2. Regression and Classification

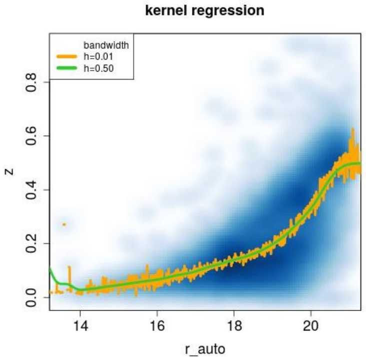
Laerte Sodré Jr.
IAG – Universidade de São Paulo

IX La Plata International School (LAPIS) on Astronomy and Geophysics
S-PLUS: The Universe in True Colors
17-21 February 2020, La Plata, Argentina

regression and classification

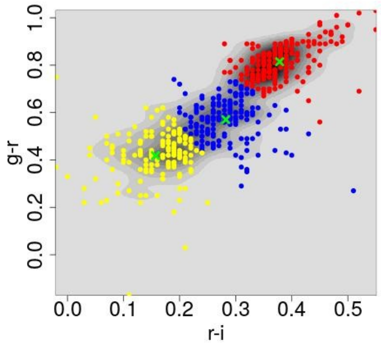
regression in ML:

- estimation of a ***continuous variable***, **y**, from data **x** in a training set
- examples: linear, kernel, Gaussian Process



classification in ML:

- estimation of a ***categorical variable***, **y**, from data **x** in a training set
- examples: k-nn, logistic, SVM, trees, random forest



what is regression?

regression in ML:

- estimation of a continuous variable, y, from data x in a training set

- ex.: photo-z estimation from galaxy colors

- model:

  $y = f(x;w) + \varepsilon$

  w: model parameters

  ε: error or noise

we need:

- loss/cost function **l(w)**: to evaluate the quality of model fitting

- optimizer: find **w** by minimization of the loss function

noise:

- models do not fit the data perfectly:

  model: y = f(x;w) + ε
  ε: error or noise

- ε can be due to measurement errors in x and/or y

- ε can be due to the inadequacy of the model (too simple, too complicate)

- ...

example: *ordinary linear regression*

- linear model (in the parameters!):
  $y = w_0+w_1x$
  parameters: $\{w_0,w_1\}$

- cost/loss function: sum of the squares of the residuals

$$\mathrm{l}(\mathbf{w}) \propto \chi^2 \propto \Sigma_{\mathbf{i}=1}^{\mathrm{N}} [\mathbf{y}_\mathrm{i} - (\mathbf{w}_0 + \mathbf{w}_1\mathbf{x}_\mathrm{i})]^2$$

- optimization: minimization of the least squares

- ML main optimizer: gradient descent

learning/optimization with gradient descent

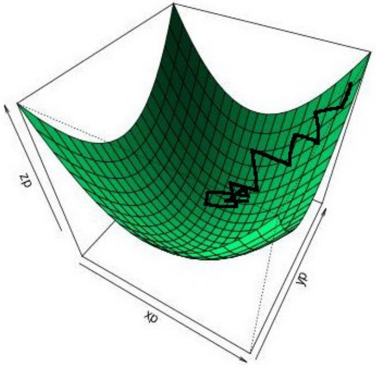
optimization of the cost/loss l(w):

$$l(\mathbf{w}) \propto \chi^2 \propto \Sigma_{i=1}^{N} \left[y_i - (w_0 + w_1 x_i)\right]^2$$

- initialization: random values for w
- learning based on the gradient:

update of w: $w \leftarrow w - \lambda\, \partial l(w)/\partial w$

λ: ‘learning rate’



for a single datum:

- $w \leftarrow w + 2\lambda\, [y_i - y(x_i;w)]$

  […] is the residual with the current value of the parameters w

- as the training proceeds, it decreases and (hopefuly!) converges to a stationary value

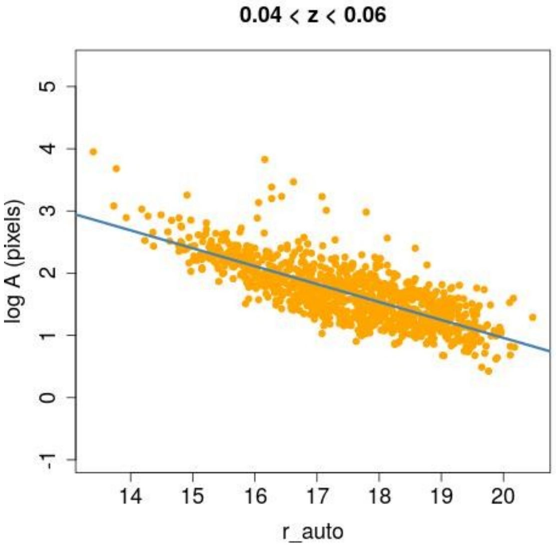
optimization strategies:

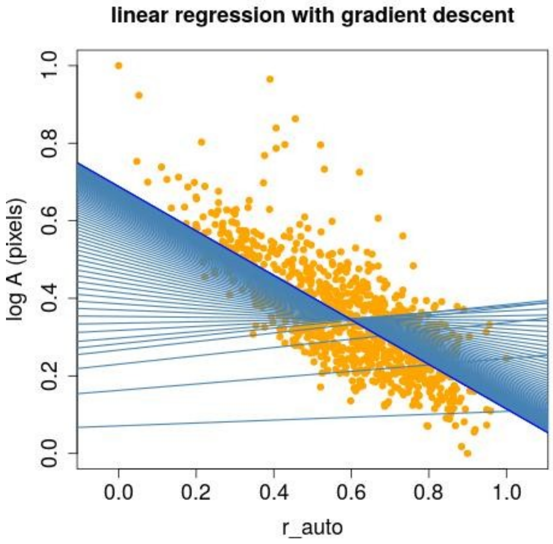
- batch: update w after presentation of all data
- mini-batch: update w with n random objects
- stochastic: update w after each object

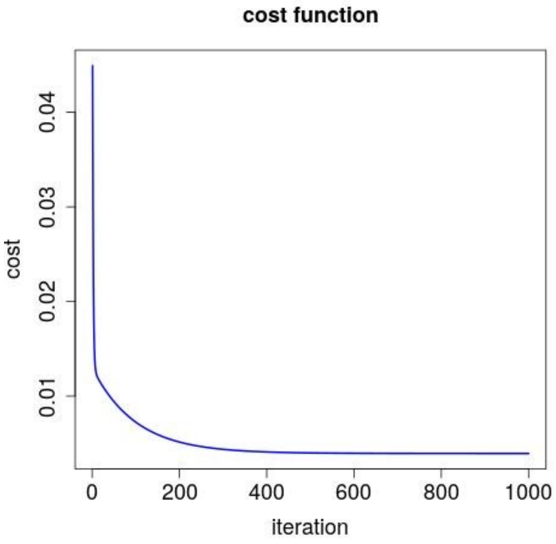
learning/optimization with gradient descent

- **example: linear regression**
  A (semi-major axis) x r_auto

$$w \leftarrow w + 2\lambda\,[y_i - y(x_i;w)]$$


0.04 < z < 0.06


linear regression with gradient descent


cost function

models and generalization

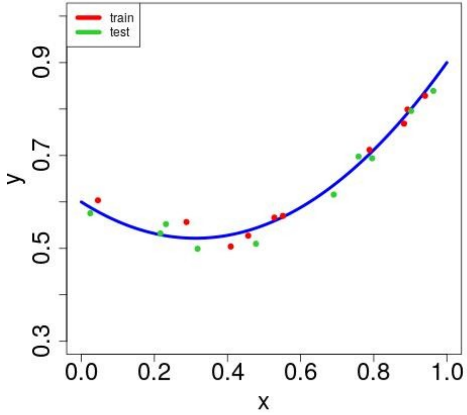
models should have the ‘right’ complexity (or ‘capacity’):

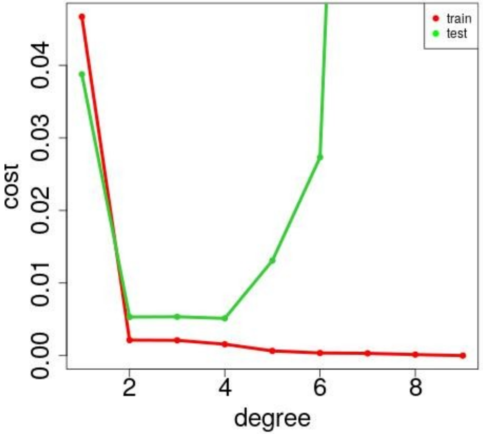
- models too simple: underfitting
- models too complex: overfitting

→ the models fit the noise!

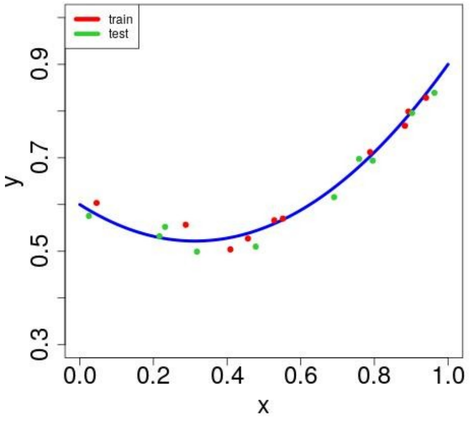
- example: fitting of a polynomium of degree M

- the polynomium is fitted with the training set and then applied to a test set





models and generalization

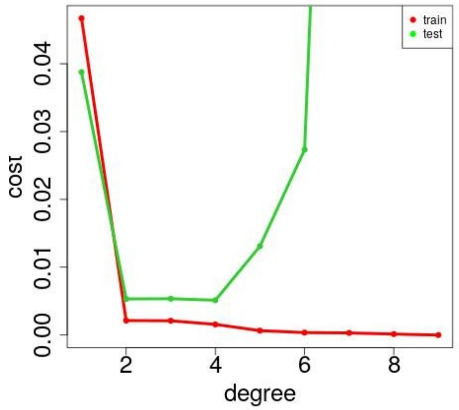


what happens for high values of M?

- the models fits the noise: w ‘explode’!

- example: weights for M = 9
30.87, -1122.61, 13019.71, -75589.66, 256956.84, -544754.35, 730907.60, -603984.86, 280679.44, -56144.84

- a way to prevent overfitting: ‘*regularization*’
it constrains the size of the weigths



regularization with a new term in the cost function

- $l(w) = \chi^2 / 2 + \alpha\, w^T.w$

  α: regularization parameter

- the additional term penalizes large absolute values of **w**

- linear model: “ridge regression”
  $w = (x^Tx+\alpha I)^{-1}x^Ty$

- gradient descent for a single datum:
  $w = w + 2\lambda\, [y_i – y(x_i;w) - \alpha w]$

- LASSO: least absolute shrinkage and selection

  $l(w) = \chi^2 / 2 + \alpha\, |w|$

- notice that OLS is a particular case of ridge regression and LASSO

kernel regression

- **K(u)**: kernel   **h**: bandwidth

$$\mathbf{y} = \mathbf{f}(\mathbf{x}|\mathbf{K}) = \frac{\sum_{\mathbf{i}=1}^{\mathbf{N}} \mathbf{K}\left(|\mathbf{x} - \mathbf{x_i}|/\mathbf{h}\right) \mathbf{y_i}}{\sum_{\mathbf{i}=1}^{\mathbf{N}} \mathbf{K}\left(|\mathbf{x} - \mathbf{x_i}|/\mathbf{h}\right)} = $$

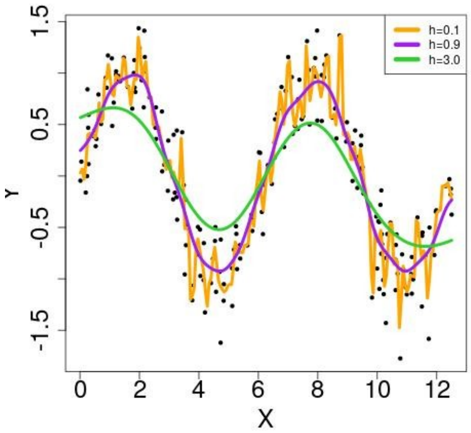
$$= \sum_{\mathbf{i}=1}^{\mathbf{N}} \mathbf{w_i}(\mathbf{x}) \mathbf{y_i}$$

- kind of "local regression": weighted mean of **y** with weights

$$\mathbf{w_i}(\mathbf{x}) = \frac{\mathbf{K}\left(|\mathbf{x} - \mathbf{x_i}|/\mathbf{h}\right) \mathbf{y_i}}{\sum_{\mathbf{i}=1}^{\mathbf{N}} \mathbf{K}\left(|\mathbf{x} - \mathbf{x_i}|/\mathbf{h}\right)}$$

- many variants: locally linear regression, locally polynomial regression, adaptive kernel regression

- **h** can be determined by cross-validation



Gaussian Process (GP) regression

modeling of functions:

parametric scenario: $y = f(x;w) + \varepsilon$

- the functional for **f** is assumed known
- regression: estimation of the parameters **w**

functional space scenario: $f \sim GP(\mu,k)$

- **f** is assumed sampled from a "functional space"
- regression: estimation of the posterior of the values of the functions at the points of interest

- Gaussian distribution vector sampling:

$$f = \{f_1 ... f_N\} \sim N(\mu,\Sigma)$$

- GP- function sampling:

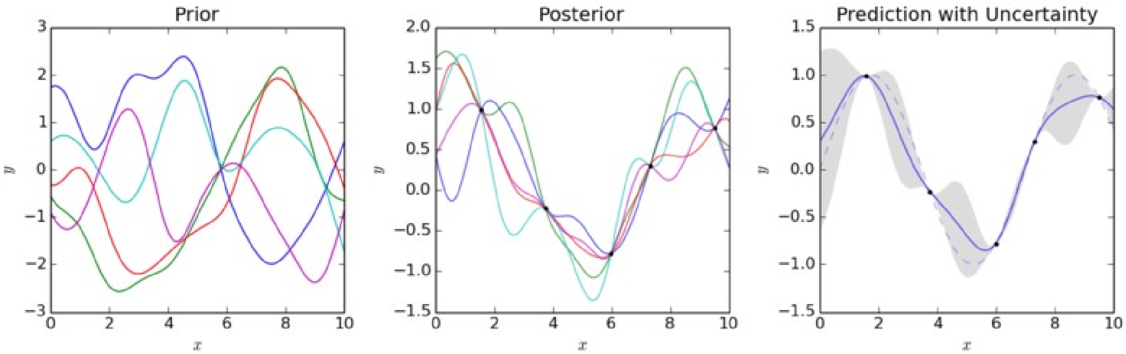
$$f(x) \sim GP(m(x),k(x,x'))$$

GP is a Gaussian process of mean $m = E(f(x))$ and covariance $k(x,x')$

- GP: f(x) ~ GP(m(x),k(x,x’))

- example: radial basis functions

$$\mathbf{k}(\mathbf{x},\mathbf{x}') = \sigma^2 \exp\left(-\frac{|\mathbf{x}-\mathbf{x}'|}{\mathbf{2}\lambda^{\mathbf{2}}}\right)$$



- main hyperparameters:
  λ controls the horizontal scale
  σ controls the vertical scale

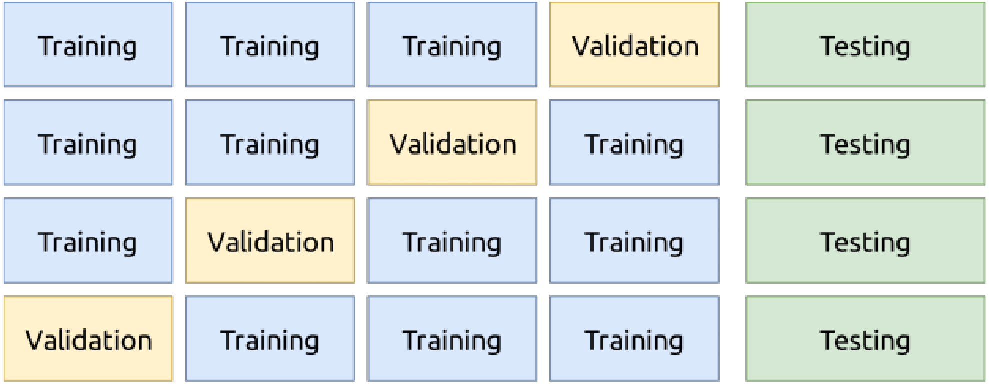
- problem: matrix inversions α N^3
  solution: sparse approximation of the data

(deep GP: a GP that is a function of another GP)

cross-validation

- estimation of “model errors” and/or of model hyperparameters

- simple CV: estimate model reliability with a test set



Training	Training	Training	Validation	Testing
Training	Training	Validation	Training	Testing
Training	Validation	Training	Training	Testing
Validation	Training	Training	Training	Testing

K-fold CV:

- data is divided in K+1 subsamples

- train K models and let one subsample aside to measure their errors

- errors in the model can be estimated from the median of the errors in each subsample

- hyperparameters can be chosen by K-fold CV plus grid search

Classification

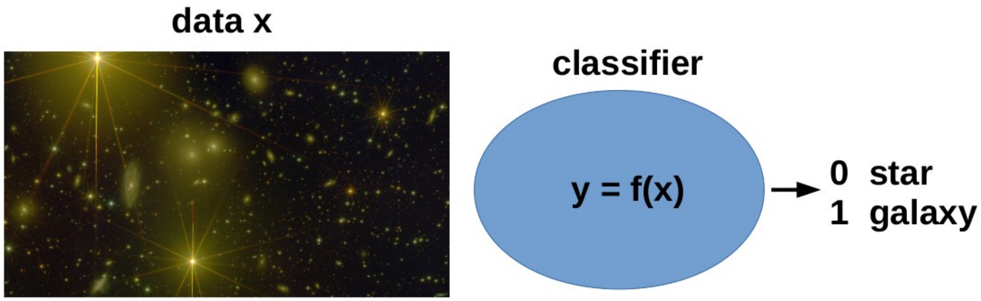
what is classification?

- sample of N objects with D features x
- objective: to determine the label or class of the object

Examples:

- Hubble types: E, S0, Sbc…
- BPT types: SF, Sey1, LINERs, transition
- detected/non-detected
- Star/galaxy
- 0/1

- the labels are categorical variables
- the data features can be real or categorical
- classification: we train a function with a training and validation sets

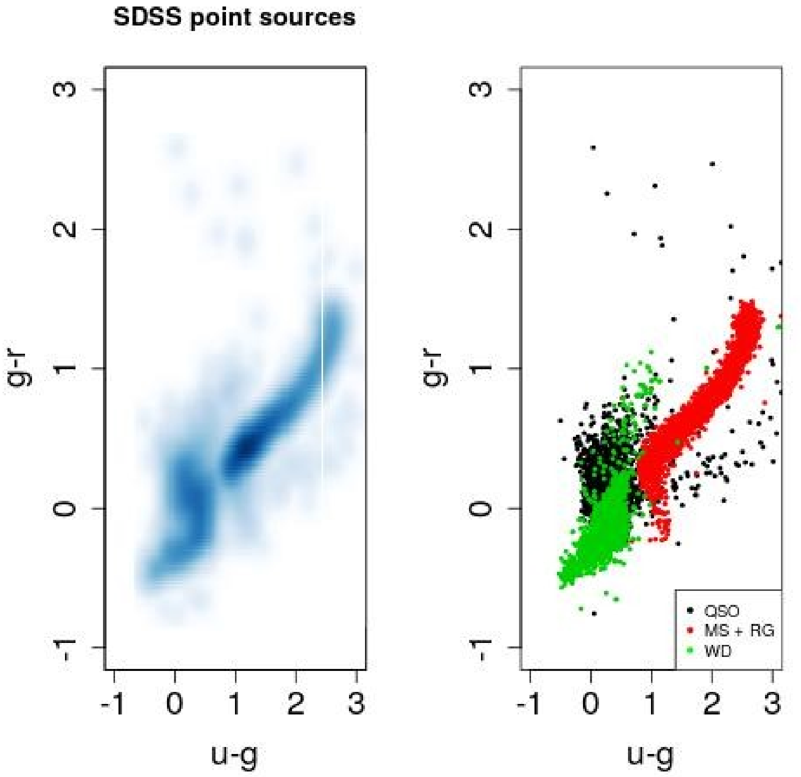


what is classification?

- classification: binary or multiclass

  Problems:
- classes are often not cleanly separable

- imbalance in the training set: very different numbers of objects in each class
  → bias toward the majority class!



cost function

- **cost 0/1: we give 0 to a correct classification and 1 to a wrong one**

  If ŷ is the estimate of y:

$$\mathbf{L}(\mathbf{y},\hat{\mathbf{y}}) = \begin{cases} 1 & \textbf{if } \hat{y} \neq y \\ 0 & \textbf{if } \hat{y} = y \end{cases}$$

  classification risk (= error rate):
  E[L(y,ŷ)] = prob(y ≠ ŷ)

- **cross-entropy (multiclass):**

$$\mathbf{CE}(\mathbf{y},\hat{\mathbf{y}}) = -\frac{\mathbf{1}}{\mathbf{N}}\Sigma_{\mathrm{i=1}}^{\mathrm{N}}\left[\mathbf{y}_{\mathrm{i}}\log\hat{\mathbf{y}}_{\mathrm{i}} + (\mathbf{1}-\mathbf{y}_{\mathrm{i}})\log(\mathbf{1}-\hat{\mathbf{y}}_{\mathrm{i}})\right]$$

- **mean square error:**

$$\frac{\mathbf{1}}{\mathbf{N}}\Sigma_{\mathrm{i=1}}^{\mathrm{N}}\left(\hat{\mathbf{y}}_{\mathrm{i}}-\mathbf{y}_{\mathrm{i}}\right)^{\mathbf{2}}$$

- **binary classification (0,1):**

	prediction = 1	prediction = 0
measurement = 1	TP: True Positive	FN: False Negative error type II
measurement = 0	FP: False Positive error type I	TN: True Negative

binary classification: completeness and contamination

- detection (1)   non-detection (0)
- completeness (recall): fraction of detections

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

- contamination: fraction of wrong detections

$$\frac{\mathrm{FP}}{\mathrm{TP} + \mathrm{FP}}$$

- accuracy: fraction of correct detections

$$\frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

- error rate (misclassification)

$$\frac{\mathrm{FP} + \mathrm{FN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

- precision (positive predictive value) = 1-contamination

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

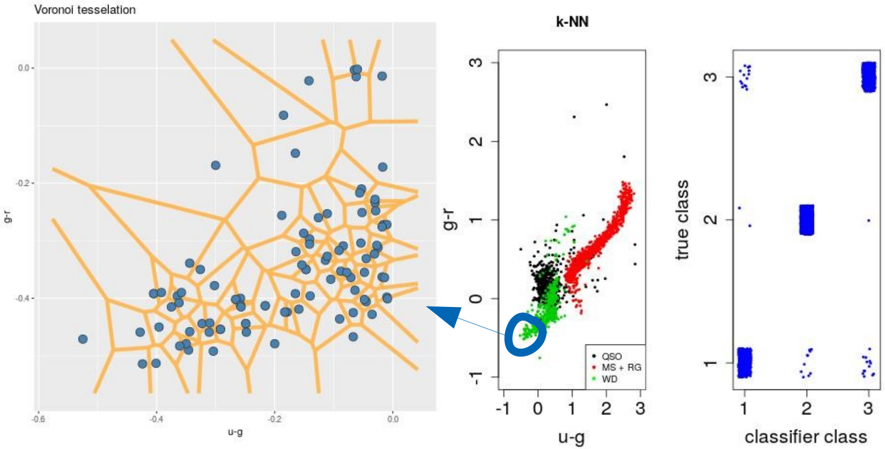
- F_1 score: harmonic mean of precision and completeness

$$\mathrm{F}_1 = 2\frac{\mathrm{P} \times \mathrm{R}}{\mathrm{P} + \mathrm{R}}$$

- ***Depending of the problem we may want to optimize the completeness, or the accuracy, or the precision...***

Classification with k nearest neighbors (k-NN)

- adopt the class of the nearest neighbor in the training set
- the decision limits between classes form a Voronoi Tesselation
- non-parametric method



variants:

- adopt the most frequent class among k nearest neighbors in the training set
- weight by the neighbor distance

(method sensitive to the definition of distance)

- k can be obtained by CV

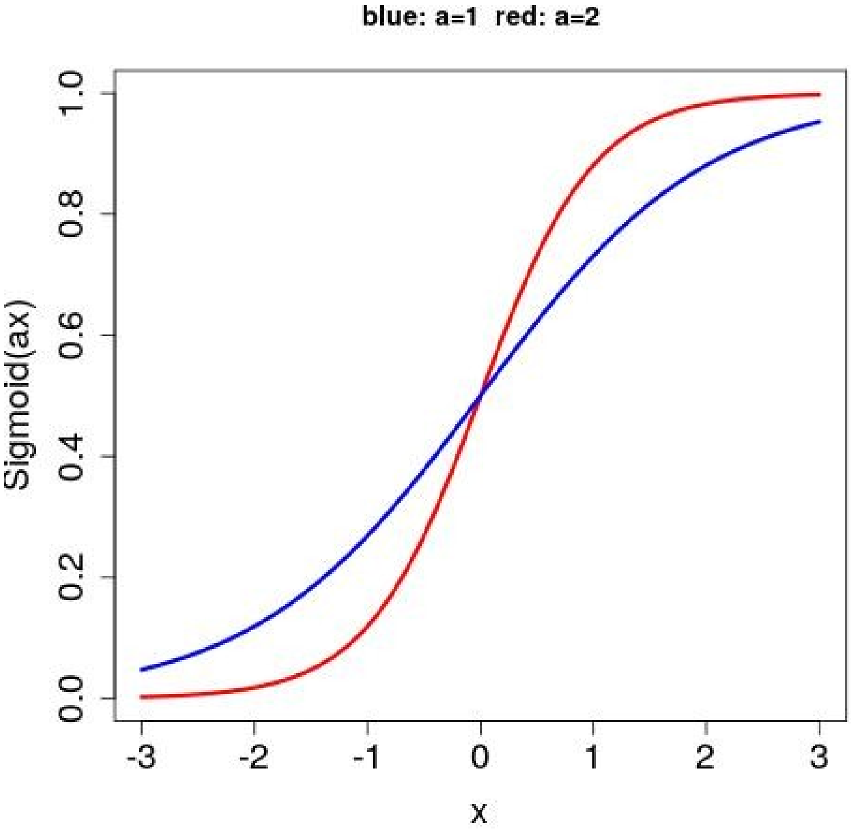
classification with logistic regression

- Classification in two classes: 0 or 1
- probability of class 1 for a certain object:

$$p(y=1|x_i) = S(\Sigma_j w_j x_{ij}) = f(x|w)$$

- S(x): sigmoid or logistic function

$$S(x) = \frac{1}{1+e^{-x}} \qquad 0 < S(x) < 1$$



classification with logistic regression

- Classification in two classes: 0 or 1
- probability of class 1 for a certain object:

$$p(y = 1|x_i) = S(\Sigma_j w_j x_{ij}) = f(x|w)$$

- S(x): sigmoid or logistic function

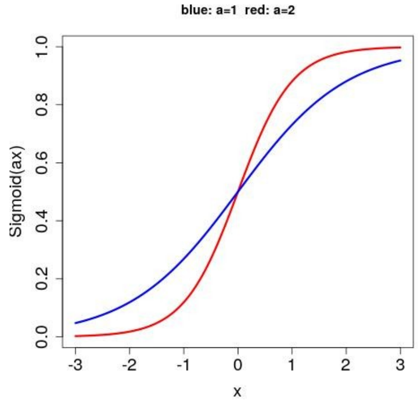
$$S(x) = \frac{1}{1 + e^{-x}} \qquad 0 < S(x) < 1$$

cost function:

- since y is binary, the cost can be modelled as a Bernoulli function:

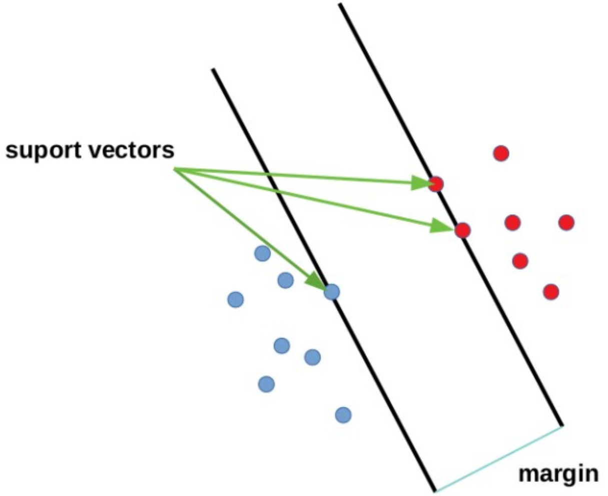
$$l(w) = \prod_{i=1}^{N} p_i(w)^{y_i}(1 - p_i(w))^{1-y_i}$$

- w is obtained by minimazing **l(w)**, i.e., the classification error

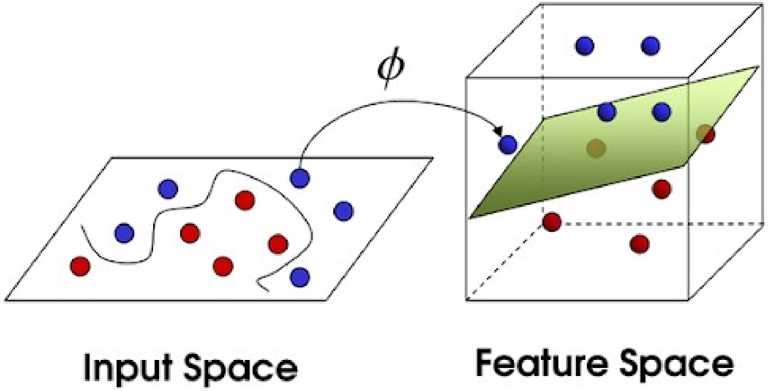


classification with SVM
support vector machine

- margin: hyperplane which maximizes the distance to the closest points of other classes
- the concept applies even when different classes overlap each other
- points in the margin define the *support vectors*

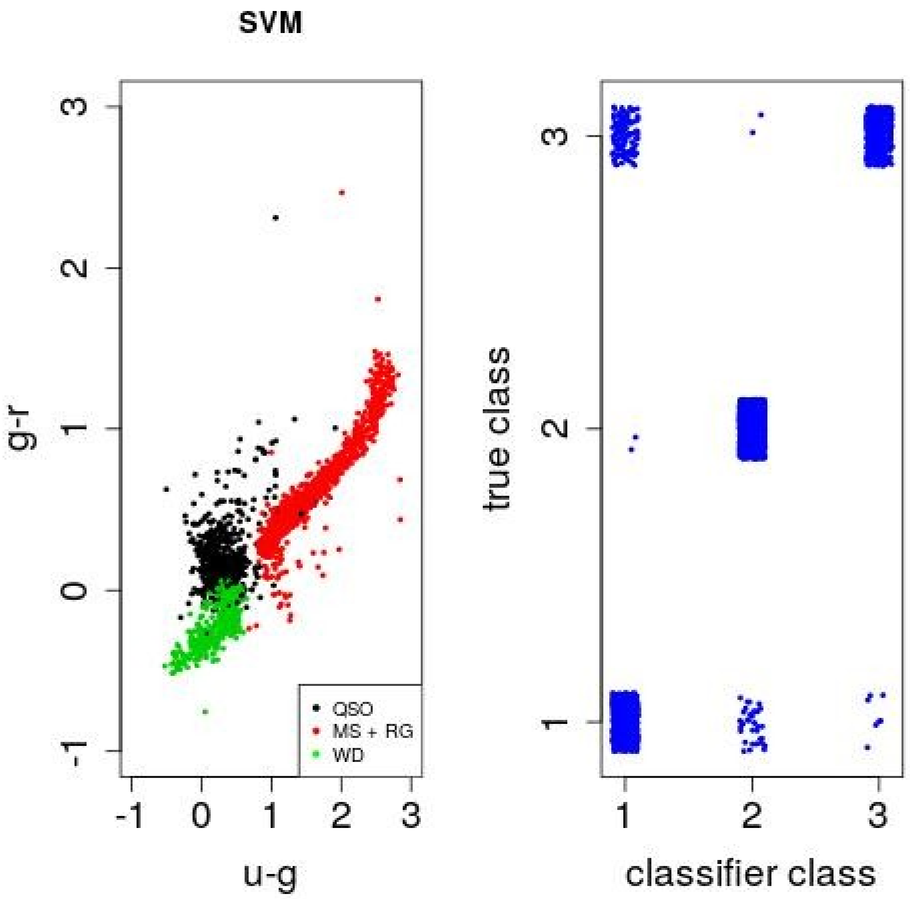


- to reduce contamination in the data space, SVM transforms the data to a higher dimensional space
- this is done using kernels (the “kernel trick”)



classification with SVM support vector machine

- SVM is less sensitive to imbalance than other methods
- can achieve high completeness but with high contamination



classification with decision trees

- decision trees: nodes, branches, leaves

- the top node contains all data

- at each level of the tree the nodes are divided in two (or more) branches

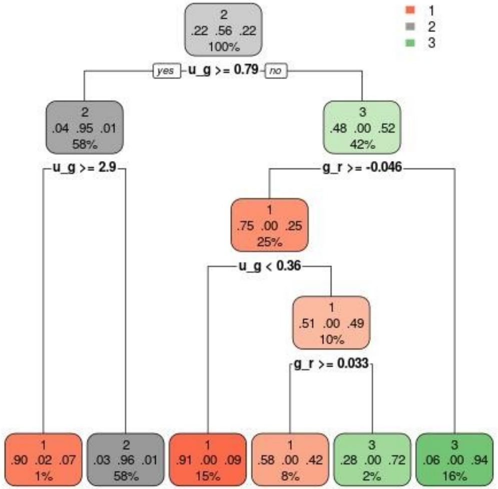
- the divisions are based on *decision limits*: values above the limit go to a branch, and values below the limit to the other branch

- the divisions proceed until a convergence criterion is achieved

- the terminal nodes- *leaves*- register the fraction of data points within each class

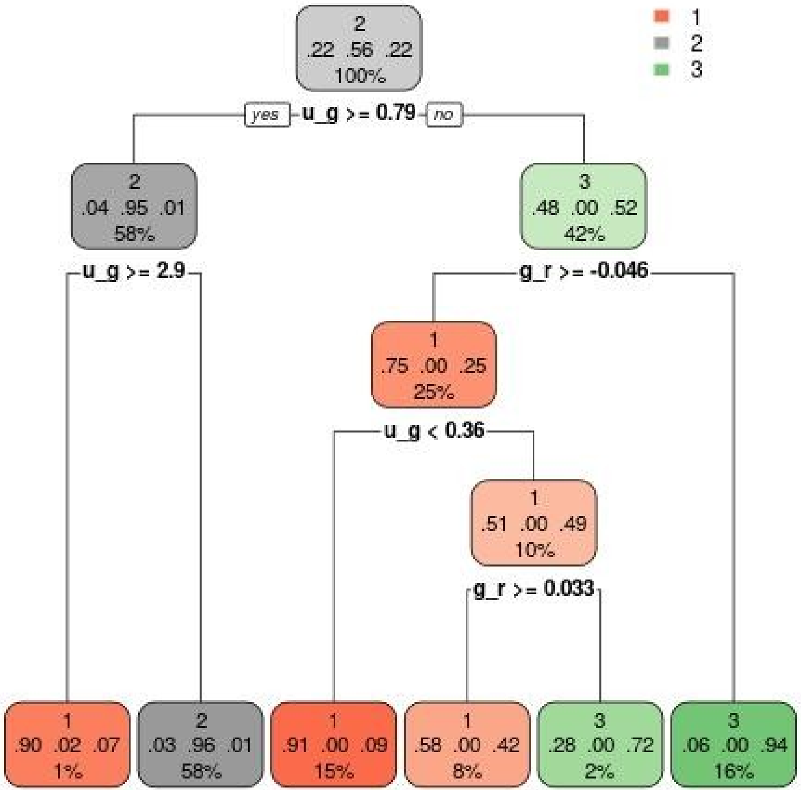
- classification of a leave: majoritary class

- classification of a new datum: follow the branches through binary decisions until arriving in a leave



classification with decision trees

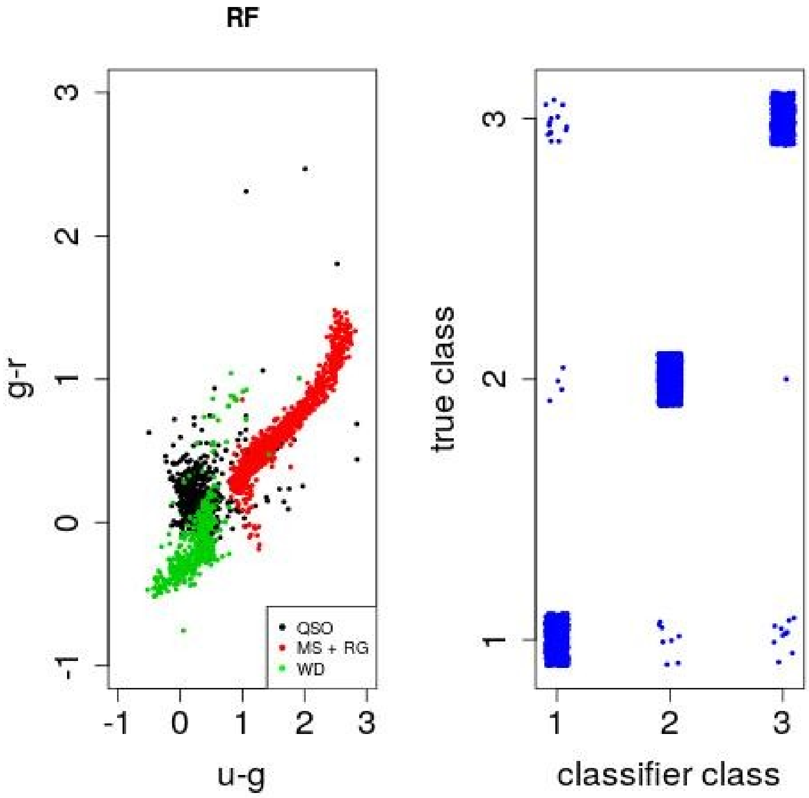
- complexity of the tree: number of levels, or depth
- controling the tree growth may be necessary to avoid overfitting
- tree pruning: use CV to remove nodes that do not contribute much to the result



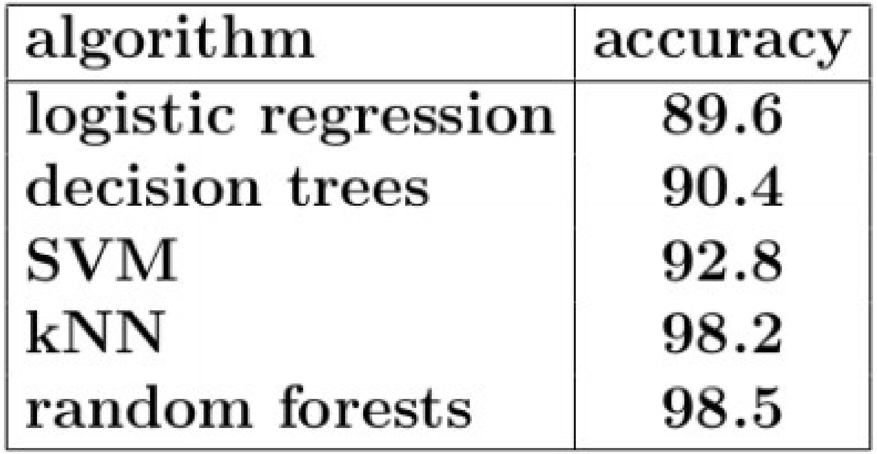
classification with random forest

random forest: type of ***ensemble learning:*** combination of results of several models

- generates many decision trees, each using only a subset of the data features
- the final classification is the mean of the classifications of the decision trees
- parameters: **n**- number of trees and **m**- number of features per tree
- **m** small reduces overfitting and improves predictivity
- **n** and **m** can be obtained by CV



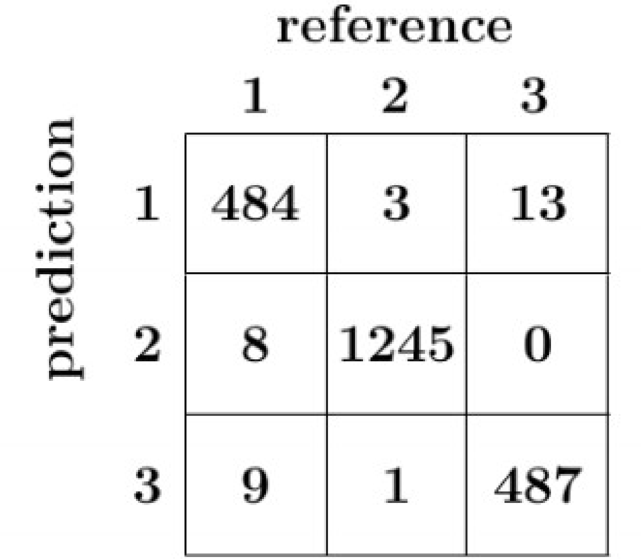
comparison of classifiers



algorithm	accuracy
logistic regression	89.6
decision trees	90.4
SVM	92.8
kNN	98.2
random forests	98.5

caution: no free lunch theorem!

RF: confusion matrix



		reference		
		1	2	3
prediction	1	484	3	13
	2	8	1245	0
	3	9	1	487