Machine Learning (Semester 1 2024)

Classification

Nina Hernitschek
Centro de Astronomía CITEVA
Universidad de Antofagasta

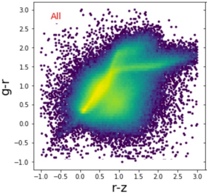
June 11, 2024

Motivation

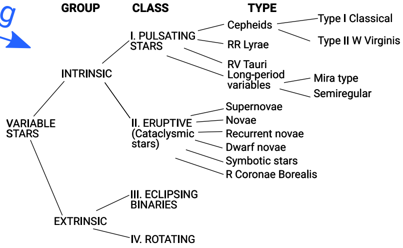
In the previous sessions, we saw that various approaches and applications involving **regression**.

We will focus now on **classification problems**:

parameter space of measurements



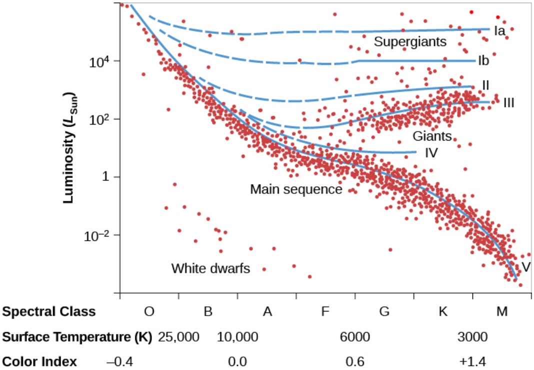
parameter space of astrophysical objects



Regression vs. Classification

The linear regression model discussed so far assumes that the response variable Y is **quantitative**.

But in many situations, the response variable is instead **qualitative**, also referred to as **categorical**.
For example, *stellar spectral class* is qualitative.



Regression vs. Classification

Predicting a qualitative response for an observation can be referred to as classifying that observation, since it involves assigning the observation to a category, or class.

In many cases, the methods used for classification first predict the **probability** that the observation belongs to each of the categories of a qualitative variable, as the basis for making the classification.

e.g.:
$P_{Quasar}(X) = 0.55$
$P_{Star}(X) = 0.34$
$P_{other}(X) = 0.21$

In this sense they behave similar to regression methods.

Classification Problems

Classification problems occur often, perhaps even more so than regression:

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?

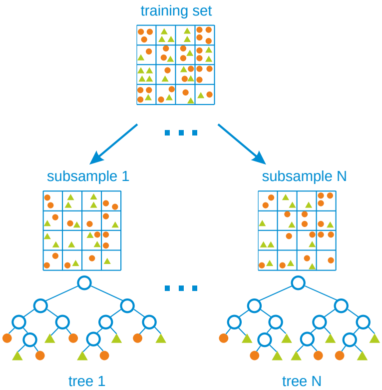
An online banking service must be able to determine whether or not a transaction being performed on the site is fraudulent, on the basis of the user's IP address, past transaction history...

A person must be identified from a camera image or video to allow or deny access to a building.

An astronomical survey can contain billions of objects. They must be classified to provide researchers with data for e.g. stars, quasars, galaxies...

Supervised Machine Learning

The classification problems mentioned belong into the regime of **supervised classification**, where we actually know the 'truth' for a subset of our data and use that to *train* a classifier.



Supervised vs. Unsupervised Machine Learning

Goals:

- In **supervised learning**, the goal is to predict outcomes for new data.
- In **unsupervised learning**, the goal is to get insights from large volumes of new data, were machine learning itself determines what is *interesting* from the dataset.

Supervised vs. Unsupervised Machine Learning

Goals:

- In **supervised learning**, the goal is to predict outcomes for new data.
- In **unsupervised learning**, the goal is to get insights from large volumes of new data, were machine learning itself determines what is *interesting* from the dataset.

Applications:

- **Supervised learning** models are ideal for e.g. astronomical source classification, e-mail spam detection, weather forecasting.
- In contrast, **unsupervised learning** is a great fit for anomaly detection, recommendation engines, and medical imaging.

The Role of the Training Set

The objective of a supervised learning model is to predict the correct label for newly presented input data.

The Role of the Training Set

The objective of a supervised learning model is to predict the correct label for newly presented input data.

When training a supervised learning algorithm, the **training set** will consist of inputs paired with the correct outputs. Inputs in the training set should represent the **target set** which we have to classify: composition of the data, data quality.

The Role of the Training Set

The objective of a supervised learning model is to predict the correct label for newly presented input data.

When training a supervised learning algorithm, the **training set** will consist of inputs paired with the correct outputs. Inputs in the training set should represent the **target set** which we have to classify: composition of the data, data quality.

During **training**, the algorithm will search for patterns in the data that correlate with the desired outputs.

The Role of the Training Set

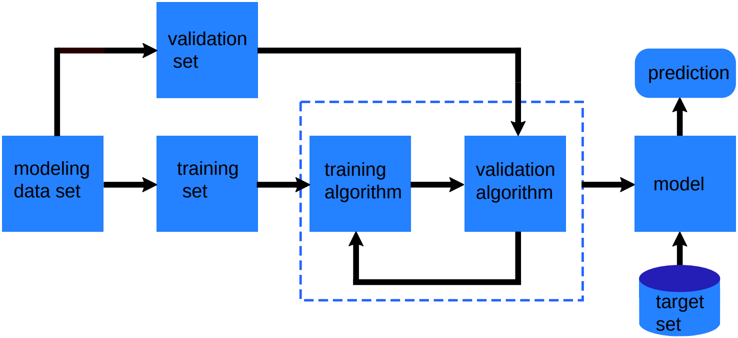
The objective of a supervised learning model is to predict the correct label for newly presented input data.

When training a supervised learning algorithm, the **training set** will consist of inputs paired with the correct outputs. Inputs in the training set should represent the **target set** which we have to classify: composition of the data, data quality.

During **training**, the algorithm will search for patterns in the data that correlate with the desired outputs.

After training, a supervised learning algorithm will take in new unseen inputs and will determine which label the new inputs will be classified based on prior training data.

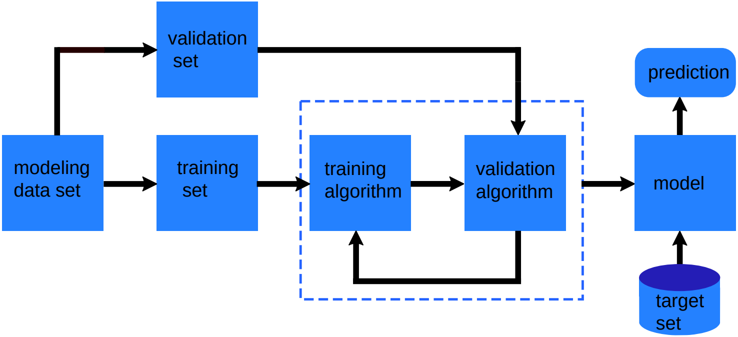
Classification Workflow



split modeling set into training set and validation set:
the validation set (also: test set) is used for the testing the model after the model has been trained on the training set - it is extremely important to test the model on data not being part of the training set

A **fundamental assumption** of supervised machine learning is that the distribution of training examples is identical to the distribution of validation examples and future unseen examples (the target set).

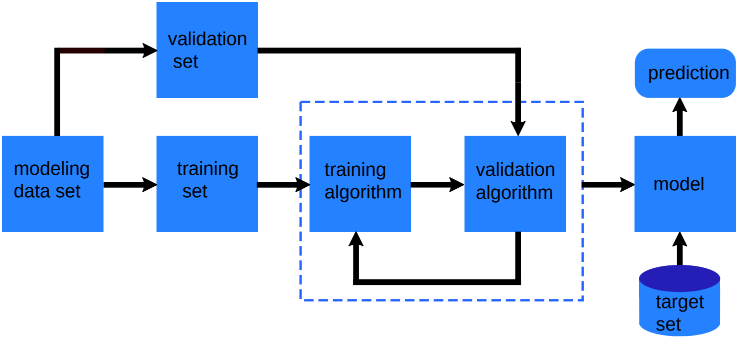
Classification Workflow



Training:
given a training set of labeled examples $\{(x_1, y_1), ..., (x_n, y_n)\}$, estimate the prediction function f and parameters θ which minimizes the prediction error on the training set

Validation:
apply f to validation set x, output predicted value $y = f(x)$
from this we generate performance measures, also called accuracy measures

Classification Workflow



Application:
Run the model on the target set.

Pitfalls in Classification

Over- and Underfitting can not only happen in regression, but also in classification:

Overfitting: The model models the training data too well, thus does not **generalizes** well to unseen data (target set). Generalization refers to how well the concepts learned by a machine learning model apply to specific examples not seen by the model when it was learning.

Pitfalls in Classification

Over- and Underfitting can not only happen in regression, but also in classification:

Overfitting: The model models the training data too well, thus does not **generalizes** well to unseen data (target set). Generalization refers to how well the concepts learned by a machine learning model apply to specific examples not seen by the model when it was learning.

Underfitting: Underfitting refers to a model that can **neither model** the training data **nor generalize** to new data.
An underfit machine learning model is not a suitable model and will be obvious as it will have poor performance on the training data.

Pitfalls in Supervised Machine Learning

Data leakage:
Data leakage (also known as *feature leakage* or *target leakage*) happens when the training data contains information about the label, but similar data will not be available when the model is used for prediction. This leads to overly optimistic performance on the training and validation data, but the model will perform poorly in production on the target set data.

They are usually caused by one of the following: a duplicate label, a proxy for the label, or the label itself. These features will not be available when the model is used for predictions.

Pitfalls in Supervised Machine Learning

Data leakage:

Data leakage (also known as *feature leakage* or *target leakage*) happens when the training data contains information about the label, but similar data will not be available when the model is used for prediction. This leads to overly optimistic performance on the training and validation data, but the model will perform poorly in production on the target set data.

They are usually caused by one of the following: a duplicate label, a proxy for the label, or the label itself. These features will not be available when the model is used for predictions.

examples:

- objects from a data source containing only stars have an object identifier that starts with a number, whereas for those who are not stars it starts with a letter
- a certain waveband from a targeted survey for e.g. only exoplanet host stars and is `NaN` otherwise.

Verification

never apply classification as a black box!

various **verification techniques** can be applied by splitting the modeling set into training and validation set

Verification

never apply classification as a black box!

various **verification techniques** can be applied by splitting the modeling set into training and validation set

For simplicity, we consider here binary classification where each observation is assigned to either class 1 or 0 ($=$ not 1).
In that case, there are the following outcomes (if you want identify class 1):

- True Positive = correctly identified (class 1 identified as class 1)
- True Negative = correctly rejected (class 0 rejected as class 0)
- False Positive = incorrectly identified (class 0 identified as class 1)
- False Negative = incorrectly rejected (class 1 rejected as class 0)

Verification

Based on these, we define the following pairs of terms of **completeness** and **purity**:

$$\text{completeness} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$
$$\text{contamination} = \frac{\text{false positives}}{\text{true positives} + \text{false positives}} = \text{false discovery rate}$$

Instead of contamination, often also efficiency (also called purity) is used:

$$\text{efficiency} = (1 - \text{contamination})$$

Verification

A different way to do this is the **true positive** and **false positive** rate:

$$\text{true positive rate} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$
$$\text{false positive rate} = \frac{\text{false positives}}{\text{true negatives} + \text{false positives}}$$

Similarly

$$\text{efficiency} = 1 - \text{contamination} = \text{precision}.$$

Verification

To illustrate the differences between these measures, let's look at the following **example:**

We have a modeling set containing 100,000 stars and 1000 quasars. If you correctly identify 900 quasars and mistake 1000 stars for quasars, we have:

Verification

To illustrate the differences between these measures, let's look at the following **example:**

We have a modeling set containing 100,000 stars and 1000 quasars. If you correctly identify 900 quasars and mistake 1000 stars for quasars, we have:

- TP $= 900$ (true positive)
- FN $= 100$ (false negative)
- TN $= 99{,}000$ (true negative)
- FP $= 1000$ (false positive)

Which gives

$$\text{true positive rate} = \frac{900}{900+100} = 0.9 = \text{completeness}$$
$$\text{false positive rate} = \frac{1000}{99000+1000} = 0.01$$

Verification

To illustrate the differences between these measures, let's look at the following **example:**

We have a modeling set containing 100,000 stars and 1000 quasars. If you correctly identify 900 quasars and mistake 1000 stars for quasars, we have:

- TP $= 900$ (true positive)
- FN $= 100$ (false negative)
- TN $= 99{,}000$ (true negative)
- FP $= 1000$ (false positive)

Which gives

$$\text{true positive rate} = \frac{900}{900 + 100} = 0.9 = \text{completeness}$$

$$\text{false positive rate} = \frac{1000}{99000 + 1000} = 0.01$$

question:
What do you think about these results?

Verification

To illustrate the differences between these measures, let's look at the following **example:**

We have a modeling set containing 100,000 stars and 1000 quasars. If you correctly identify 900 quasars and mistake 1000 stars for quasars, we have:

- TP $=$ 900 (true positive)
- FN $=$ 100 (false negative)
- TN $=$ 99,000 (true negative)
- FP $=$ 1000 (false positive)

Which gives

$$\text{true positive rate} = \frac{900}{900 + 100} = 0.9 = \text{completeness}$$
$$\text{false positive rate} = \frac{1000}{99000 + 1000} = 0.01$$

answer:
Despite the FPR doesn't look bad, there are a lot of stars, so the contamination rate isn't good: $\text{contamination} = \frac{1000}{900+1000} = 0.53$

Verification

To illustrate the differences between these measures, let's look at the following **example:**

We have a modeling set containing 100,000 stars and 1000 quasars. If you correctly identify 900 quasars and mistake 1000 stars for quasars, we have:

- TP $= 900$ (true positive)
- FN $= 100$ (false negative)
- TN $= 99{,}000$ (true negative)
- FP $= 1000$ (false positive)

Which gives

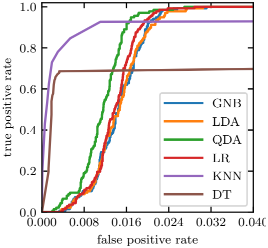
$$\text{true positive rate} = \frac{900}{900 + 100} = 0.9 = \text{completeness}$$
$$\text{false positive rate} = \frac{1000}{99000 + 1000} = 0.01$$

however:
The classifier might be sufficient as one step in a classification pipeline.

Classifier Performance

tradeoff: contamination versus completeness

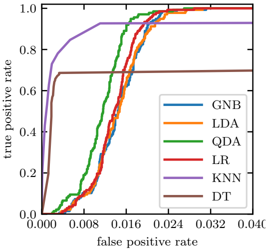
quantify this with a **Receiver Operating Characteristic (ROC)** curve which plots the true-positive vs. the false-positive rate



Classifier Performance

tradeoff: contamination versus completeness

quantify this with a **Receiver Operating Characteristic (ROC)** curve which plots the true-positive vs. the false-positive rate

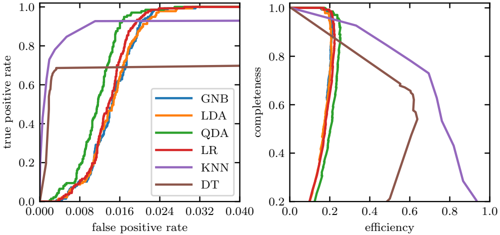


One concern about ROC curves is that they are sensitive to the relative sample sizes: if there are many more background events than source events, small false positive results can dominate a signal.

For these cases we can plot completeness versus efficiency.

Classifier Performance

Here is a comparison of the two types of plots:



Here we see that to get higher completeness, you could actually suffer significantly in terms of efficiency, but your FPR might not go up that much if there are lots of true negatives.

Note that the desired completeness and efficiency is chosen by selecting a decision boundary. The curves show what these possible choices are. Generally, one wants to chose a decision boundary that maximizes the area under the ROC (or completeness versus efficiency) curve.

Classification Algorithms

With some assessment criteria defined, we can look at classification algorithms itself.

Classification Algorithms

Within classification algorithms, we can further differentiate into **Generative** vs. **Discriminative Classification**:

Which category is most likely to generate the observed result? using **density estimation** for classification ⇒ a full model of the density for each class is necessary

not caring about the full distribution, just defining boundaries ⇒ classification that finds the **decision boundary** that separates classes

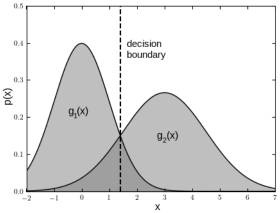
Classification Algorithms

Within classification algorithms, we can further differentiate into **Generative** vs. **Discriminative Classification**:

Which category is most likely to generate the observed result? using **density estimation** for classification ⇒ a full model of the density for each class is necessary

not caring about the full distribution, just defining boundaries ⇒ classification that finds the **decision boundary** that separates classes

example:



With these distributions, to classify a new object with $x = 1$, it would suffice to know that either

1. model 1 is a better fit than model 2 (generative classification), or
2. that the decision boundary is at $x = 1.4$ (discriminative classification).

Generative Classification

We can use **Bayes' theorem** to relate the labels to the features in an $N \times D$ data set X. The jth feature of the ith point is x_i^j and there are k classes giving discrete labels y_k.

We have

$$p(y_k|x_i) = \frac{p(x_i|y_k)p(y_k)}{\sum_i p(x_i|y_k)p(y_k)},$$

where x_i is assumed to be a vector with j components.

$p(y = y_k)$ is the probability of any point having class k (equivalent to the prior probability of the class k).

In generative classifiers we model class-conditional densities $p(x|y = y_k)$.

Generative Classification

The Discriminant Function

We can relate classification to density estimation and regression.
$\hat{y} = f(y|x)$ represents the best guess of y given x. So classification is just regression with discrete y values, e.g., $y = \{0, 1\}$.
In classification we refer to $f(y|x)$ as the **discriminant function**.

Generative Classification

The Discriminant Function

We can relate classification to density estimation and regression.
$\hat{y} = f(y|x)$ represents the best guess of y given x. So classification is just regression with discrete y values, e.g., $y = \{0, 1\}$.
In classification we refer to $f(y|x)$ as the **discriminant function**.

For a simple 2-class example, where $y = \{0, 1\}$:

$$\begin{aligned} g(x) = f(y|x) &= \int y\, p(y|x)\, dy \\ &= 1 \cdot p(y=1|x) + 0 \cdot p(y=0|x) = p(y=1|x). \end{aligned}$$

and then using Bayes' rule:

$$g(x) = \frac{p(x|y=1)\, p(y=1)}{p(x|y=1)\, p(y=1) + p(x|y=0)\, p(y=0)}$$

The first equation is just the expectation value of y.

Bayes Classifier

If the discriminant function gives a binary prediction, we call it a Bayes classifier, formulated as

$$\begin{aligned} \hat{y} &= \begin{cases} 1 & \text{if } g(x) > 1/2, \\ 0 & \text{otherwise,} \end{cases} \\ &= \begin{cases} 1 & \text{if } p(y=1|x) > p(y=0|x), \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

This can be generalized to any number of classes, k, and not just two.

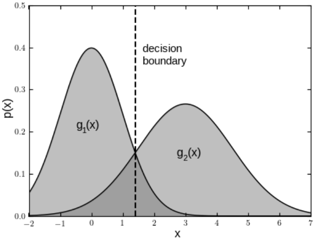
Decision Boundary

A decision boundary is set of x values at which each class is equally likely:

$$p(x|y=1)p(y=1) \;=\; p(x|y=0)p(y=0)$$

$$g_1(x) = g_2(x) \text{ or } g(x) = 1/2$$

Below is an example of a decision boundary in 1D, where each class is equally likely so we can just look at $p(x)$.



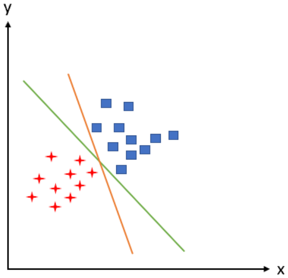
Discriminative Classification

Discriminative classification consists of methods that seek only to determine the **decision boundary in feature space**.

Discriminative Classification

Discriminative classification consists of methods that seek only to determine the **decision boundary in feature space**.

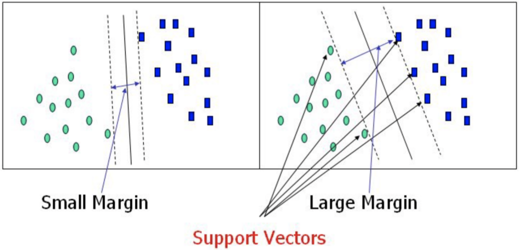
example:
We have the data as shown in the plot below. We could separate them by a line.
But: There are clearly lots of different lines that that would work. How do you do this optimally so it also works for the future target set? And what if the blobs are not perfectly well separated?



Discriminative Classification: Support Vector Machines

Support Vector Machines (SVM) define a **hyperplane** in $N-1$ dimensions that maximizes the distance (the *margin*) of the closest point from each class. The points that touch the margin (or that are on the wrong side) are the **support vectors**.

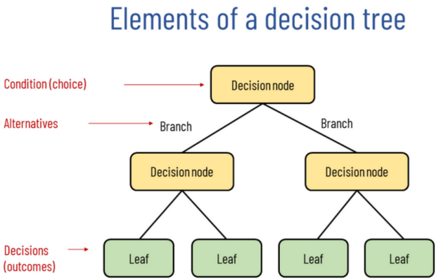
There are lots of potential decision boundaries, but we want the one that maximize the distance of the support vectors from the decision hyperplane.



Discriminative Classification: Decision Trees

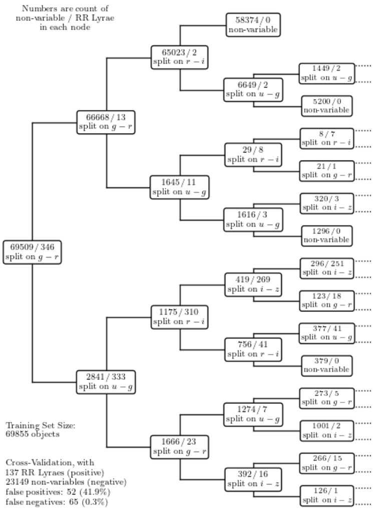
A **decision tree** is a hierarchical application of decision boundaries:

- the top node contains the entire data set
- define some criteria to split the sample into 2 groups (not necessarily equal)
- splitting repeats, recursively, until a predefined stopping criteria is reached



Discriminative Classification: Decision Trees

example:



The terminal nodes (leaf nodes) record the fraction of points that have one classification or the other in the training set.

The fraction of points from the training set classified as one belonging to one class or the other (in the leaf node) defines the class associated with that leaf node.

The binary splitting makes this extremely efficient. The trick is to ask the right questions.

Discriminative Classification: Decision Trees

One way to define **Splitting Criteria** is to use the information content (or *entropy*), $E(x)$, of the data

$$E(x) = -\sum_i p_i(x) \ln(p_i(x)),$$

where i is the class and $p_i(x)$ is the probability of that class given the training data. We can define the **information gain** as the reduction in entropy due to the partitioning of the data (i.e. by partitioning the data you have reduced the disorder). For a binary split with $i = 0$ representing those points below the split threshold and $i = 1$ as those points above the split threshold, the information gain $IG(x)$ is

$$IG(x|x_i) = E(x) - \sum_{i=0}^{1} \frac{N_i}{N} E(x_i),$$

where N_i is the number of points, x_i, in the i-th class, and $E(x_i)$ is the entropy of that class. We are assessing the information gain as the difference between the entropy of the parent node and the sum of the entropies of the child nodes.

Discriminative Classification: Decision Trees

The typical process for finding the optimal decision boundary is to perform trial splits along each feature one at a time, within which the value of the feature to split at is also trialed. The feature that allows for the maximum information gain is the one that is split at this level.
Another commonly used "loss function" (especially for categorical classification) is the Gini coefficient:

$$G = \sum_{i}^{k} p_i(1 - p_i).$$

It essentially estimates the probability of incorrect classification by choosing both a point and (separately) a class randomly from the data.

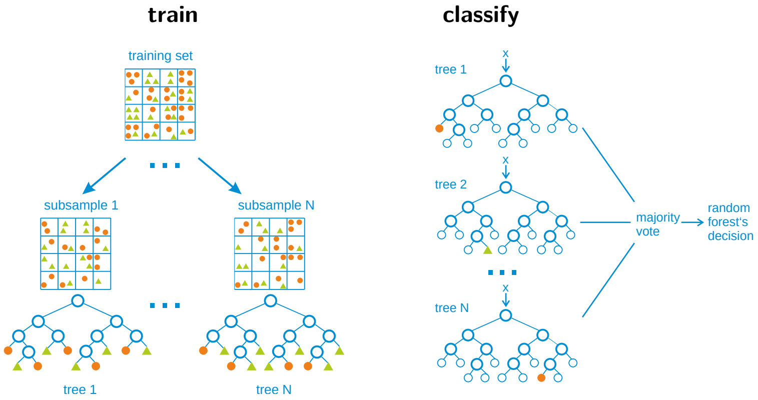
Ensemble Learning

Ensemble learning is the process of using multiple models, trained over the same data, averaging the results of each model ultimately finding a more powerful prediction/classification result.

Ensemble Learning

The Random Forest algorithm combines ensemble learning methods with the decision tree framework to create multiple randomly drawn decision trees from the data, averaging the results to output a result that often times leads to strong predictions/classifications.

The result is a **more robust classifier**.



Ensemble Learning: Random Forests

Random forests generate decision trees from bootstrap samples (drawing from the observed data set with replacement). This helps to overcome some of the limitations of decision trees.

In Random Forests, the splitting features on which to generate the tree are selected at random from the full set of features in the data.

The number of features selected per split level is typically the square root of the total number of features, $\sqrt{D}$.

The final classification from the random forest is based on the averaging of the classifications of each of the individual decision trees.

As before: cross-validation can be used to determine the optimal depth. Generally the number of trees, n, that are chosen is the number at which the cross-validation error plateaus.

Chosing the Right Classifier

no single model can be known in advance to be the best classifier

There are many factors, such as the size and structure of your dataset.

Advice: try many different (appropriate) algorithms for your problem, evaluate the performance for each and select the winner

Of course, the algorithms you try must be appropriate for your problem, which is where picking the right machine learning task comes in.

Chosing the Right Classifier

In general, the level of accuracy increases for parametric models as:

- naive Bayes,
- linear discriminant analysis (LDA)
- logistic regression,
- linear support vector machines,
- quadratic discriminant analysis (QDA)
- linear ensembles of linear models.

For non-parametric models accuracy increases as:

- decision trees
- K-nearest-neighbor
- neural networks
- kernel discriminant analysis
- kernelized support vector machines
- random forests
- boosting

See also Ivezic, Table 9.1.

Why not classify with Linear Regression?

Suppose that we are trying to predict the medical condition of a patient in the emergency room on the basis of symptoms. In this simplified example, there are three possible diagnoses: stroke, drug overdose, and epileptic seizure.

We could consider encoding these values as a **quantitative response variable**, Y, as follows:

$$Y = \begin{cases} 1, & \text{if stroke} \\ 2, & \text{if drug overdose} \\ 3, & \text{if epileptic seizure} \end{cases}$$

We then could use this to fit a linear regression model to predict Y on the basis of a set of predictors $X_1, ...m, X_p$.

Logistic Regression

Unfortunately, this coding implies an **ordering** on the outcomes, and insisting that the difference between two consecutive ones is always the same.

Each possible codings would produce **fundamentally different linear models** that would ultimately lead to different sets of predictions on test observations.
To summarize, there are at least two reasons **not to perform classification using the (linear) regression method**:

- a regression method cannot accommodate a qualitative response with more than two classes
- a regression method will not provide meaningful estimates of $P(Y|X)$, even with just two classes.

Thus, a classification method that is truly suited for qualitative response values must be used.
We can derive such a classification method from regression.

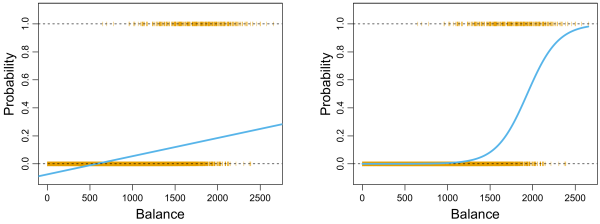
Logistic Regression

How to model the relationship between $p(X) = Pr(Y = 1|X)$ and X?

We must use a function that gives outputs between 0 and 1 for all values of X. In logistic regression, we use the **logistic function**.

$$p(X) = \frac{\exp(\beta_0 + \beta_1 X)}{1 + \exp(\beta_0 + \beta_1 X)}$$

This function produces an S-shaped curve between Y values 0 and 1. With that it captures the range of probabilities better than the linear regression model (left-hand plot), for which some estimated probabilities are negative.



Source: Fig. 4.2 from `https://www.statlearning.com/`

The Odds

From

$$p(X) = \frac{\exp(\beta_0 + \beta_1 X)}{1 + \exp(\beta_0 + \beta_1 X)}$$

we get

$$\frac{p(X)}{1p(X)} = \exp(\beta_0 + \beta_1 X)$$

This quantitiy is called the **odds**, and can take on any value between 0 and ∞. Values of the odds close to 0 and ∞ indicate very low and very high probabilities.

For example, on average 1 in 5 people with an odds of $1/4$ will default, since p(X) $=$ 0.2 implies an odds of $1/4$.
By taking the logarithm of both sides, we arrive at

$$\log \frac{p(X)}{1p(X)} = \beta_0 + \beta_1 X$$

The left-hand side is called the log odds or logit. We see that the logistic regression model has a logit that is linear in X.

Maximum Likelihood

This model is then fit using the **maximum likelihood** method.

Although we could use (non-linear) least squares to fit the logistic regression model, the more general method of maximum likelihood is preferred, since it has better statistical properties.

The **basic intuition** behind using maximum likelihood:
We try to find $\hat{\beta}_0$ and $\hat{\beta}_1$ such that plugging these estimates into the model for $p(X)$ gives a number close to one for all individuals who have this status, and a number close to zero for all individuals who do not.

This intuition can be formalized using a mathematical equation called a likelihood function:

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

The estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ are chosen to maximize this likelihood function.

Maximum Likelihood

In general, we do not need to concern ourselves with the details of the maximum likelihood fitting procedure.

It is typically implemented in statistics packages, like such used for Python, or specific statistics software.

Many aspects of the logistic regression output are similar to the linear regression output we saw before.

For example, we can measure the accuracy of the coefficient estimates by computing their standard errors.

Multiple Logistic Regression

We now consider the problem of predicting a binary response using multiple predictors. By analogy with the extension from simple to multiple linear regression, we can generalize as follows:

$$\log\left(\frac{p(x)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$$

where $X = (X_1, \ldots, X_p)$ are p predictors.

We then rewrite our logistic regression equation as follows:

$$p(X) = \frac{\exp(\beta_0 + \beta_1 X_1 \ldots + \beta_p X_p)}{1 + \exp(\beta_0 + \beta_1 X_1 \ldots + \beta_p X_p)}$$

Again, the maximum likelihood method is used to estimate the coefficients.

Summary

Today we have seen a general overview about **Classification**.

Next time we will learn about **Support Vector Machines**, a type of classifier.