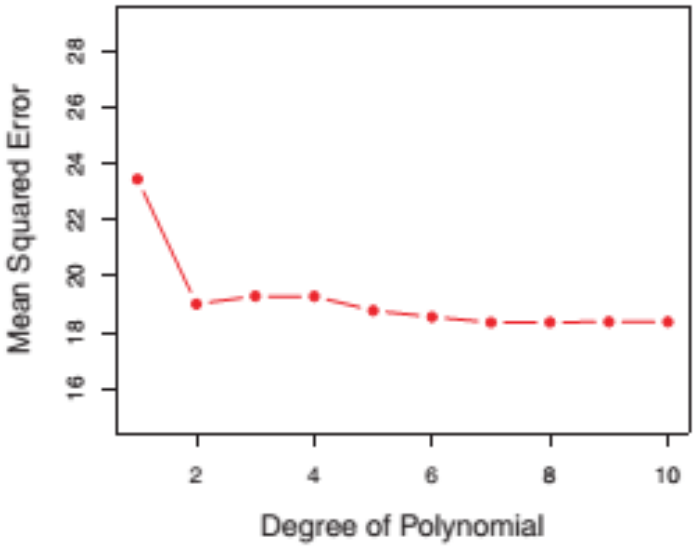
- In the absence of a very large designated test set that can be used to directly estimate the test error rate, a number of techniques can be used to estimate this quantity using the available training data.

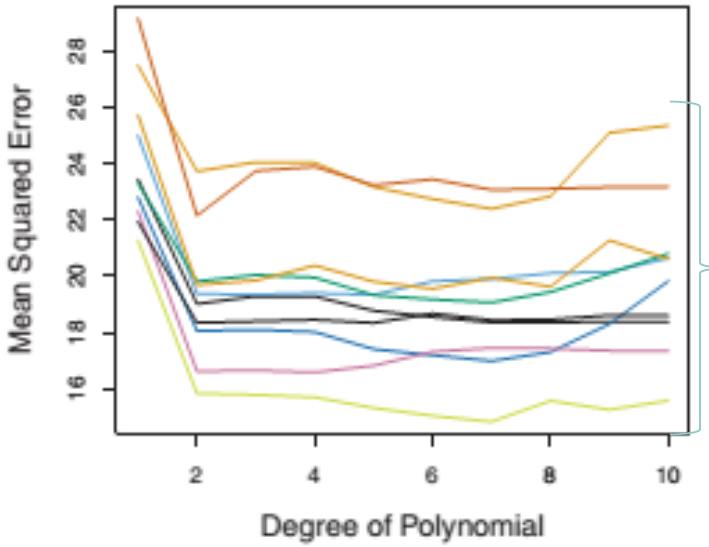
- *The Validation Set Approach*



- *A schematic display of the validation set approach. A set of **n** observations are randomly split into a training set (shown in blue, containing observations 7, 22, and 13, among others) and a validation set (shown in beige, and containing observation 91, among others). The statistical learning method is fit on the training set, and its performance is evaluated on the validation set.*

Validation set approach



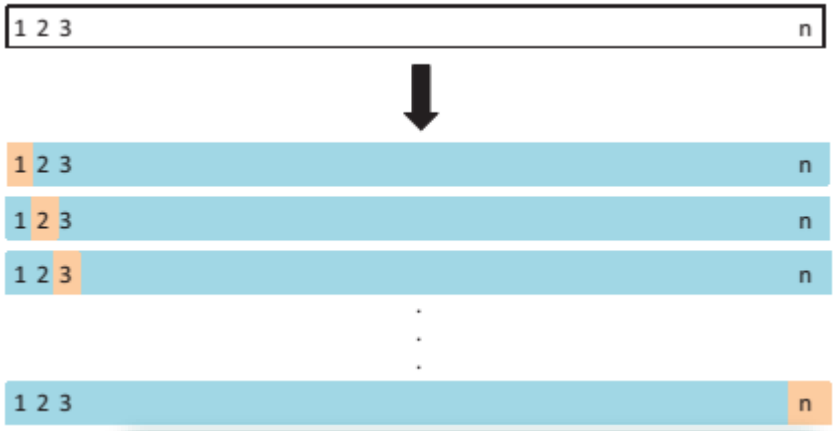


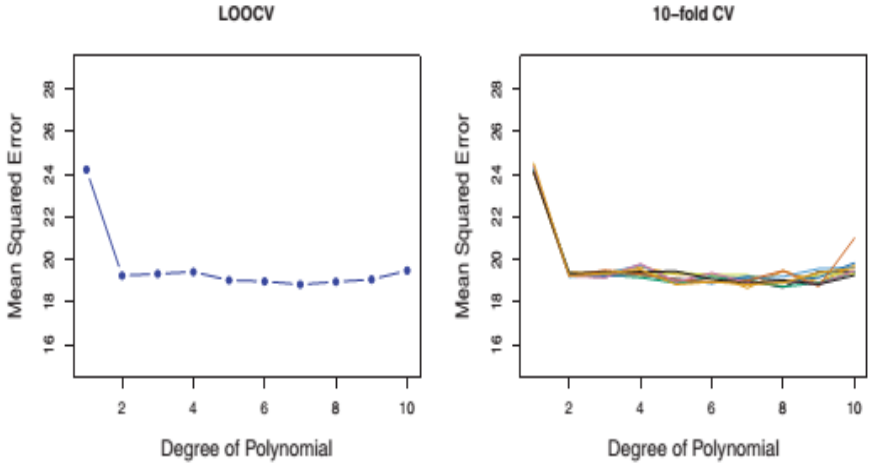
Variability? Overestimation?

- *The validation set approach was used on the* Auto *data set in order to estimate the test error that results from predicting* mpg *using polynomial functions of* horsepower.
- Left: *Validation error estimates for a single split into training and validation data sets.*
- Right: *The validation method was repeated ten times, each time using a different random split of the observations into a training set and a validation set. This illustrates the variability in the estimated test MSE that results from this approach.*

Leave - One - Out Cross - Validation

- Involves splitting the set of observations into two parts.
- However, instead of creating two subsets of comparable size, a single observation (x_1,y_1) is used for the validation set, and the remaining observations $\{(x_2,y_2), \ldots, (x_n,y_n)\}$ make up the training set.

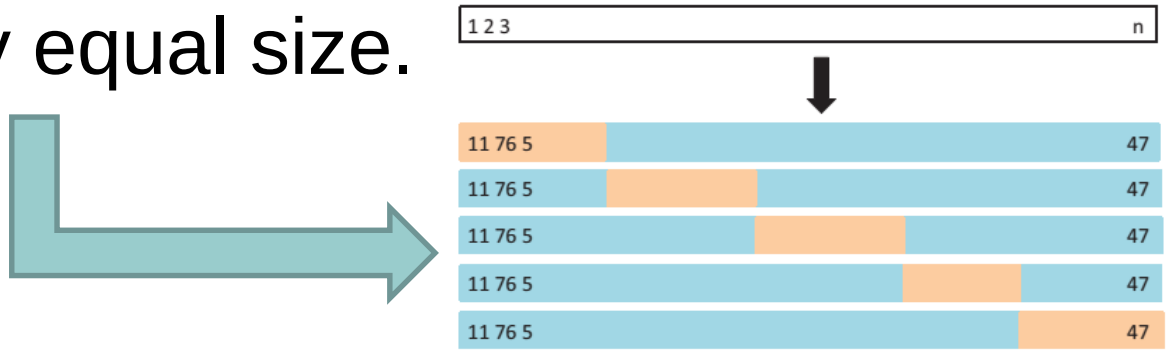




$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\mathrm{MSE}_i.$$

k-Fold Cross-Validation

- This approach involves randomly dividing the set of observations into *k* groups, or *folds*, of approximately equal size.



- The first fold is treated as a validation set, and the method is fit on the remaining $k - 1$ folds.
- The mean squared error, MSE_1, is then computed on the observations in the held-out fold. This procedure is repeated *k* times

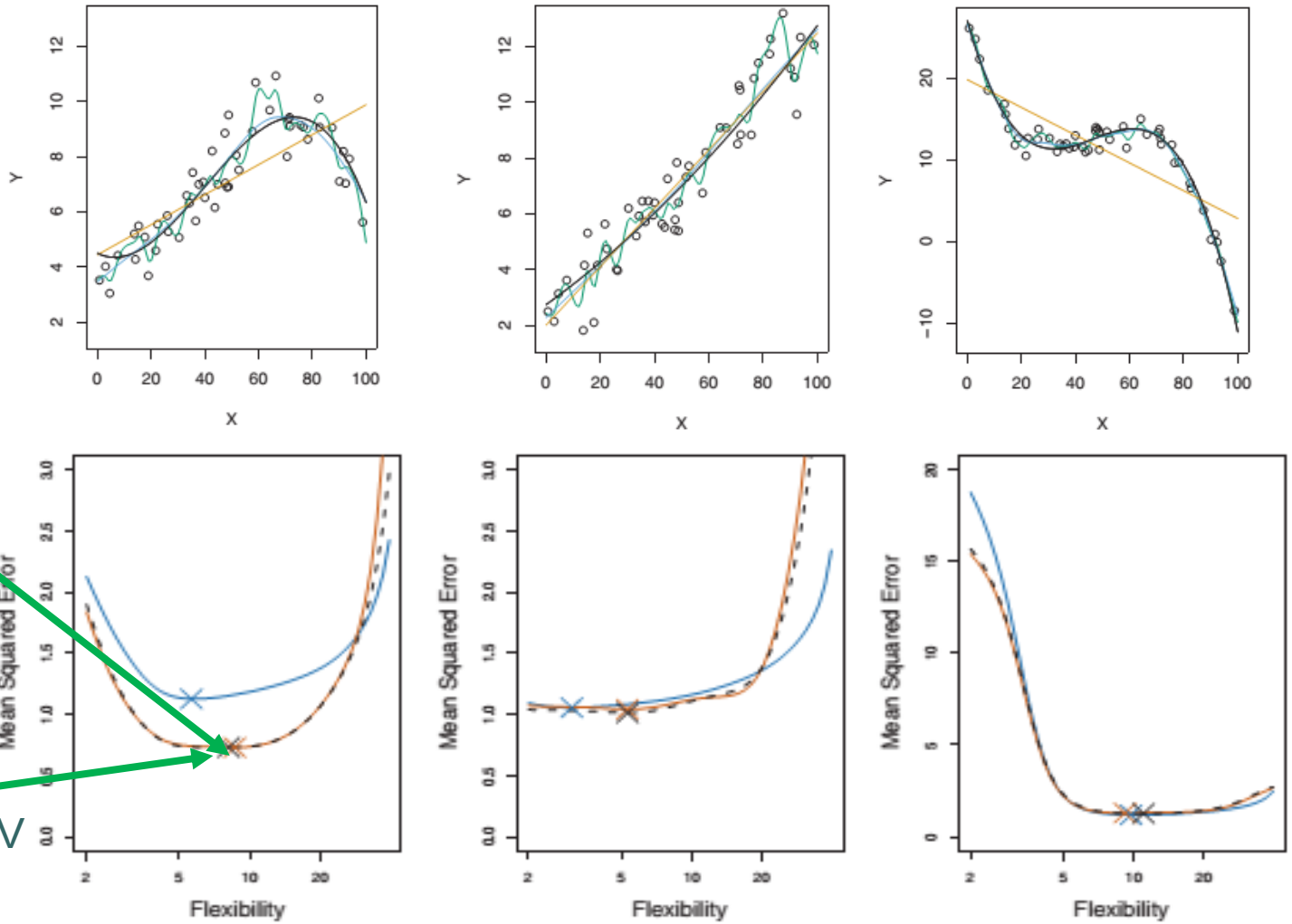
k-Fold Cross-Validation

- This process results in *k* estimates of the test error, MSE_1, MSE_2,..., MSE_k. The *k*-fold CV estimate is computed by averaging these values.

$$\text{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \text{MSE}_i.$$

- Cross-validation is a very general approach that can be applied to almost any statistical learning method.

k-Fold Cross-Validation



An overview of Classification

- **Classification** problems occur often, perhaps even more so than **regression** problems. Some examples include:

- A person arrives at the emergency room with a set of symptoms that could possibly be attributed to one of three medical conditions. Which of the three conditions does the individual have?

- An online banking service must be able to determine whether or not a transaction being performed on the site is fraudulent, on the basis of the user's IP address, past transaction history, and so forth.

- On the basis of DNA sequence data for a number of patients with and without a given disease, a biologist would like to figure out which DNA mutations are deleterious (disease-causing) and which are not.

K-Nearest Neighbours

- Many approaches attempt to estimate the *conditional distribution* of *Y* given *X*, and then classify a given observation to the class with **highest *estimated* probability**.

- Given a positive integer *K* and a test observation x_0, the *KNN* classifier first identifies the neighbors *K* points in the training data that are closest to x_0, represented by N_0

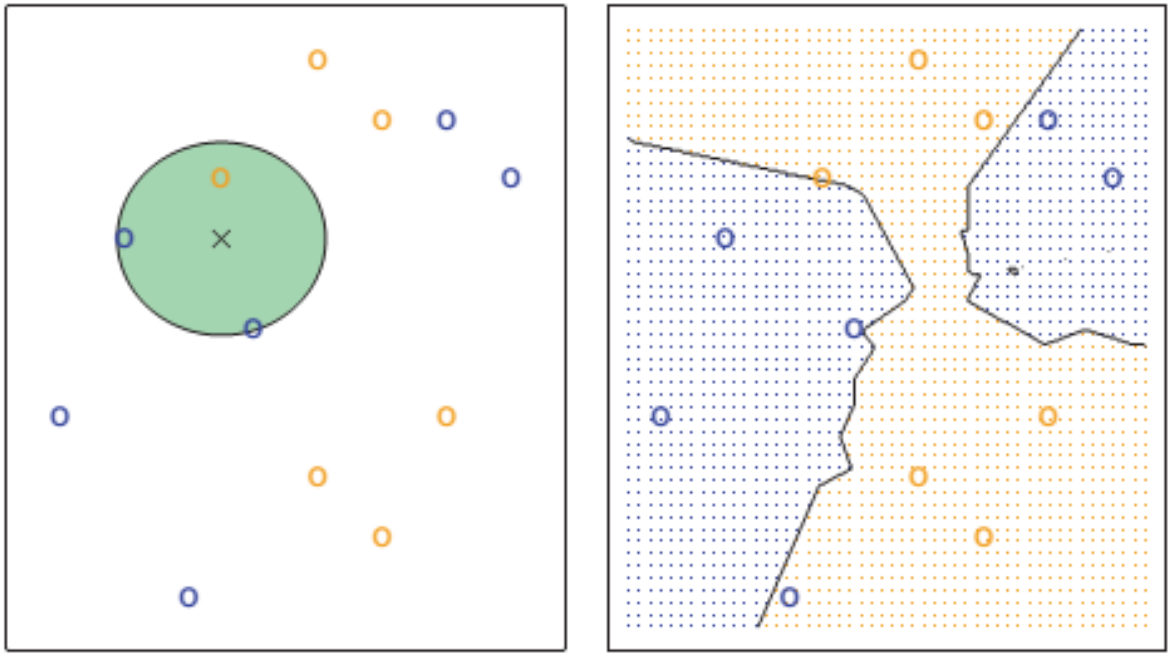
- The conditional probability for class *j* as the fraction of points in N_0 whose response values equal *j:*

$$\Pr(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j)$$

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{p} (x_{ri} - x_{rj})^2}$$

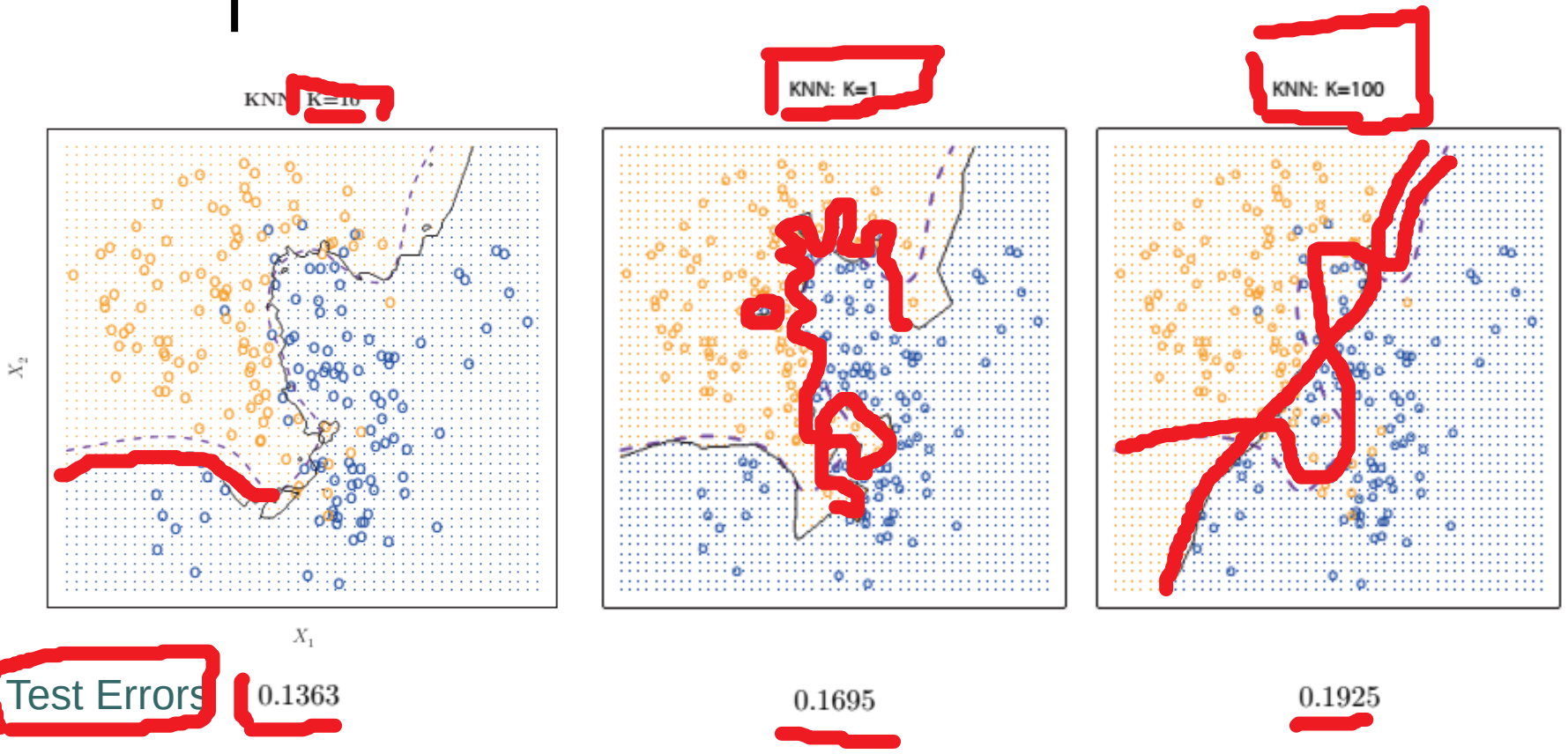
$$d = |\mathbf{x} - \mathbf{y}| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \ .$$

KNN



- *The **KNN** approach, using K = 3, is illustrated in a simple situation with six blue observations and six orange observations.*
- ***Left**: a test observation at which a predicted class label is desired is shown as a black cross. The three closest points to the test observation are identified, and it is predicted that the test observation belongs to the most commonly-occurring class, in this case blue.*
- ***Right**: The KNN decision boundary for this example is shown in black. The blue grid indicates the region in which a test observation will be assigned to the blue class, and the orange grid indicates the region in which it will be assigned to the orange class.*

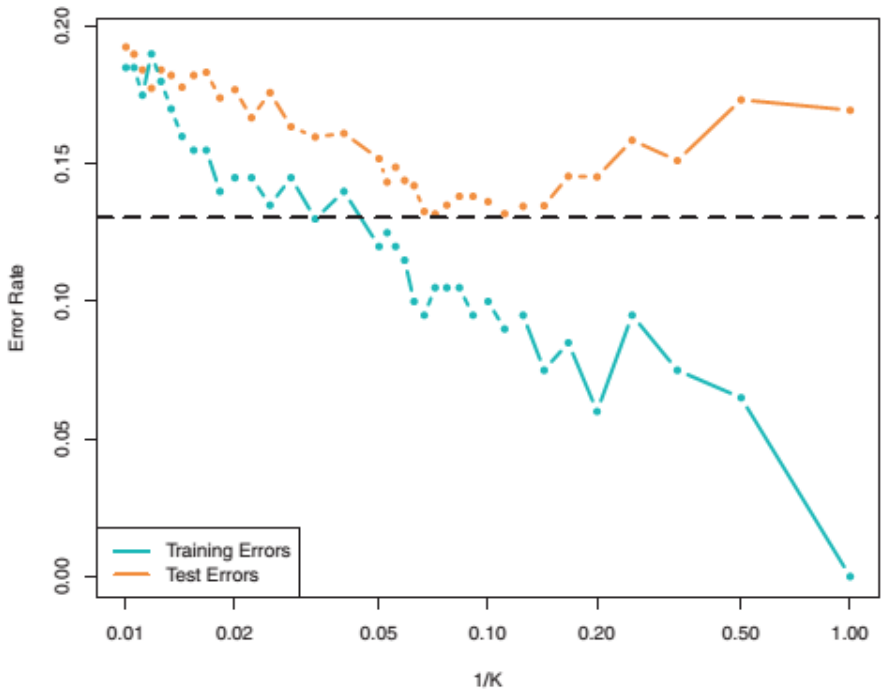
KNN



- *The black curve indicates the KNN decision boundary using K* = 10, 1, 100. *The Bayes decision boundary is shown as a purple dashed line.*

Error: 0.1303

KNN Training-Test Errors

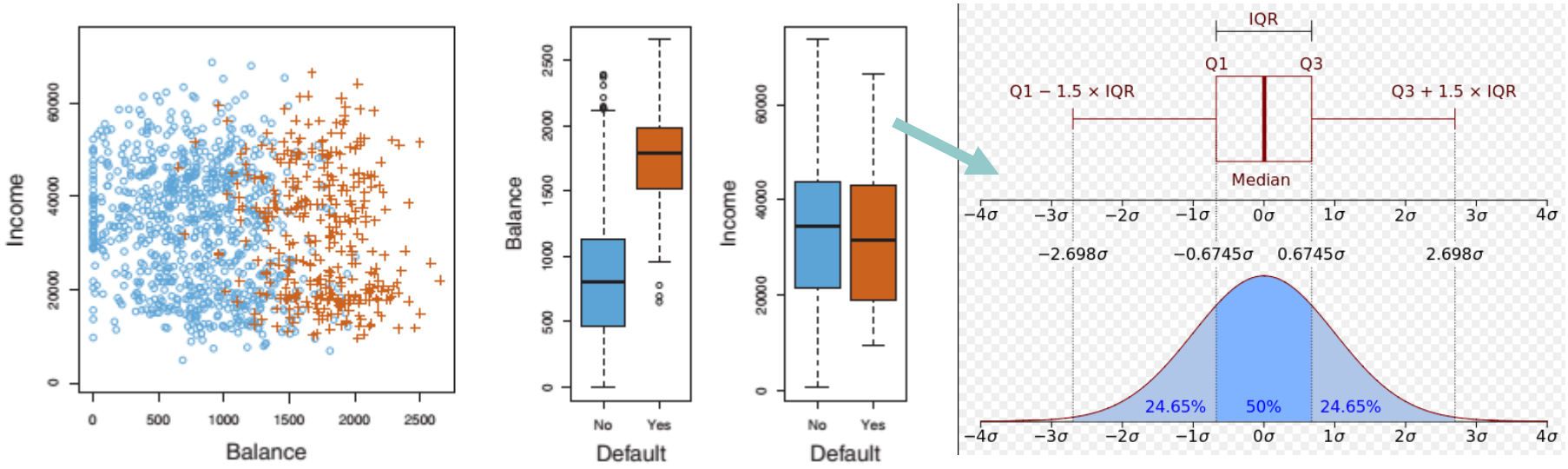


K decreases

- *The **KNN** training error rate (blue, 200 observations) and test error rate (orange, 5,000 observations), as the level of flexibility (assessed using 1/K) increases, or equivalently as the number of neighbors K decreases. The black dashed line indicates the Bayes error rate. The jumpiness of the curves is due to the small size of the training data set.*

Default Dataset

- We are interested in predicting whether an individual will default on his/her credit card payment, on the basis of annual income and monthly credit card balance.

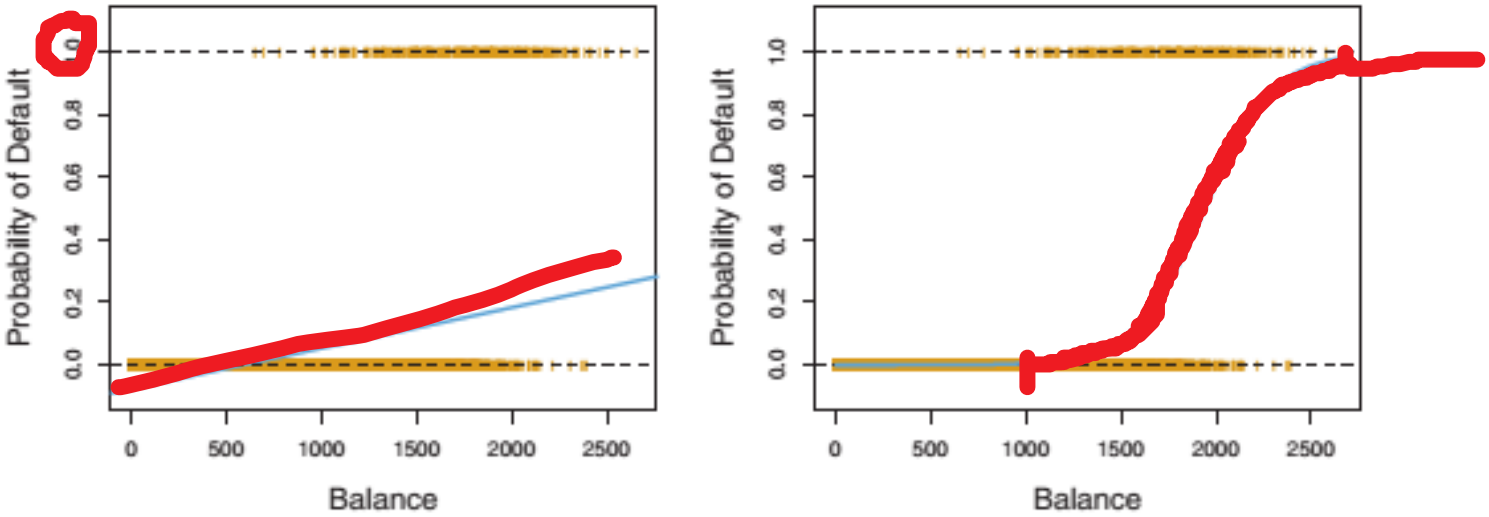


- *The* Default *data set.* Left: *The annual incomes and monthly credit card balances of a number of individuals. The individuals who defaulted on their credit card payments are shown in orange, and those who did not are shown in blue.* Center: *Boxplots of* balance *as a function of* default *status.* Right: *Boxplots of* income *as a function of* default *status.*

Logistic Regression

- Consider again the Default data set, where the response default falls into one of two categories, Yes or No.
- Rather than modeling this response *Y* directly, logistic regression models the *probability* that *Y* belongs to a particular category.

Logistic Regression



- *Classification using the* Default *data.*
- ***Left***: *Estimated probability of* default *using linear regression. The orange ticks indicate the 0/1 values coded for* default(No *or* Yes*).*
- ***Right***: *Predicted probabilities of* default *using logistic regression. All probabilities lie between* 0 *and* 1*.*

Logistic Regression

- For the Default data, logistic regression models the probability of default.
- For example, the probability of default given balance can be written as *Pr(default = Yes|balance), abbreviated as* ***p(balance)***
- For example, one might predict default = Yes for any individual for whom ***p(balance)*** > 0.5
- Alternatively, if a company wishes to be conservative in predicting individuals who are at risk for default, then they may choose to use a lower threshold, such as ***p(balance) > 0.1***

The Logistic Model

- How should we model the relationship between $p(X) = \Pr(Y = 1|X)$ and X? (For convenience we are using the generic 0/1 coding for the response).

$$p(X) = \beta_0 + \beta_1 X.$$

- For balances close to zero we predict a negative probability of default; if we were to predict for very large balances, we would get values bigger than 1.

- These predictions are not sensible, since of course the true probability of default, regardless of credit card balance, must fall between 0 and 1.

Logistic Regression

- Any time a straight line is fit to a binary response that is coded as 0 o r 1 , in principle we can always predict $p(X) < 0$ for some values of X and $p(X) > 1$ for others (unless the range of X is limited).
- To avoid this problem, we must model $p(X)$ using a function that gives outputs between 0 and 1 for all values of X.
- In logistic regression, we use the *logistic function*

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}.$$

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'})).$$

- We seek estimates for β_0 and β_1 such that the predicted probability $\hat{p}(x_i)$ of default for each individual, corresponds as closely as possible to the individual's observed default status.
- We try to find $\hat{\beta}_0$ and $\hat{\beta}_1$ such that plugging these estimates into the model for $p(X)$, yields a number close to one for all individuals who defaulted, and a number close to zero for all individuals who did not.

Making Predictions

- Once the coefficients have been estimated, it is a simple matter to compute the probability of default for any given credit card balance.
- For example, using the coefficient estimates, we predict that the default probability for an individual with a balance of \$1, 000 is

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055\times 1,000}}{1+e^{-10.6513+0.0055\times 1,000}} = 0.00576$$

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}.$$

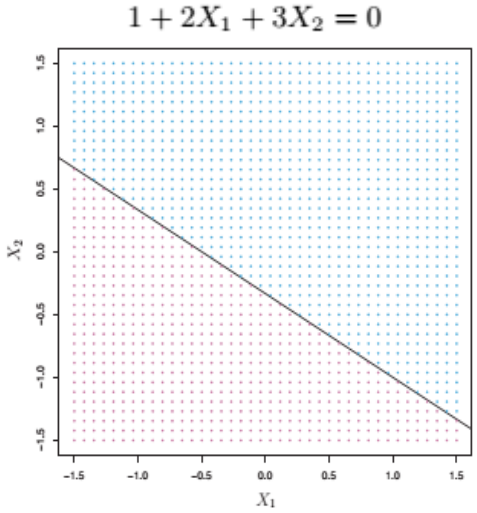
→ Multiple predictors

Maximal Margin Classifier

- *Hyperplane.* In a *p*-dimensional space, a *hyperplane* is a flat affine subspace of hyperplane dimension $p - 1$.
- In two dimensions, a hyperplane is defined by the equation.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0$$



p-dimensions

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p < 0$$

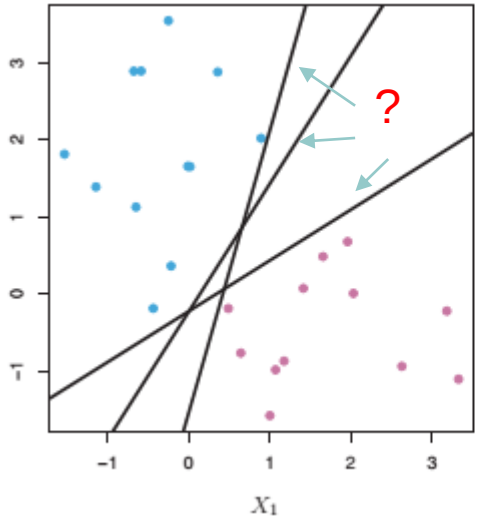
The example lies under the hyperplane

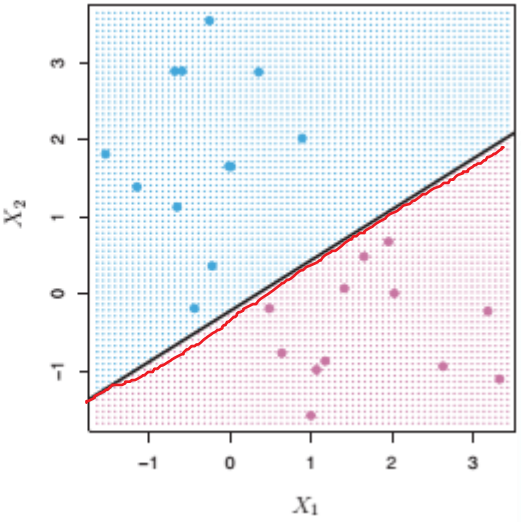
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p > 0$$

The example lies above the hyperplane

Classification Using a Separating Hyperplane

- *Separating hyperplane*. Suppose that it is possible to construct a hyperplane that separates the hyperplane training observations perfectly according to their class labels.





$$f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \ldots + \beta_p x_p^*$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) > 0$$

- If $f(x*)$ is positive, then we assign the test observation to class 1, and if $f(x*)$ is negative, then we assign it to class −1. We can also make use of the *magnitude* of $f(x*)$.
- If $f(x*)$ is far from zero, then this means that $x*$ lies far from the hyperplane, and so we can be confident about our class assignment for $x*$

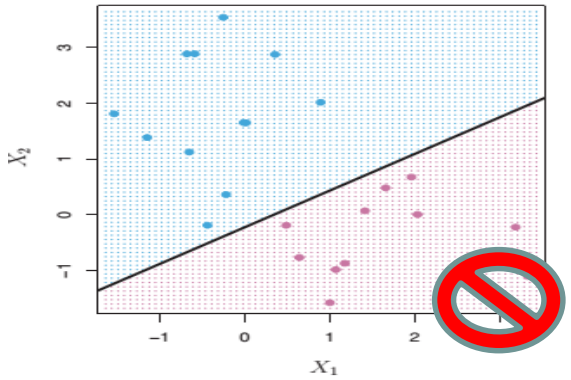
The Maximal Margin Classifier

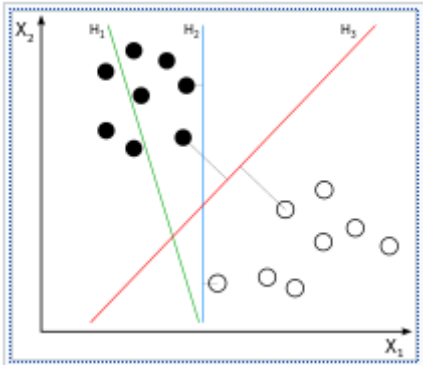
- In general, if our data can be perfectly separated using a hyperplane, then there will in fact exist an **infinite number of such *hyperplanes***.
- A natural choice is the ***maximal margin hyperplane*** (also known as the ***optimal separating hyperplane***), which is the separating hyperplane that is ***farthest*** from the training observations.
- We can compute the (perpendicular) distance from each training observation to a given separating hyperplane; the **smallest** such distance is the **minimal distance** from the observations to the hyperplane, and is known as the ***margin.***
- The **maximal margin hyperplane** is the separating hyperplane for which the margin is ***largest***.

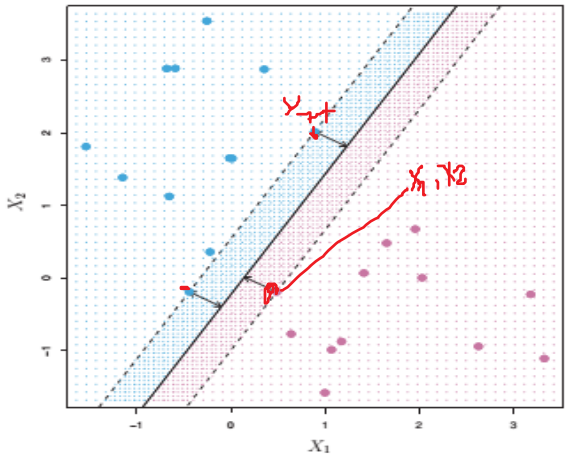
- *Separates two 'clouds' of points and is at equal distance from the two*

Support Vectors

- the maximal margin hyperplane represents the mid-line of the widest “slab” that we can insert between the two classes.







Support Vectors: they “support” the maximal margin hyperplane in the sense that if these points were moved slightly, then the maximal margin hyperplane would move as well. Interestingly, the maximal margin hyperplane depends directly on the support vectors

The maximal margin hyperplane depends directly on only a **small subset of the observations**

Added to ensure that the perpendicular distance of the point to the hyperplane is:

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip})$$

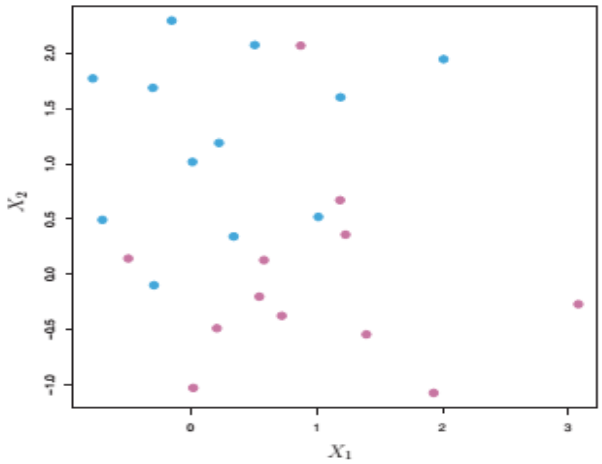
$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}}\, M$$

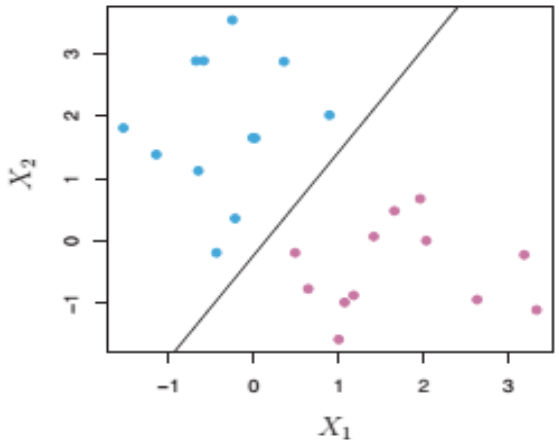
$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1,$$

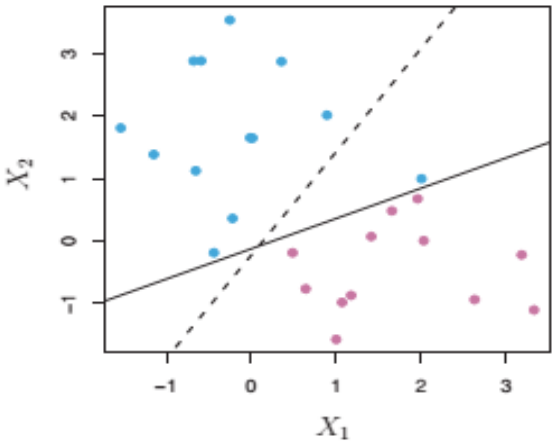
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \;\; \forall\, i = 1, \ldots, n.$$

The Non-separable Case

- As we have hinted, in many cases no separating hyperplane exists, and so there is no maximal margin classifier. In this case, the optimization problem showed before has no solution with $M > 0$.

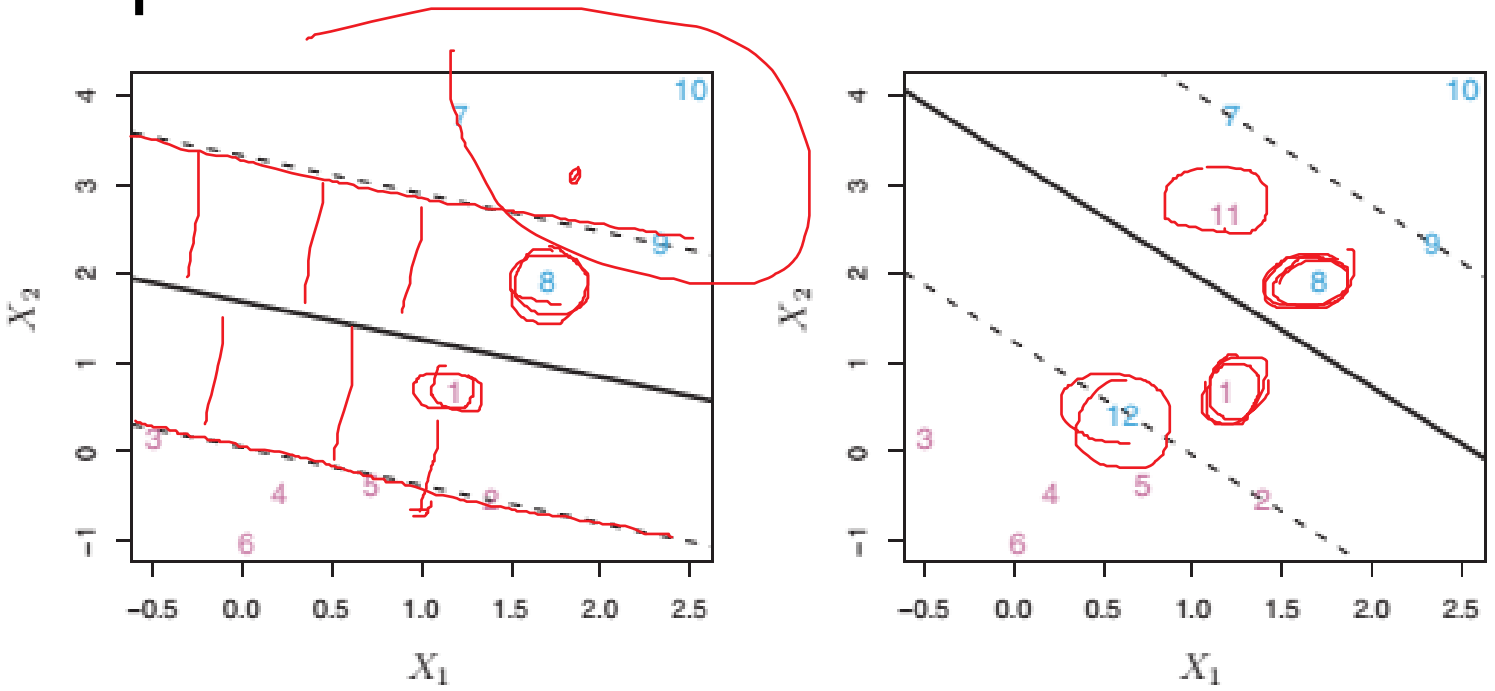






- That is, it could be worthwhile to misclassify a few training observations in order to do a better job in classifying the remaining observations.
- The ***support vector classifier***, sometimes called a *soft margin classifier*, does exactly this. Rather than seeking the largest possible margin so that every observation is not only on the correct side of the hyperplane but also on the correct side of the margin, we instead allow some observations to be on the incorrect side of the margin, or even the incorrect side of the hyperplane. (The margin is *soft* because it can be violated by some of the training observations.)

Support Vector Classifier



- ***Left***: *Observations* 3, 4, 5, *and* 6 *are on the correct side of the margin, observation* 2 *is on the margin, and observation* 1 *is on the wrong side of the margin.* Blue observations: *Observations* 7 *and* 10 *are on the correct side of the margin, observation* 9 *is on the margin, and observation* 8 *is on the wrong side of the margin. No observations are on the wrong side of the hyperplane.* ***Right***: *Same as left panel with two additional points,* 11 *and* 12. *These two observations are on the wrong side of the hyperplane and the wrong side of the margin.*

Support Vector Classifier

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C,$$

The slack variable ϵ_i tells us where the *i*th observation is located, relative to the hyperplane and relative to the margin. If $\epsilon_i = 0$ then the *i*th observation is on the correct side of the margin.

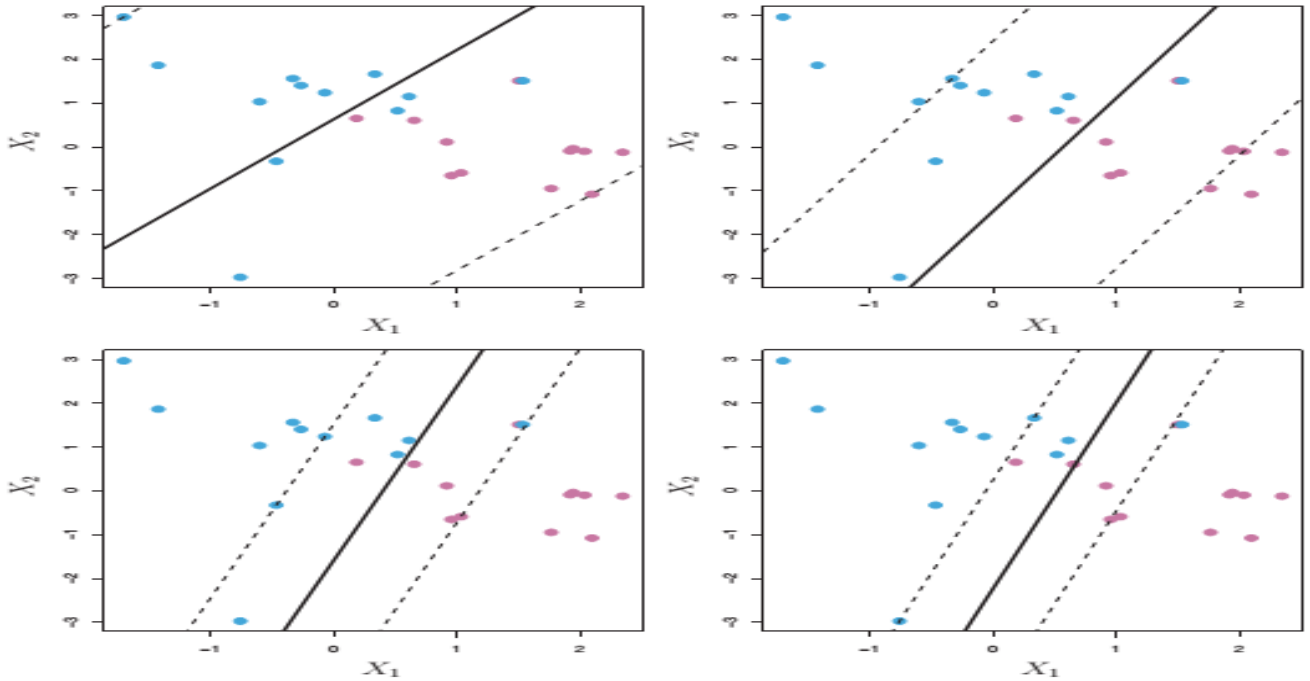
If $\epsilon_i > 0$ then the *i*th observation is on the wrong side of the margin, and we say that the *i*th observation has *violated* the margin.

If $\epsilon_i > 1$ then it is on the wrong side of the hyperplane.

C determines the number and severity of the violations to the margin (and to the hyperplane) that we will tolerate, so it is a *budget* for the amount that the margin can be violated by the *n* observations.
If *C* = 0 then there is no budget for violations to the margin. As the budget *C* increases, we become more tolerant of violations to the margin, and so the margin will widen. Conversely, as *C* decreases, we become less tolerant of violations to the margin and so the margin narrows.

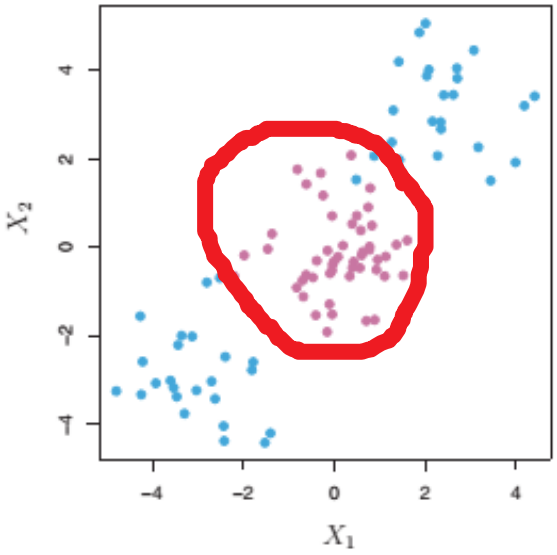
Support Vector Classifier

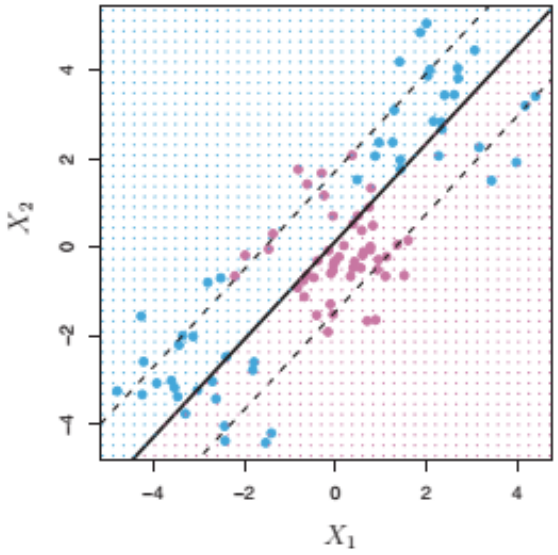
- In practice, *C* is treated as a **tuning parameter** that is generally chosen via cross-validation.
- *C* controls the **bias-variance trade-off of the statistical learning technique**. When *C* is small, we seek narrow margins that are rarely violated; this amounts to a classifier that is highly fit to the data, which may have low bias but high variance.
- On the other hand, when *C* is larger, the margin is wider and we allow more violations to it; this amounts to fitting the data less hard and obtaining a classifier that is potentially more biased but may have lower variance.



Support Vector Machines

- The support vector classifier is a natural approach for classification in this setting, if the boundary between the two classes is linear**. However, in practice we are sometimes faced with non-linear class boundaries.**
- In the case of the support vector classifier, we could address the problem of possibly non- linear boundaries between classes in a similar way, by enlarging the feature space using quadratic, cubic, and even higher-order polynomial functions of the predictors.





$$X_1, X_2, \ldots, X_p$$

$$\downarrow$$

$$X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2$$

$$\underset{\beta_0,\beta_{11},\beta_{12}\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

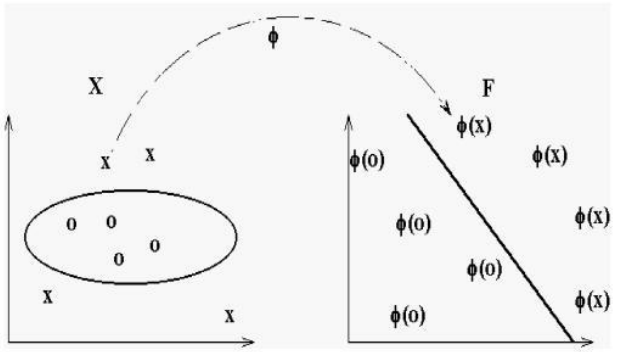
$$\text{subject to } y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2 \right) \geq M(1 - \epsilon_i)$$

$$\sum_{i=1}^{n} \epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1.$$

The Support Vector Machine

- The *support vector machine* (SVM) is an extension of the support vector classifier that results from enlarging the feature space in a specific way, using *kernels*.
- *Kernels* projects the original information to a space of more dimensions, so the input space X is mapped to a new space of a greater dimensionality (Hilbert):

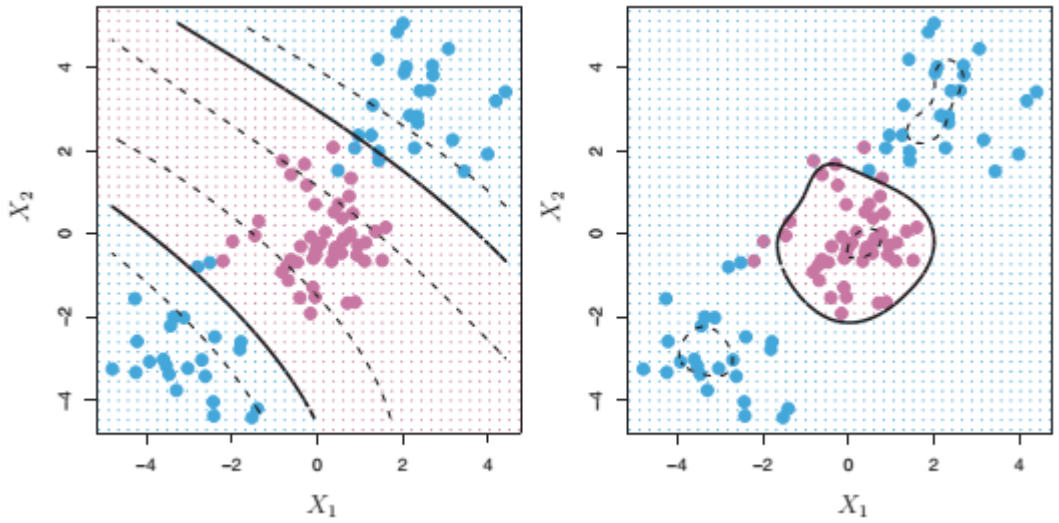
Kernel trick →



- $F = \{\varphi(x) | x \in X\}$
- $x = \{x_1, x_2, \cdots, x_n\} \rightarrow \varphi(x) = \{\varphi_1(x), \varphi_2(x), \cdots, \varphi_n(x)\}$
- When the **support vector classifier** is combined with a **non-linear kernel**, the resulting classifier is known as a ***support vector machine***

SVM

RBF



$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i)$$

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$$

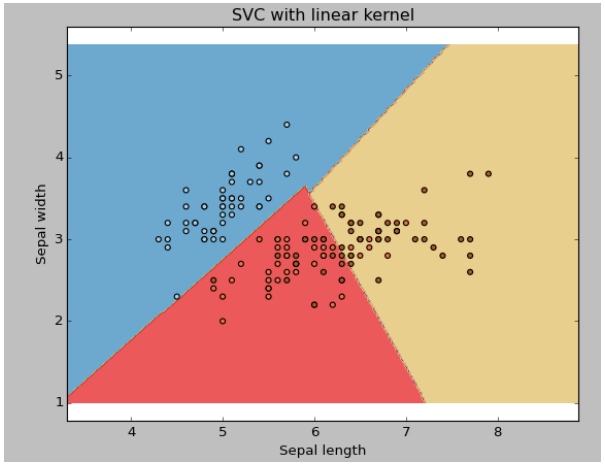
$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

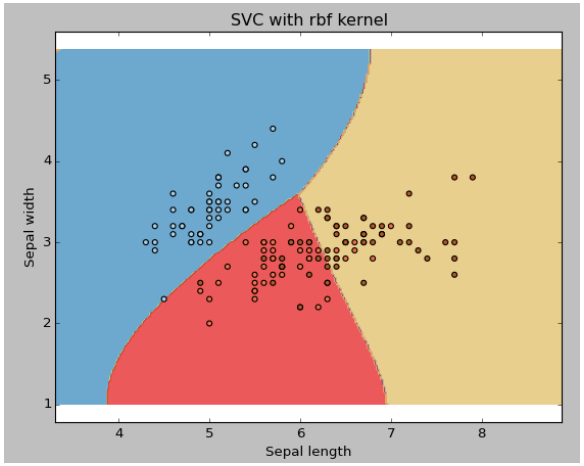
$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2).$$

- ***Left***: *An SVM with a polynomial kernel of degree 3.* ***Right***: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

SVM's with More than Two Classes

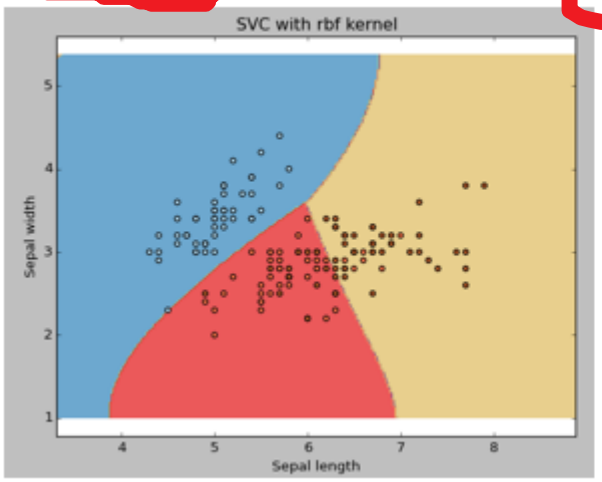
- ***One - Versus - One Classification****:* The final classification is performed by assigning the test observation to the class to which it was most frequently assigned in the $\frac{K}{2}$ pairwise classifications. ***(K(K−1)/2)***.
- ***One – Versus – All****:* We fit ***K SVMs***, each time comparing one of all the ***K*** classes to the remaining *K* − 1 classes. We assign the observation to the class for which **(1)** is largest, as this amounts to a high level of confidence that the test observation belongs to the *k*th class rather than to any of the other classes. (1) $\beta_{0k} + \beta_{1k}x_1^* + \beta_{2k}x_2^* + \ldots + \beta_{pk}x_p^*$



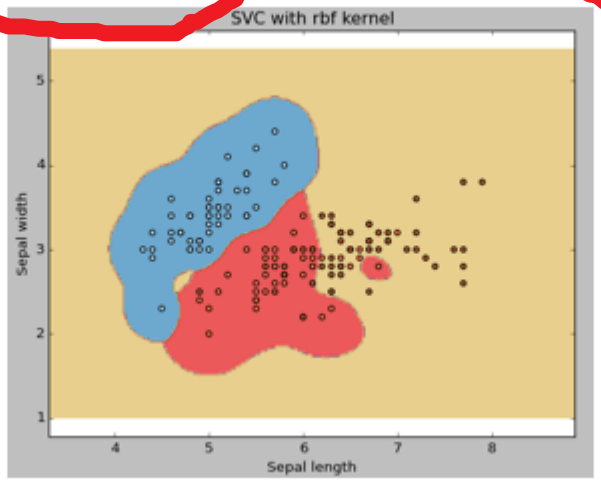


SVM

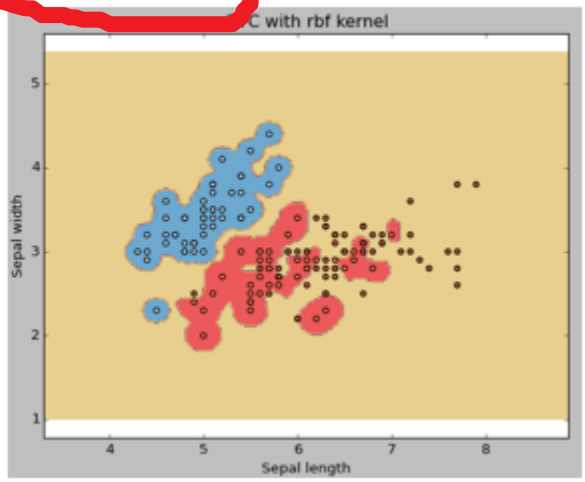
gamma =0



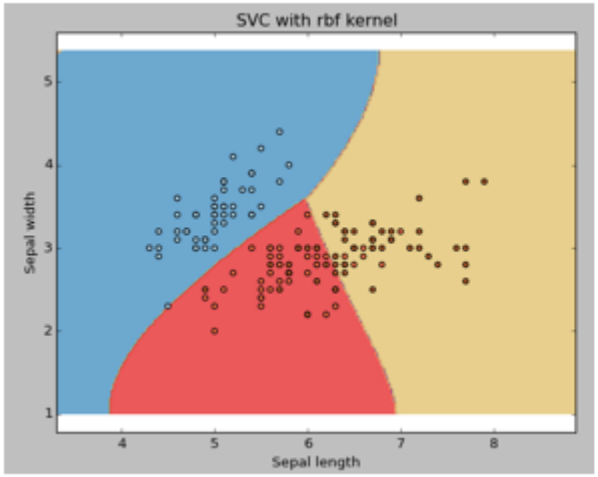
gamma =10



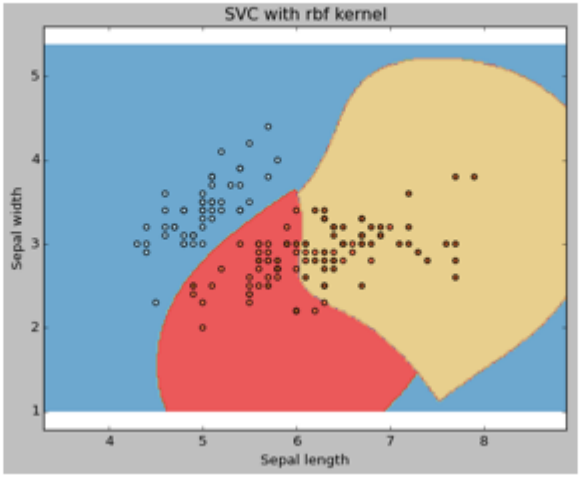
gamma =100



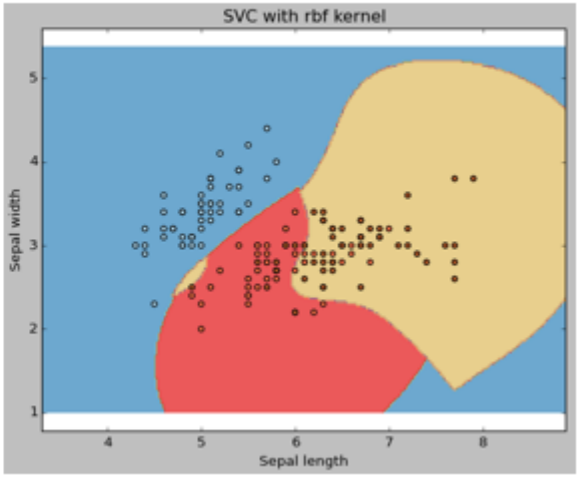
c =1



C =100



c =1000



Application Examples

- Spam filtering, a process which tries to discern E-mail spam messages from legitimate emails.
- Email routing, sending an email sent to a general address to a specific address or mailbox depending on topic
- Genre classification, automatically determining the genre of a text
- Readability assessment, automatically determining the degree of readability of a text, either to find suitable materials for different age groups or reader types or as part of a larger text simplification system
- Sentiment analysis, determining the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document.
- Health-related classification using social media in public health surveillance
- Article triage, selecting articles that are relevant for manual literature curation, for example as is being done as the first step to generate manually curated annotation databases in biology.

Application Examples

- Classification of images can also be performed using SVM.

Computers and Electronics in Agriculture 135 (2017) 81–95

Contents lists available at ScienceDirect

Computers and Electronics in Agriculture

journal homepage: www.elsevier.com/locate/compag

Image classification for detection of winter grapevine buds in natural conditions using scale-invariant features transform, bag of features and support vector machines

Diego Sebastián Pérez [a,b,*], Facundo Bromberg [c], Carlos Ariel Diaz [a]

[a] Universidad Tecnológica Nacional, Facultad Regional Mendoza, Laboratorio de Inteligencia Artificial DHARMa, Dpto. de Sistemas de la Información, Rodríguez 273, CP 5500 Mendoza, Argentina
[b] Universidad Nacional de Cuyo, Instituto universitario para las Tecnologías de la Información y las Comunicaciones, CONICET, Padre Jorge Contreras 1300, CP 5500 Mendoza, Argentina
[c] Universidad Tecnológica Nacional, Facultad Regional Mendoza, CONICET, Laboratorio de Inteligencia Artificial DHARMa, Dpto. de Sistemas de la Información, Rodríguez 273, CP 5500 Mendoza, Argentina

ARTICLE INFO

Article history:
Received 30 April 2016
Received in revised form 6 January 2017
Accepted 23 January 2017
Available online 10 February 2017

ABSTRACT

In viticulture, there are several applications where bud detection in vineyard images is a necessary task, susceptible of being automated through the use of computer vision methods. A common and effective family of visual detection algorithms are the *scanning-window* type, that slide a (usually) fixed size window along the original image, classifying each resulting windowed-patch as containing or not containing the target object. The simplicity of these algorithms finds its most challenging aspect in the classification