Introduction to Machine Learning with Scikit-Learn

District Data Labs

These slides are based on:
https://www.slideshare.net/BenjaminBengfort/introduction-to-machine-learning-with-scikitlearn
by Benjamin Bengfort

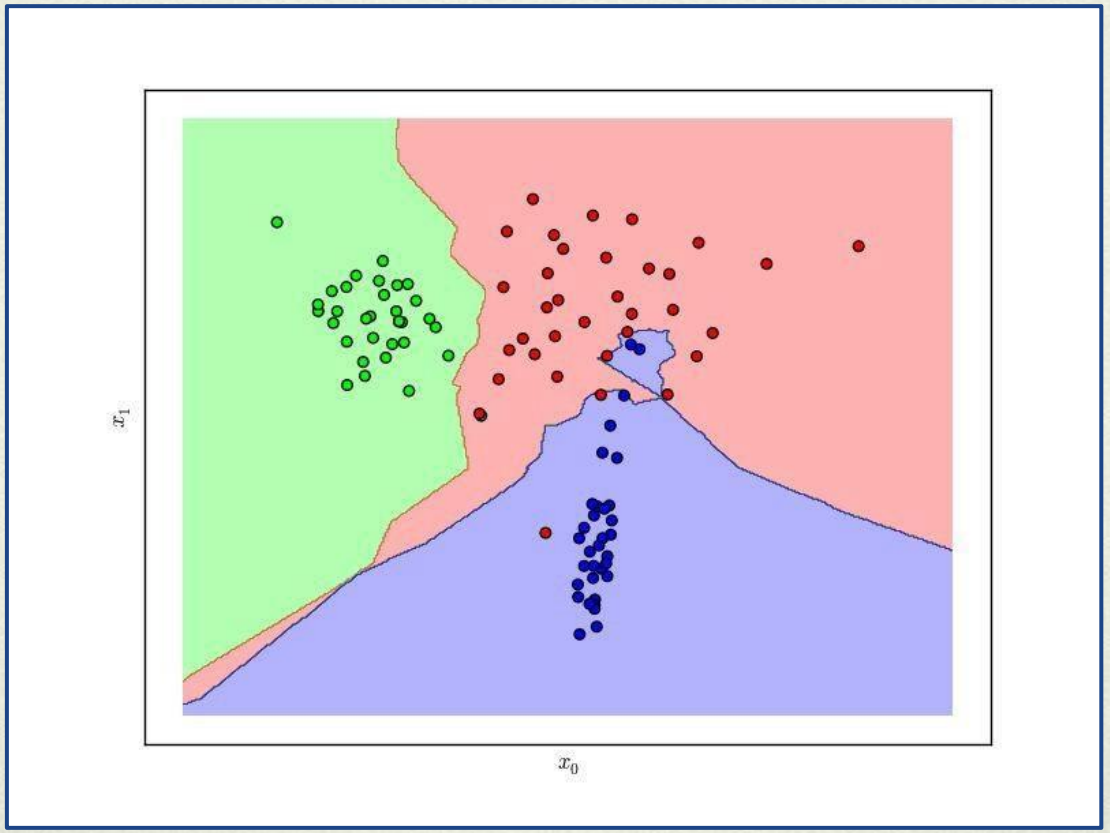
Types of Algorithms by Output

Input *training* data to *fit* a model which is then used to *predict* incoming inputs into ...

Type of Output	Algorithm Category
Output is one or more discrete classes	**Classification (supervised)**
Output is continuous	**Regression (supervised)**
Output is membership in a similar group	**Clustering (unsupervised)**
Output is the distribution of inputs	Density Estimation
Output is simplified from higher dimensions	Dimensionality Reduction

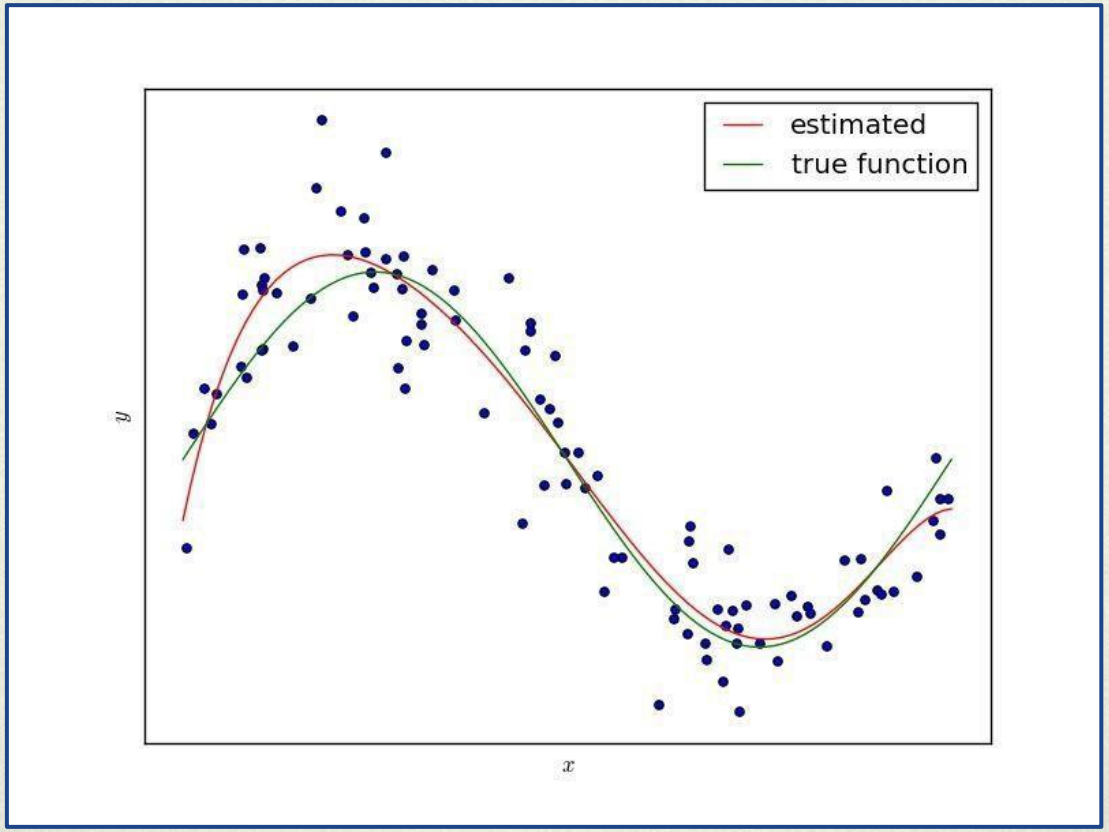
Other: semi-supervised (lot of outputs unknown), reinforcement learning (decision -> feedback principle)

Classification



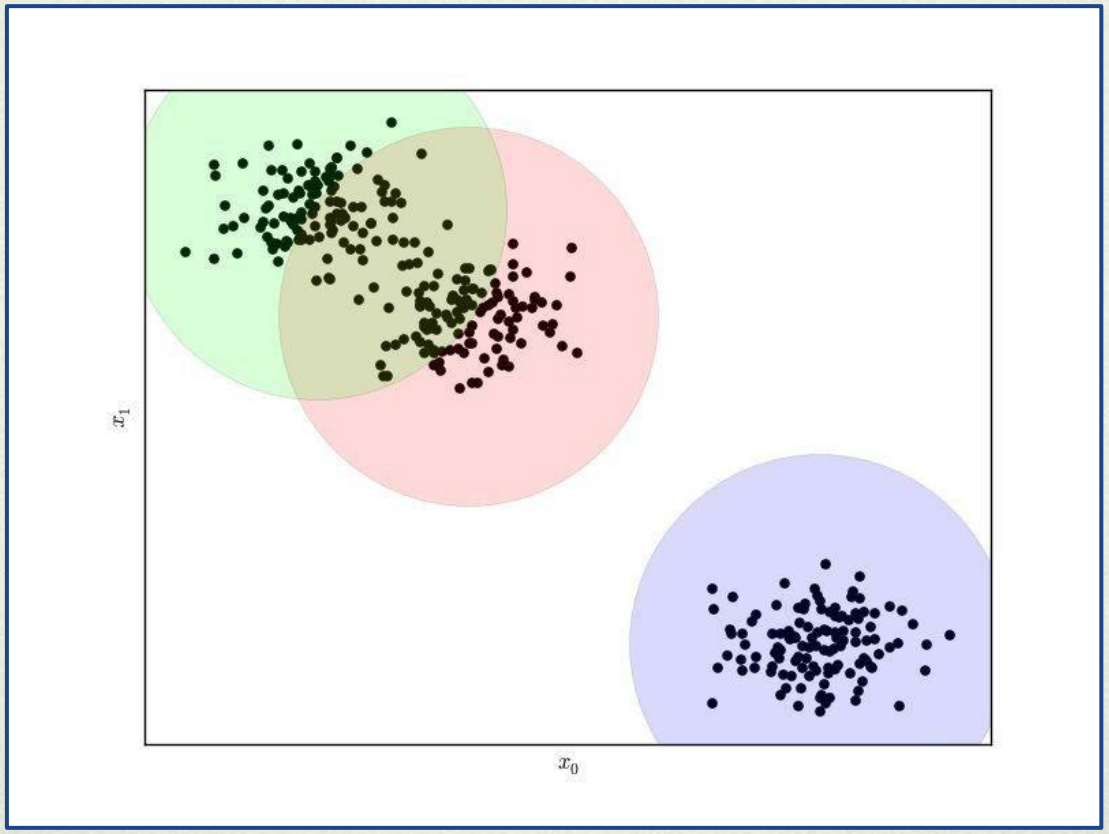
Given labeled input data (with two or more labels), fit a function that can determine for any input, what the label is.

Regression



Given continuous input data fit a function that is able to predict the continuous value of input given other data.

Clustering



Given data, determine a pattern of associated data points or clusters via their similarity or distance from one another.

Dimensions and Features

In order to do machine learning you need a data set containing *instances* (examples) that are composed of *features* from which you compose dimensions.

Instance: a single data point or example composed of fields

Feature: a quantity describing an instance

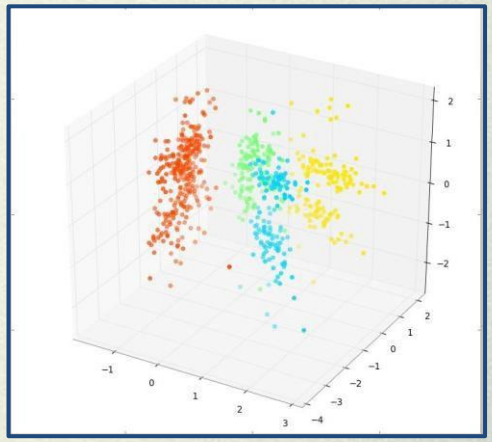
Dimension: one or more attributes that describe a property

```
from sklearn.datasets import load_digits
digits =load_digits()

X=digits.data          # X.shape ==(n_samples, n_features)
y = digits.target      # y.shape == (n_samples,)
```

Feature Space

Feature space refers to the *n*-dimensions where your variables live (not including a target variable or class). The term is used often in ML literature because in ML all variables are features (usually) and feature extraction is the art of creating a space with decision boundaries.



Target

1. $Y \equiv$ Thickness of car tires after some testing period

Variables

1. $X_1 \equiv$ distance travelled in test
2. $X_2 \equiv$ time duration of test
3. $X_3 \equiv$ amount of chemical *C* in tires

The feature space is R^3, or more accurately, the positive quadrant in R^3 as all the *X* variables can only be positive quantities.

http://stats.stackexchange.com/questions/46425/what-is-feature-space

Mappings

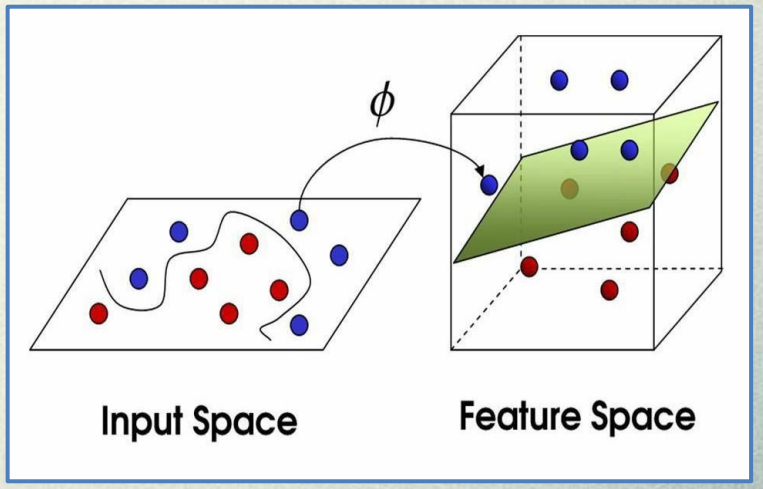
Domain knowledge about tires might suggest that the speed the vehicle was moving at is important, hence we generate another variable, X_4 (this is the feature extraction part):

$X_4 = X_1 * X_2 \equiv$ the speed of the vehicle during testing.

This extends our old feature space into a new one, the positive part of R^4.

A mapping is a function, ϕ, from R^3 to R^4:

$\phi(x_1,x_2,x_3) = (x_1,x_2,x_3,x_1x_2)$



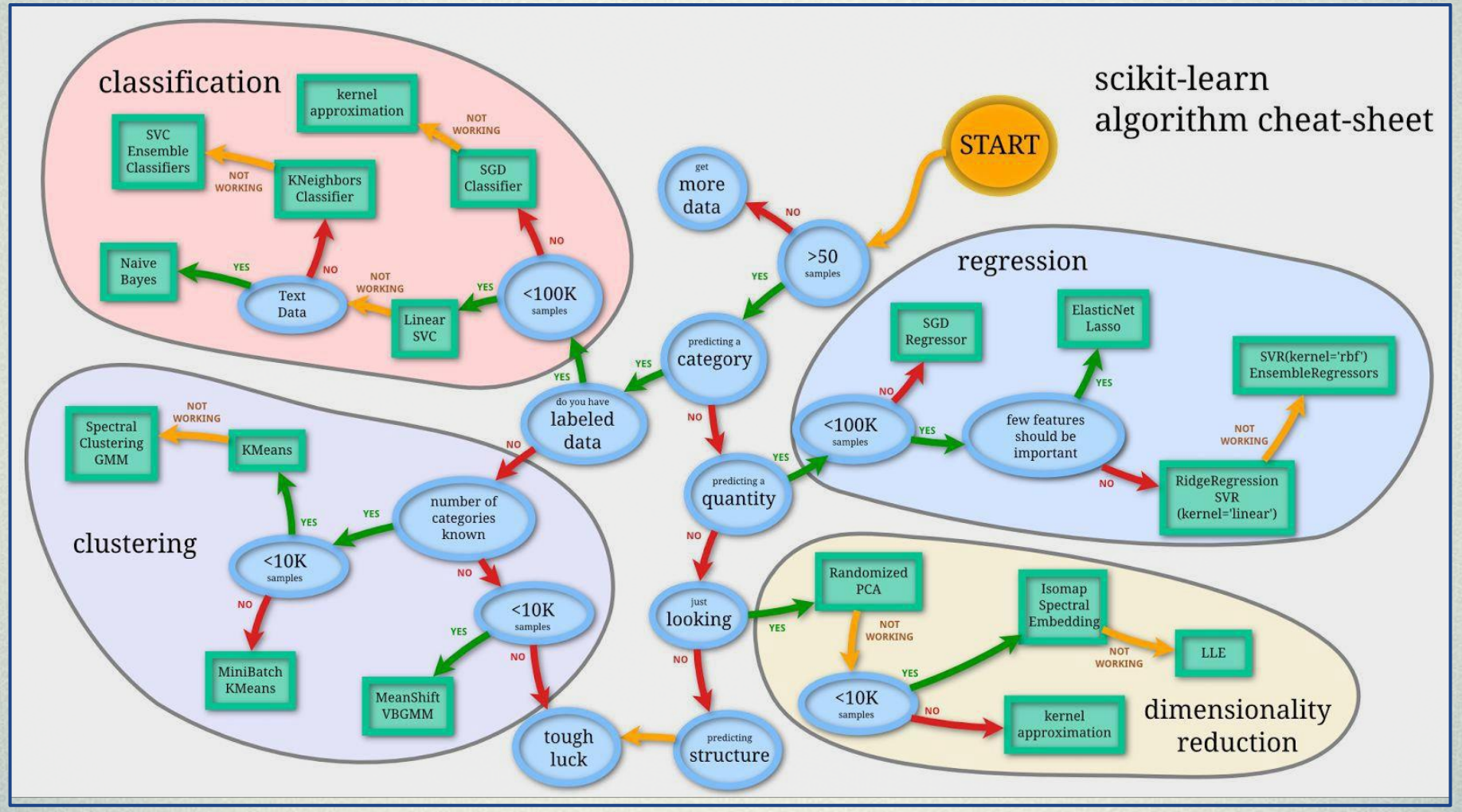
http://stats.stackexchange.com/questions/46425/what-is-feature-space

Your Task

Given a data set of instances of size *N*, create a model that is fit from the data (built) by extracting features and dimensions. Then use that model to predict outcomes …

1. Data Wrangling (normalization, standardization, imputing)
2. Feature Analysis/Extraction
3. Model Selection/Building
4. Model Evaluation
5. Operationalize Model

A Tour of Machine Learning Algorithms



A Guide to Scikit-Learn

```
class Estimator(object):

    def fit(self, X, y=None):
        """Fits estimator to data. """
        # set state of ``self``
        return self

    def predict(self, X):
        """Predict response of ``X``. """
        # compute predictions ``pred``
        return pred
```

The Scikit-Learn Estimator API

Estimators

- `fit(X,y)` sets the state of the estimator.
- Xis usually a 2D numpy array of shape (num_samples, num_features).
- y is a 1D array with shape (n_samples,)
- `predict(X)` returns the class or value
- **`predict_proba()`** returns a 2D array of shape (n_samples, n_classes)

  You often care about ranking of items

```
from sklearn import svm

estimator = svm.SVC(gamma=0.001)
estimator.fit(X, y)
estimator.predict(x)
```

Basic methodology

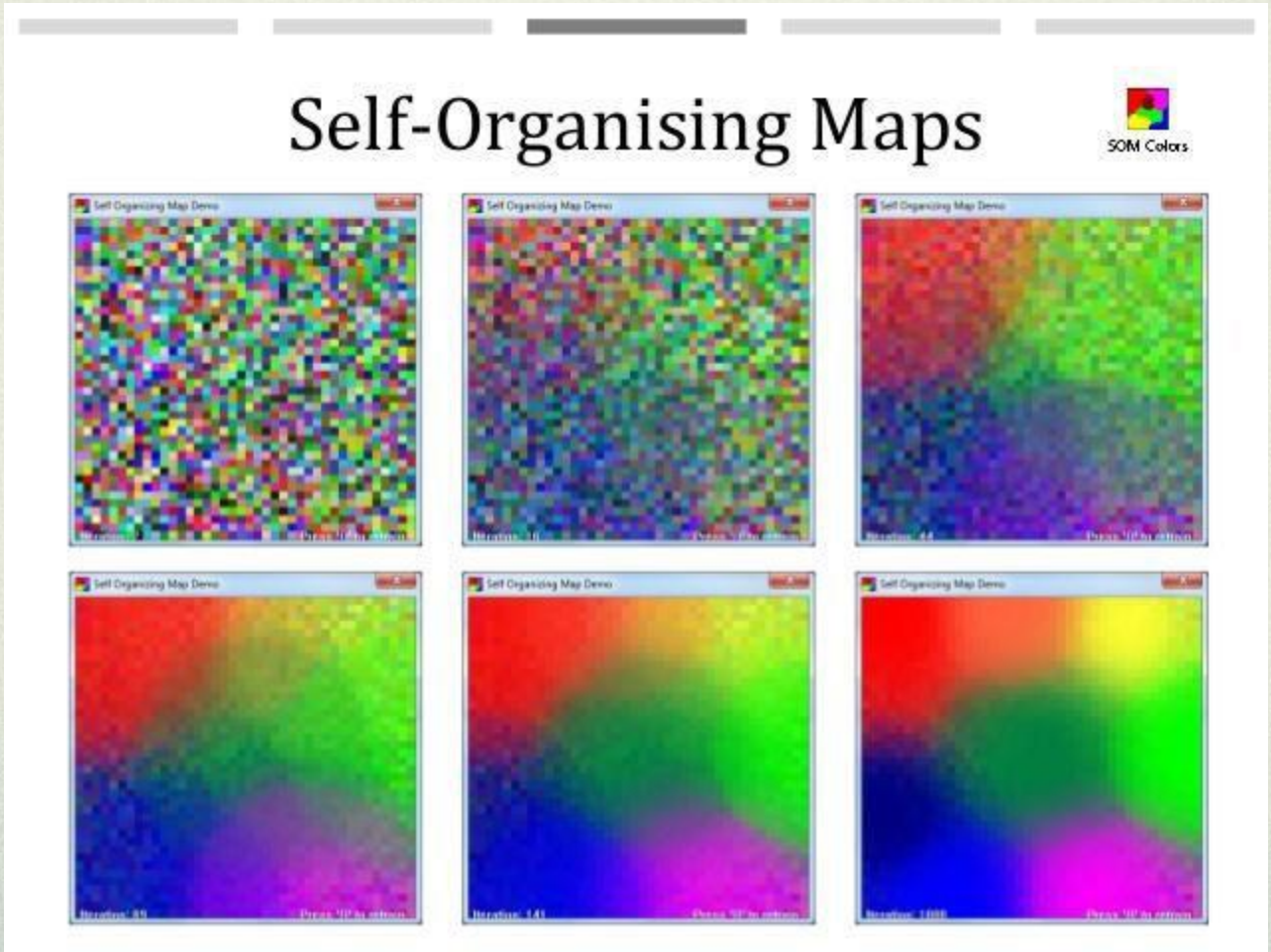
Models: Instance Methods

Compare instances in data set with a similarity measure to find best matches.

- Suffers from curse of dimensionality.
- Focus on feature representation and similarity metrics between instances

- **k-Nearest Neighbors (kNN)**
- **Self-Organizing Maps (SOM)**
- **Learning Vector Quantization (LVQ)**

Self-Organizing Maps



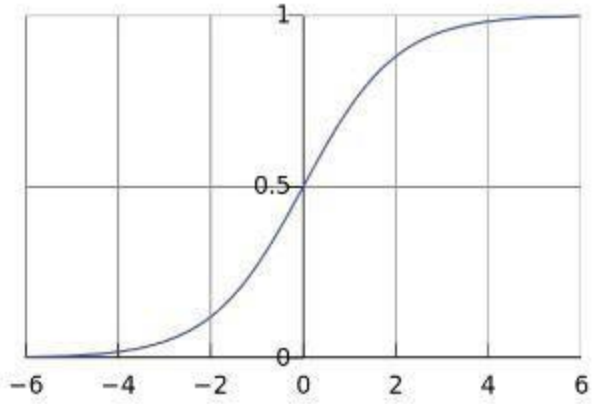
Models: Regression

Model relationship of independent variables, X to dependent variable Y by iteratively optimizing error made in predictions.

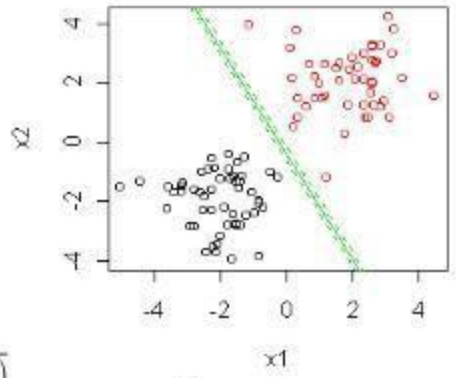
- **Ordinary Least Squares**
- **Logistic Regression**
- **Stepwise Regression**
- **Multivariate Adaptive Regression Splines (MARS)**
- **Locally Estimated Scatterplot Smoothing (LOESS)**
- **Matrix factorization techniques**

Logistic Regression

Binary Logistic Regression



$$d = \frac{ax_1 + bx_2 + cx_0}{\sqrt{a^2 + b^2}} \quad where\ x_0 = 1$$



$$P(y=1|\vec{x},\vec{w}) = \frac{\exp(d)}{1+\exp(d)} = \frac{\exp(\vec{x}\,\vec{w})}{1+\exp(\vec{x}\,\vec{w})}$$

$$P(y=0|\vec{x},\vec{w}) = \frac{1}{1+\exp(\vec{x}\,\vec{w})}$$

$$\log\frac{P(y=1|\vec{x},\vec{w})}{P(y=0|\vec{x},\vec{w})} = \vec{x}\,\vec{w}$$

$$w_0 = \frac{c}{\sqrt{a^2+b^2}} \quad where\ w_0\ is\ called\ as\ intercept$$

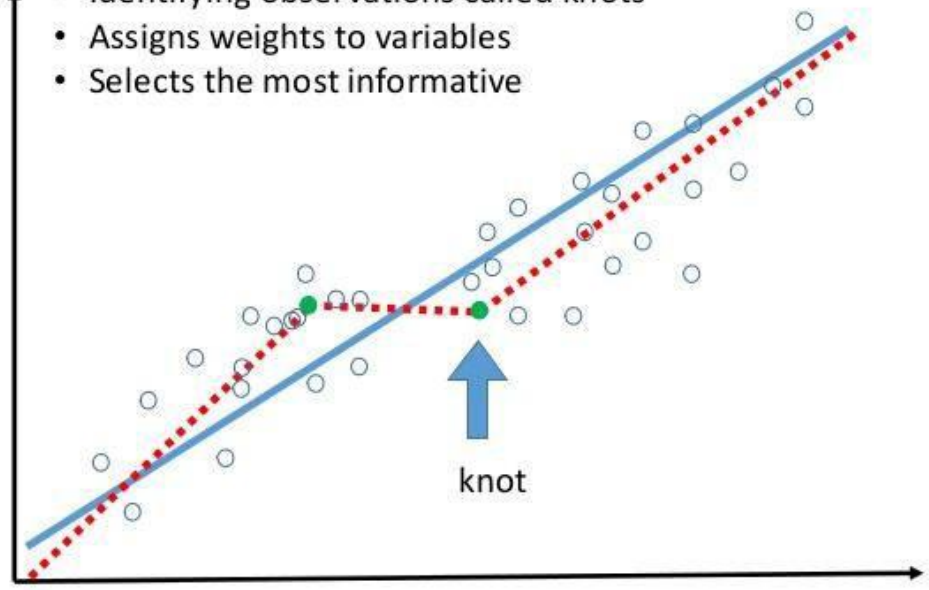
$$w_1 = \frac{a}{\sqrt{a^2+b^2}}$$

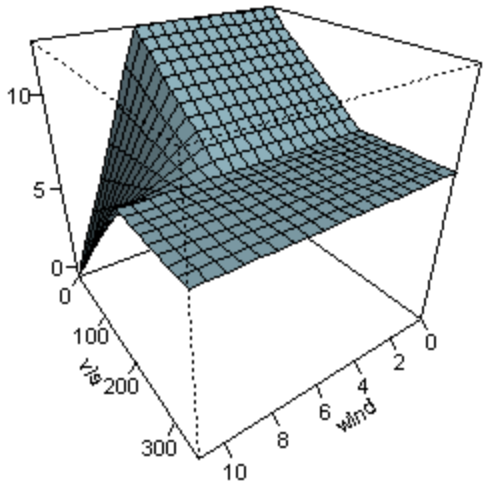
$$w_2 = \frac{b}{\sqrt{a^2+b^2}}$$

Multivariate Adaptive Regression Splines (MARS)

Variable Selection: Multivariate Adaptive Regression Splines (MARS)

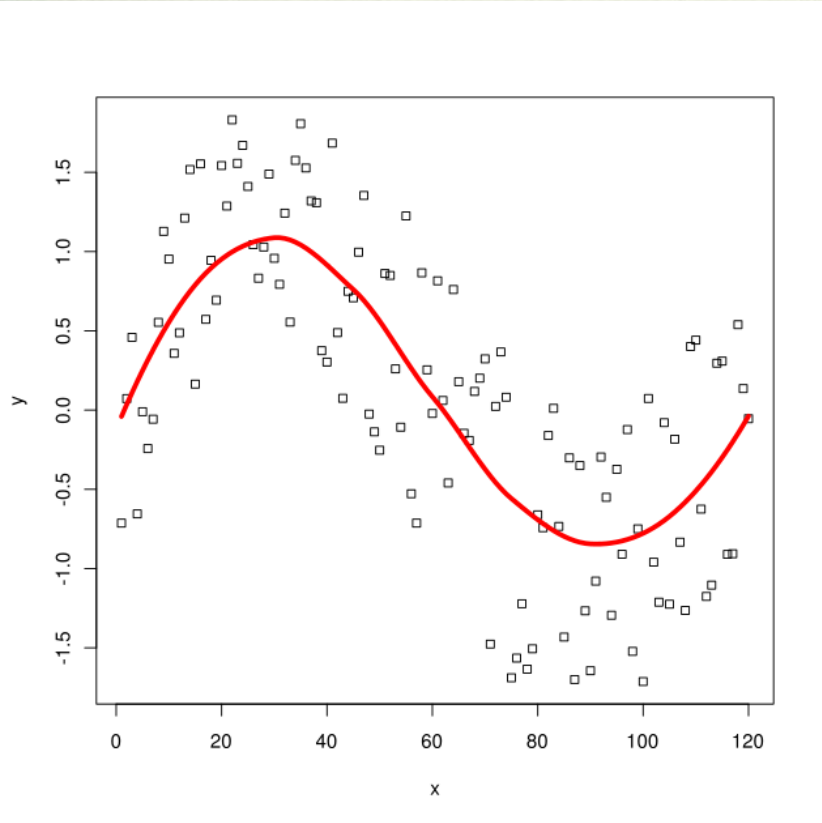
- Breaks dataset into subsamples
- Identifying observations called knots
- Assigns weights to variables
- Selects the most informative





$$\begin{aligned}\text{ozone} = 5.2 \\ &+ 0.93 \max(0, \text{temp} - 58) \\ &- 0.64 \max(0, \text{temp} - 68) \\ &- 0.046 \max(0, 234 - \text{ibt}) \\ &- 0.016 \max(0, \text{wind} - 7) \max(0, 200 - \text{vis})\end{aligned}$$

Locally Estimated Scatterplot Smoothing (LOESS)



- Combine multiple regression models in a k-nearest-neighbor-based meta-model
- Fits a low-degree polynomial to a subset of the data close to the current point
- Requires fairly large, densely sampled data sets in order to produce good models.

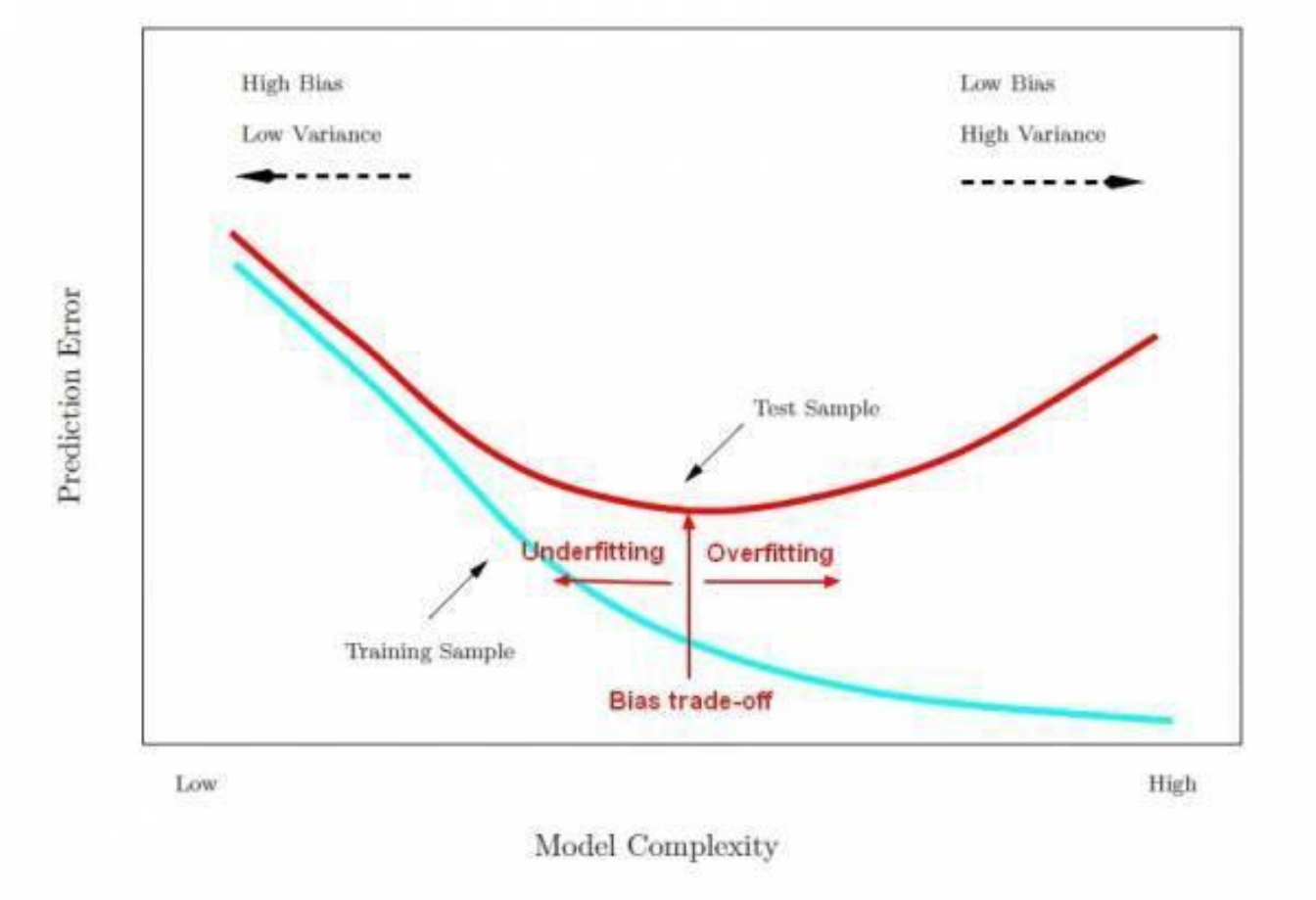
Models: Regularization Methods

Extend another method (usually regression), penalizing complexity (minimize overfit)

- simple, popular, powerful
- better at generalization

- **Ridge Regression**
- **LASSO (Least Absolute Shrinkage & Selection Operator)**
- **Elastic Net**

Models: Regularization Methods



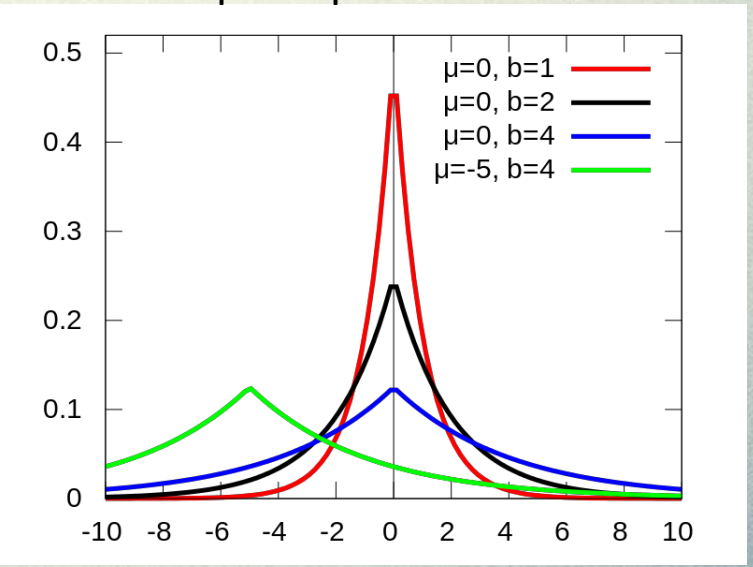
LASSO

- Limits total weight of parameters
- Can be interpreted as a prior distribution on parameters

$$\min_{\beta_0,\beta}\left\{\frac{1}{N}\sum_{i=1}^{N}(y_i-\beta_0-x_i^T\beta)^2\right\} \text{ subject to } \sum_{j=1}^{p}|\beta_j|\le t.$$

- Ridge regression: quadratic penalty
- Elastic Net combines both

Laplace prior distributions

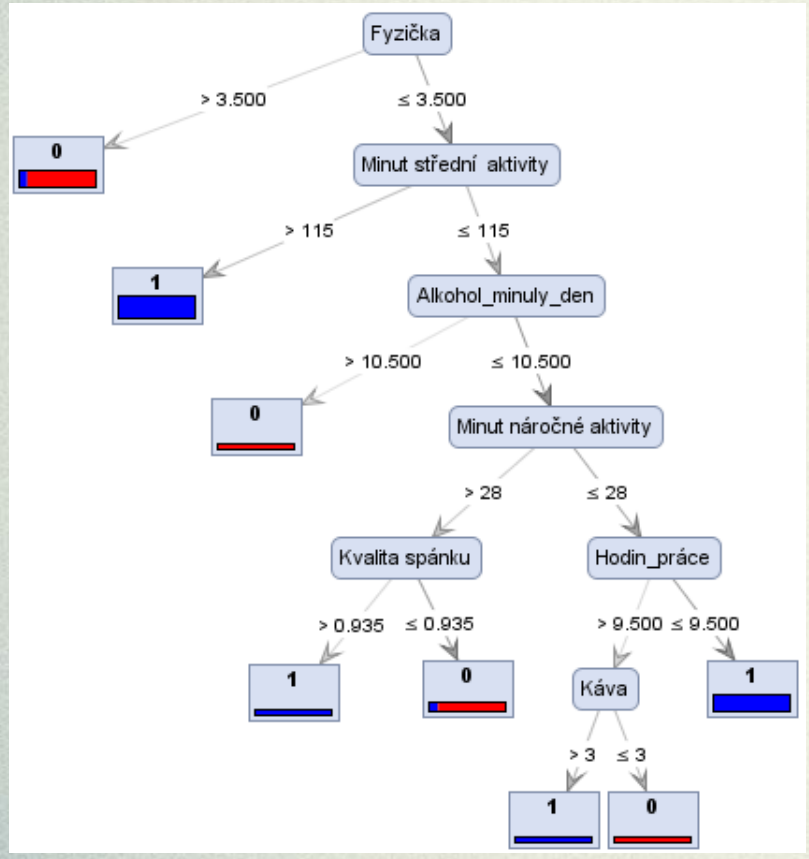


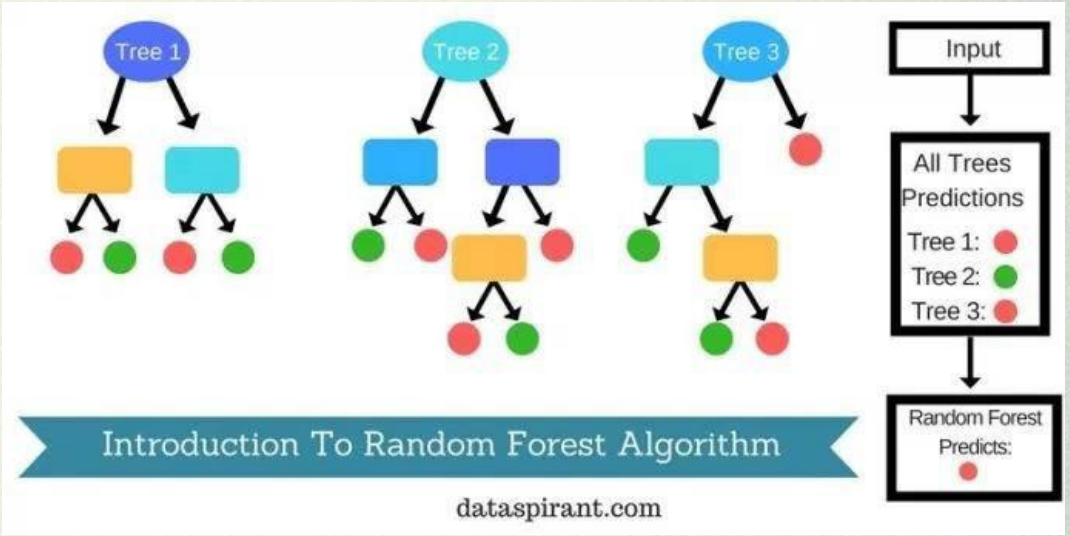
Models: Decision Trees

Model of decisions based on data attributes. Predictions are made by following forks in a tree structure until a decision is made. Used for classification & regression.

- **Classification and Regression Tree (CART)**
- **Decision Stump**
- **Random Forest**
- **Multivariate Adaptive Regression Splines (MARS)**
- **Gradient Boosting Machines (GBM)**

Models: Decision Trees





http://www.saedsayad.com/decision_tree.htm

Models: Bayesian

Explicitly apply Bayes' Theorem for classification and regression tasks. Usually by fitting a probability function constructed via the chain rule and a naive simplification of Bayes.

- **Naive Bayes**
- **Averaged One-Dependence Estimators (AODE)**
- **Bayesian Belief Network (BBN)**

Naive Bayes

- Text retrieval 1960s
- Independence of feature values (given class)
- Bayesian theorem $p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$ $\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$

Probability distribution:

$$\begin{aligned} p(C_k \mid x_1, \ldots, x_n) &\propto p(C_k, x_1, \ldots, x_n) \\ &\propto p(C_k)\, p(x_1 \mid C_k)\, p(x_2 \mid C_k)\, p(x_3 \mid C_k) \cdots \\ &\propto p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)\,. \end{aligned}$$

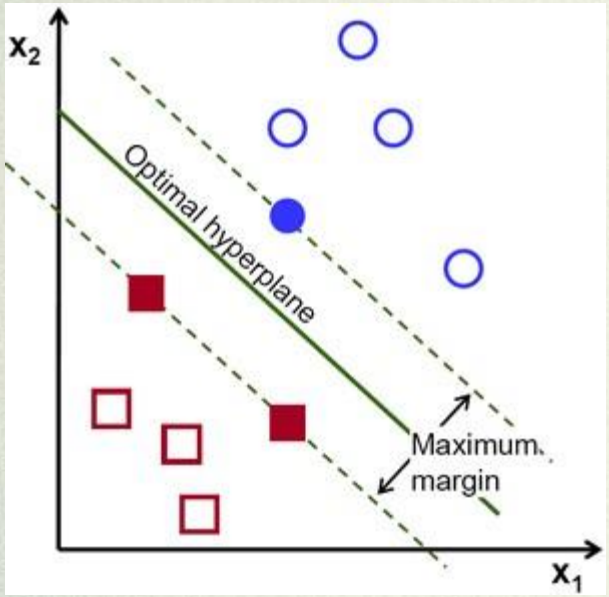
- Voting for max. posterior probability (MAP)

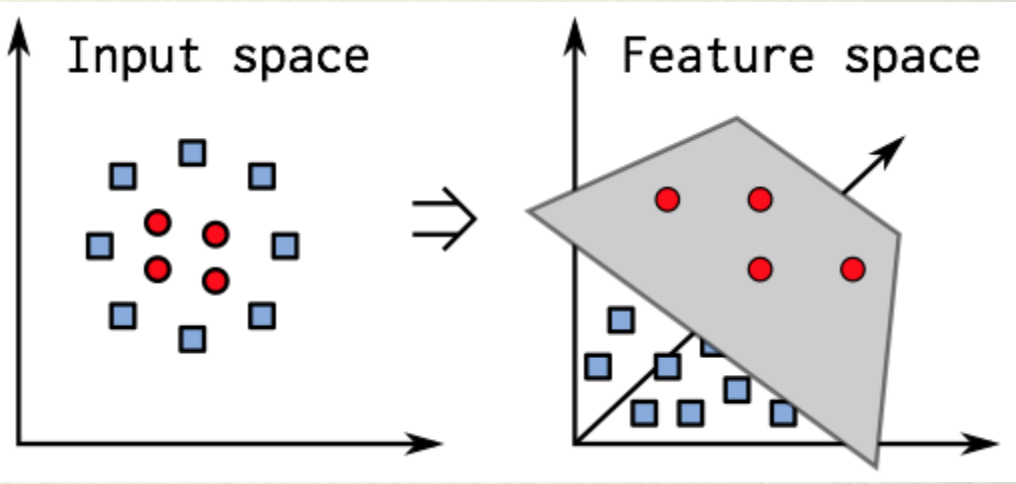
Models: Kernel Methods

Map input data into higher dimensional vector space where the problem is easier to model. Named after the “kernel trick” which computes the inner product of images of pairs of data.

- **Support Vector Machines (SVM)**
- **Radial Basis Function (RBF)**
- **Linear Discriminant Analysis (LDA)**

SVM


x_2
Optimal hyperplane
Maximum
margin
x_1


Input space
Feature space

SVM

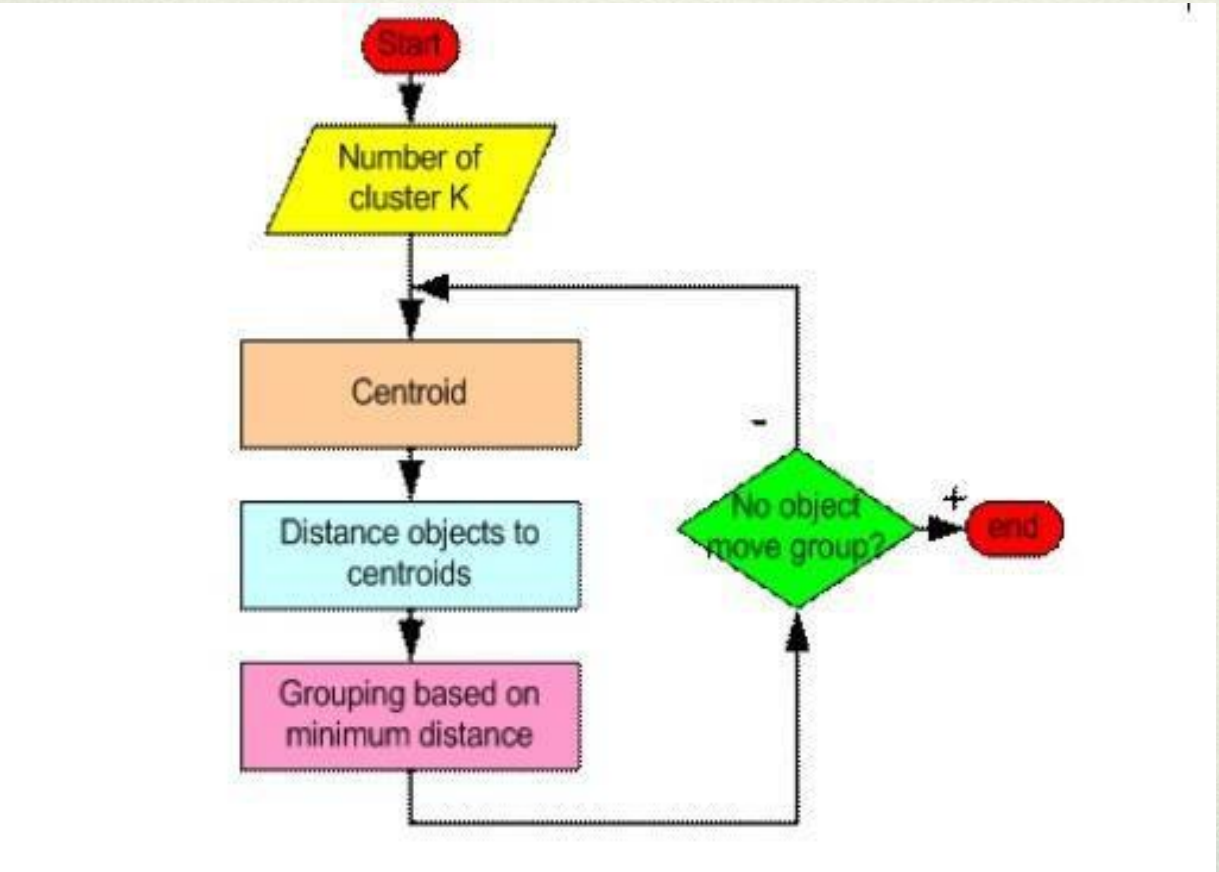
- Common kernel functions for SVM
  - linear $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 \cdot \mathbf{x}_2$
  - polynomial $k(\mathbf{x}_1, \mathbf{x}_2) = (\gamma\, \mathbf{x}_1 \cdot \mathbf{x}_2 + c)^d$
  - Gaussian or radial basis $k(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\gamma \left\|\mathbf{x}_1 - \mathbf{x}_2\right\|^2\right)$
  - sigmoid $k(\mathbf{x}_1, \mathbf{x}_2) = \tanh(\gamma\, \mathbf{x}_1 \cdot \mathbf{x}_2 + c)$

Models: Clustering Methods

Organize data into into groups whose members share maximum similarity (defined usually by a distance metric). Two main approaches: centroids and hierarchical clustering.

- **k-Means**
- **Affinity Propegation**
- **OPTICS (Ordering Points to Identify Cluster Structure)**
- **Agglomerative Clustering**

K-means clustering



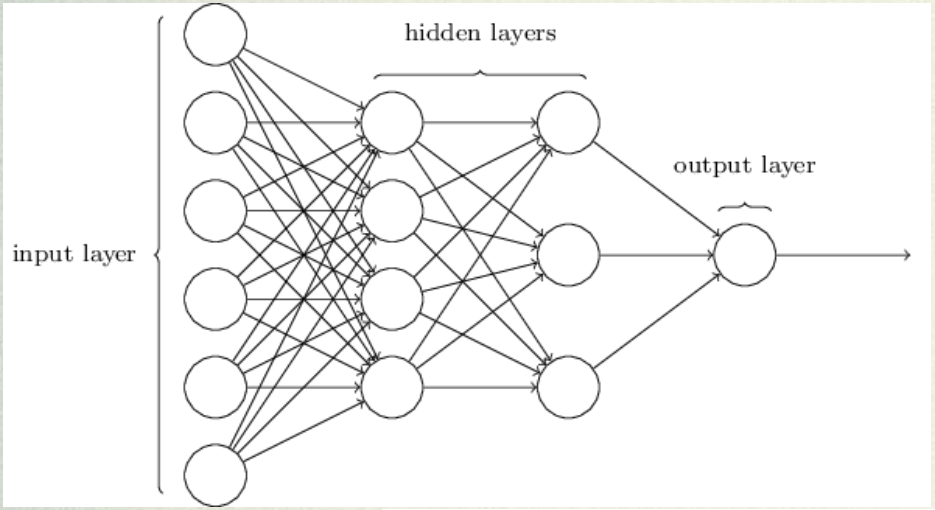
Models: Artificial Neural Networks

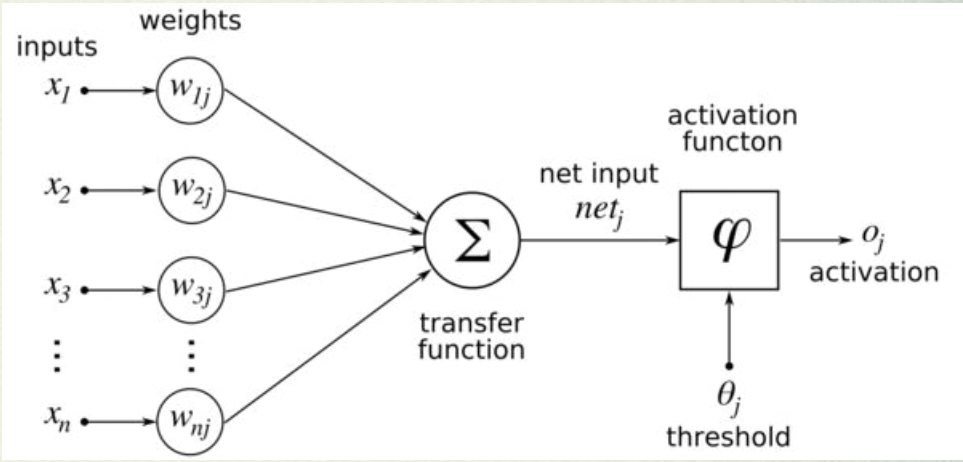
Inspired by biological neural networks, ANNs are nonlinear function approximators that estimate functions with a large number of inputs.

- System of interconnected neurons that activate
- Deep learning extends simple networks recursively

- **Restricted Boltzmann Machine (RBM)**
- **Convolutional Neural Networks (CNN)**
- **Recurrent Neural Networks (RNN)**
- **Word2Vec models**

Models: Artificial Neural Networks

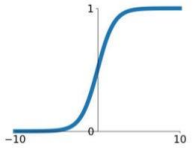




Activation Functions

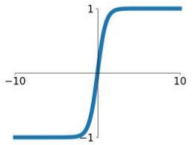
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



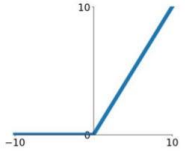
tanh

$\tanh(x)$



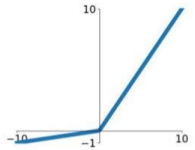
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

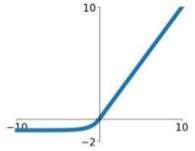


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Models: Ensembles

Models composed of multiple weak models that are trained independently and whose outputs are combined to make an overall prediction.

- **Boosting**
- **Bootstrapped Aggregation (Bagging)**
- **AdaBoost**
- **Stacked Generalization (blending)**
- **Gradient Boosting Machines (GBM)**
- **Random Forest**

AdaBoost

Input :

- A training set $S = ((\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m))$.

Initialization :

- Maximum number of iterations T;
- initialize the weight distribution $\forall i \in \{1, \ldots, m\}, D^{(1)}(i) = \frac{1}{m}$.

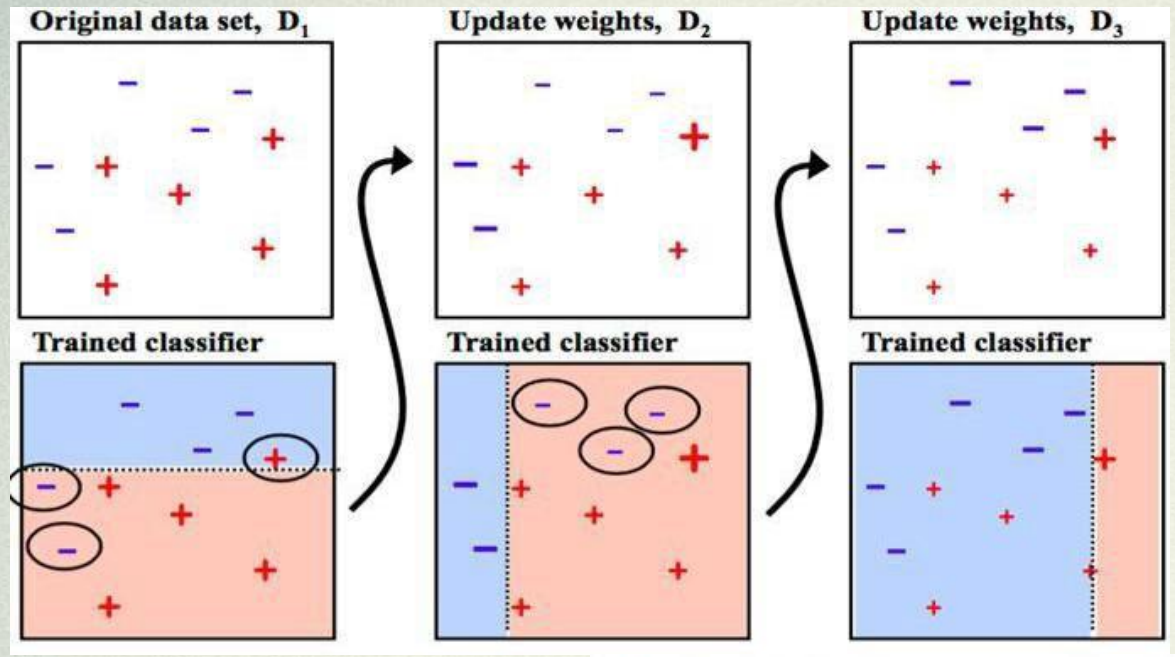
for $t = 1, \ldots, T$ **do**

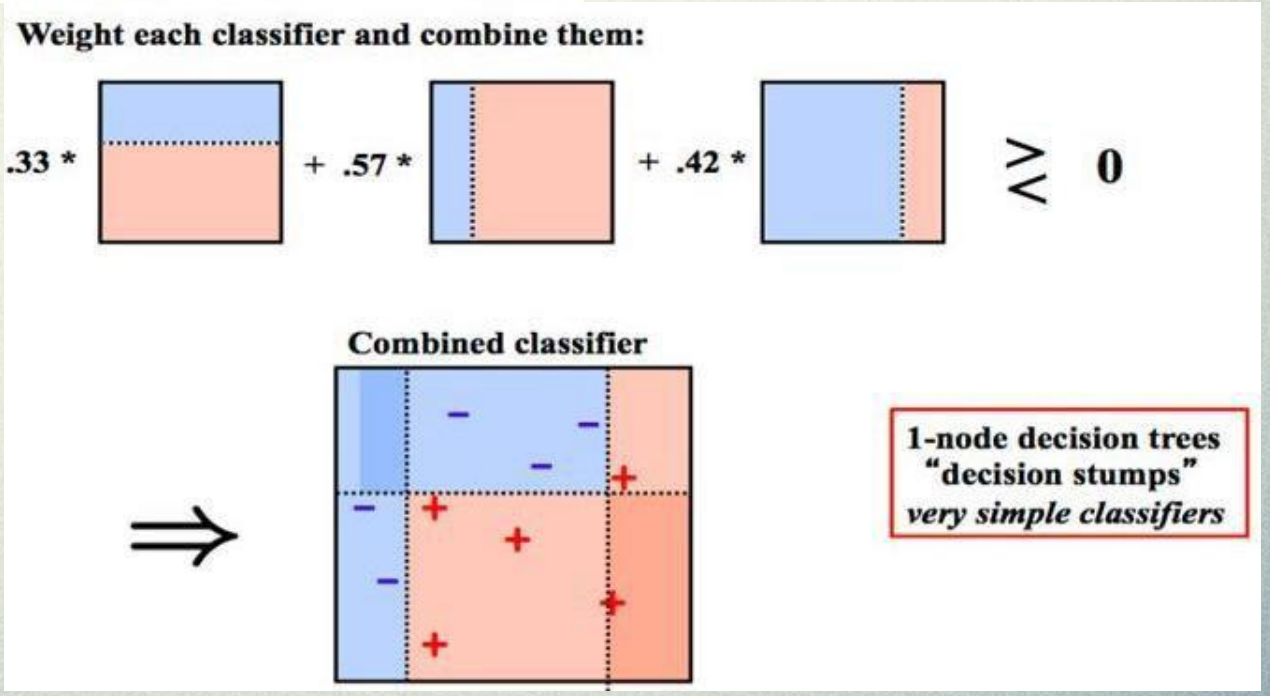
- Learn a classifier $f_t : \mathbb{R}^d \to \{-1, +1\}$ using distribution $D^{(t)}$
- Set $\epsilon_t = \sum_{i : f_t(\mathbf{x}_i) \neq y_i} D^{(t)}(i)$
- Choose $a_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
- Update the weight distribution over examples

$$\forall i \in \{1, \ldots, m\}, D^{(t+1)}(i) = \frac{D^{(t)}(i) e^{-a_t y_i f_t(\mathbf{x}_i)}}{Z^{(t)}}$$

where $Z^{(t)} = \sum_{i=1}^{m} D^{(t)}(i) e^{-a_t y_i f_t(\mathbf{x}_i)}$ is a normalization factor such that $D^{(t+1)}$ remains a distribution.

Output : The voted classifier $\forall \mathbf{x}, F(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} a_t f_t(\mathbf{x})\right)$


Original data set, D_1
Update weights, D_2
Update weights, D_3
Trained classifier
Trained classifier
Trained classifier


Weight each classifier and combine them:
.33 *
+ .57 *
+ .42 *
≥ 0
Combined classifier
1-node decision trees
"decision stumps"
very simple classifiers

Models: Other

The list before was not comprehensive, other algorithm and model classes include:

- **Conditional Random Fields (CRF)**
- **Markovian Models (HMMs)**
- **Dimensionality Reduction (PCA, PLS)**
- **Rule Learning (Apriori, Brill)**
- **More ...**

Wrapping fit and predict

We’ve already discussed a broad workflow, the following is a development workflow: