Classification

Tiphaine Viard

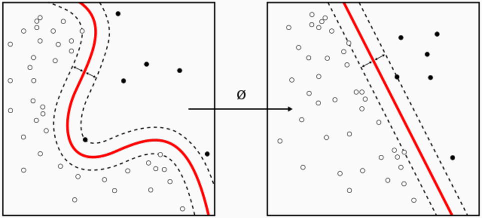
What is classification?

Wikipedia definition

Classification *is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known.*

Classification

Generalize known structures to apply to new data.



An e-mail program might attempt to classify an e-mail as "legitimate" or as "spam".

Spam example

Data set that describes e-mail features for deciding if it is spam.

Example

Contains “Money”	Domain type	Has attach.	Time received	spam
yes	com	yes	night	yes
yes	edu	no	night	yes
no	com	yes	night	yes
no	edu	no	day	no
no	com	no	day	no
yes	cat	no	day	yes

Spam example

Data set that describes e-mail features for deciding if it is spam.

Example

Contains "Money"	Domain type	Has attach.	Time received	spam
yes	com	yes	night	yes
yes	edu	no	night	yes
no	com	yes	night	yes
no	edu	no	day	no
no	com	no	day	no
yes	cat	no	day	yes

Assume we have to classify the following new instance:

Contains "Money"	Domain type	Has attach.	Time received	spam
yes	edu	yes	day	?

Definition

Given a set of classes $C_1 \ldots C_N$, a classifier algorithm builds a model that predicts for every unlabelled instance I the class C_i to which it belongs with accuracy.

Example
Spam filter

Example
Twitter Sentiment analysis: analyze tweets with positive or negative feelings

Example
Cat or Dog?

A note about classification

Finding a defining feature for classification can cause unjust consequences; **redlining** can be exacerbated by machine learning models.

Dutch welfare ordered to stop using SyRI (2020)

Find the balance between **technical** and **social** aspects

Privacy
Discrimination
Opacity
...

Symbolic AI
Baked-in fairness
"White-box" models
...

Basic Classifiers

Majority vote

Training

Compute the majority class in the dataset

Prediction

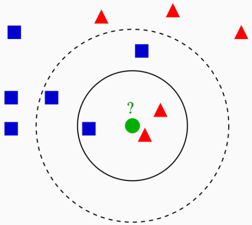
Output the majority class

Training

Store all instance (+ eventual index)

Prediction

Find the k closest point in the input and output the majority over those k points.



k-Nearest Neighbors (k-NN)

Training

Store all instance (+ eventual index)

Prediction

Find the k closest point in the input and output the majority over those k points.

Closest according to what metric?

L_1 vs L_2 vs L_∞ vs cos

Training

Store all instance (+ eventual index)

Prediction

Find the k closest point in the input and output the majority over those k points.

Closest according to what metric?

$$L_1 \quad vs \quad L_2 \quad vs \quad L_\infty \quad vs \quad COS$$

Rule of thumb: $k = \sqrt{n}$

Better k-NN

Problem. Finding distance from all points to all others is costly : $\mathcal{O}(n^2)$ time

We only care about close data points

Problem. Finding distance from all points to all others is costly : $\mathcal{O}(n^2)$ time

We only care about close data points

Solution : use **locality-sensitive hashing** (LSH)

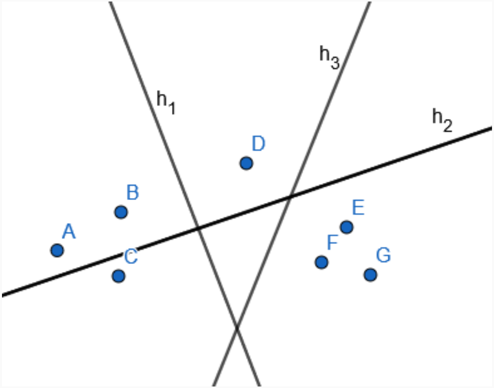
- Close points in the same bucket with high probability,
- Distant points are in different buckets with high probability.

Random hyperplanes for LSH

```
Generate K hyperplanes h_1, ..., h_K;
H_{i,k} ← −1, ∀i, ∀k;
for every datapoint x_i do
    if x_i · h_k ≥ 0 then
        H_{i,k} ← 0;
    end
    else
        H_{i,k} ← 1;
    end
end
```

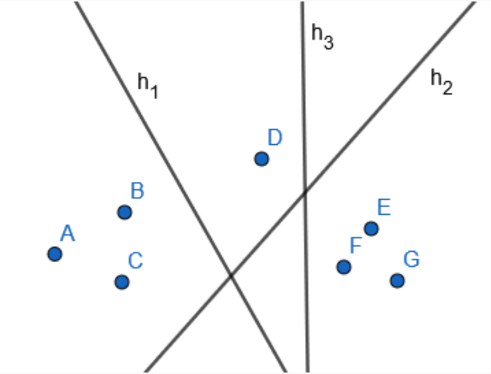
Repeat multiple times $\rightarrow$ more robust

LSH : an example



Hash	Data points
000	A, B
001	
010	C
011	
100	D
101	
110	
111	E, F, G

LSH : an example



Hash	Data points
000	A, B, C
001	
010	
011	
100	D
101	
110	
111	E, F, G

Bayes theorem

Formula

$$\frac{P(A) \times P(B|A)}{P(B)} = P(A|B)$$

Proof.

$$P(A \cap B) = P(A) \times P(B|A)$$

Proof.

$$P(A \cap B) = P(A) \times P(B|A)$$

$$P(A \cap B) = P(B) \times P(A|B)$$

Bayes theorem

Proof.

$$P(A \cap B) = P(A) \times P(B|A)$$

$$P(A \cap B) = P(B) \times P(A|B)$$

$$P(A) \times P(B|A) = P(B) \times P(A|B)$$

Bayes theorem

Proof.

$$P(A \cap B) = P(A) \times P(B|A)$$

$$P(A \cap B) = P(B) \times P(A|B)$$

$$P(A) \times P(B|A) = P(B) \times P(A|B)$$

$$\frac{P(A) \times P(B|A)}{P(B)} = P(A|B)$$

$\square$

Bayes theorem

Formula

$$\frac{P(A) \times P(B|A)}{P(B)} = P(A|B)$$

Interpretation

$$\text{prior} \times \frac{\text{likelihood}}{\text{evidence}} = \text{posterior}$$

Grouping attributes

$$P(C_i) \times \frac{P(\bar{x}|C_i)}{P(\bar{x})} = P(C_i|\bar{x})$$

Multiple attributes

$$P(C_i) \times \frac{\prod_j P(x_j|C_i)}{P(\bar{x})} = P(C_i|\bar{x})$$

Naive Bayes Classifier

Grouping attributes

$$P(C_i) \times \frac{P(\bar{x}|C_i)}{P(\bar{x})} = P(C_i|\bar{x})$$

Multiple attributes

$$P(C_i) \times \frac{\prod_j P(x_j|C_i)}{P(\bar{x})} = P(C_i|\bar{x})$$

With independence hypothesis!

Naive Bayes Classifier

Grouping attributes

$$P(C_i) \times \frac{P(\bar{x}|C_i)}{P(\bar{x})} = P(C_i|\bar{x})$$

Multiple attributes

$$P(C_i) \times \frac{\prod_j P(x_j|C_i)}{P(\bar{x})} = P(C_i|\bar{x})$$

With independence hypothesis!
$P(\bar{x})$ does not change with the class

Tree Methods

Classification

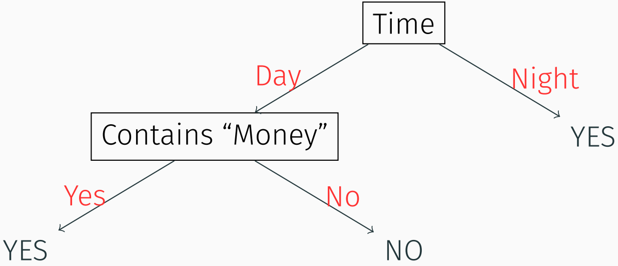
Data set that describes e-mail features for deciding if it is spam.

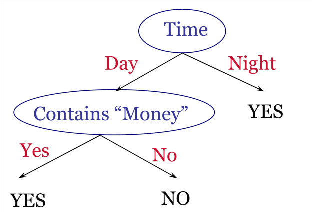
Example

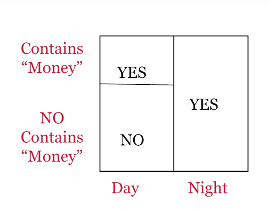
Contains "Money"	Domain type	Has attach.	Time received	spam
yes	com	yes	night	yes
yes	edu	no	night	yes
no	com	yes	night	yes
no	edu	no	day	no
no	com	no	day	no
yes	cat	no	day	yes

Assume we have to classify the following new instance:

Contains "Money"	Domain type	Has attach.	Time received	spam
yes	edu	yes	day	?







Recursive construction technique

- $A \leftarrow$ the *best* decision attribute for next *node*
- Assign A as decision attribute for *node*
- For each value of A, create new descendant of *node*
- Sort training examples to leaf nodes
- If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Decision Trees

- Interpretable: the DT is easy to understand
- Training is fast (greedy algorithm)
- Expressive: can approximate complex non-linear functions
- Complex function = large tree
- Large tree = more variance = overfitting

In practice, decision trees underperform compared with other methods

Bagging (Breiman, 1996)

Example

Dataset of 4 Instances : A, B, C, D

Classifier 1: B, A, C, B
Classifier 2: D, B, A, D
Classifier 3: B, A, C, B
Classifier 4: B, C, B, B
Classifier 5: D, C, A, C

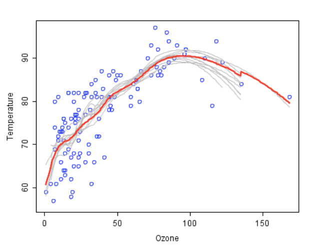
Bagging:

1. Bootstrap: generate multiple samples of data, train decision tree
2. Aggregate: output the average output of all models

Bagging

Bagging builds a set of M base models, with a bootstrap sample created by drawing random samples with replacement.

- Highly expressive: each model can estimate complex functions/boundaries
- a low-variance method: averaging the prediction of all models reduces variance (if M large enough)



Bagging implies **multiple models**, running on **different, overlapping** datasets

How can we evaluate its performance?

Evaluating bagging: out-of-bag error

Bagging implies **multiple models**, running on **different, overlapping** datasets

How can we evaluate its performance?

The **out-of-bag error**

1. for each data point x, average predicted output over models that do **not** contain x in bootstrap $\Rightarrow$ **point-wise out-of-bag error**;
2. **Average** point-wise out-of-bag error over training set.

Problems with bagging

In practice, trees are **strongly correlated**.

- Suppose that x_j is a strong predictor. Then most models will split on x_j;
- Then, each tree is essentially the same, and the averaged output is irrelevant.

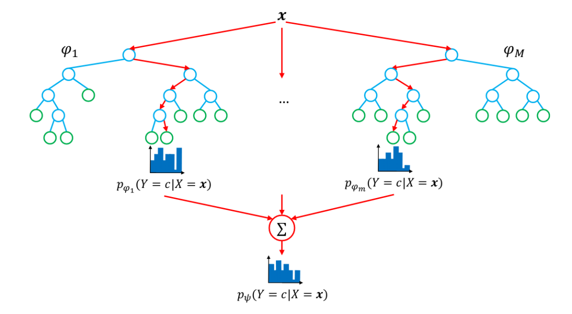
For F identically, dependently distributed variables with pairwise correlation ρ and variance σ^2, variance of mean:

$$\rho\sigma^2 + \frac{1-\rho}{F}\sigma^2$$

Variance reduction of bagging is **limited**.

Random Forests

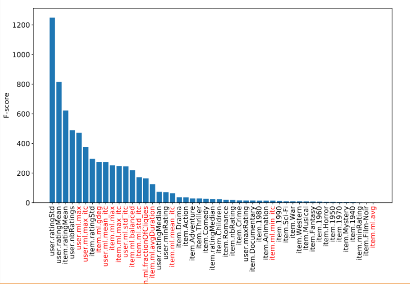
- Bagging
- Random Trees: trees that in each node only uses a random subset of k of the attributes



$\Rightarrow$ one of the most popular methods in machine learning.

Feature importance with random forests

- Record prediction accuracy on the oob samples for each tree;
- Randomly permute the data for column *j* in the oob samples;
- Record the accuracy again;
- Average the decrease in accuracy over all trees, and use as a measure of the importance of variable *j*.



Final thoughts on Random Forest

- Ensemble methods: **not easily interpretable**
- One of the best "off-the-shelf" methods, near to 0 tuning necessary
- **Averaging** and **randomization** offer fine control of the bias-variance trade-off
- Reasonably efficient: $\Omega(mk\hat{n}log^2\hat{n})$, with $\hat{n} \approx 0.63n$
- use `n_jobs` to make parallel
- `sklearn` implementation (Python + Cython) is by far the fastest available

Gradient Boosting

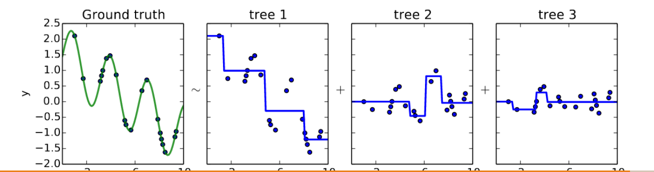
Train each of the M trees on the error of the precedent trees

$$\phi(x) = \sum_{m=0}^{M} \phi_m(x)$$

with each step being staged:

$$\phi_m(x) = \phi_{m-1}(x) + \hat{\phi}_m(x)$$

and $\hat{\phi}_m$ is a tree that approximates the gradient step.



Last thoughts on gradient boosting

- Usually more accurate than Random Forests
- Can adapt to most loss functions
- Under- and overfitting are adressed through regularization (learning rate, subsampling, term in loss function...)
- Tuning is harder than for random forests
- Slow to train (no parallelism!), fast predictions
- Still, blazing-fast implementation (Python and C++): XGBoost
- Easy to get *feature importance*

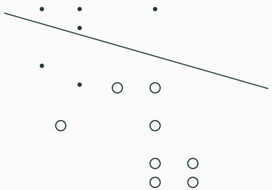
Gradient-based Methods

Training

Learn an hyperplan $\mathcal{P}$ separating well the two classes.

Prediction

What side of the hyperplan $\mathcal{P}$ is the point?



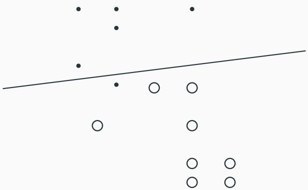
Based on the gradient of the logit function.

Training

Learn an hyperplan $\mathcal{P}$ separating well the two classes.

Prediction

What side of the hyperplan $\mathcal{P}$ is the point?



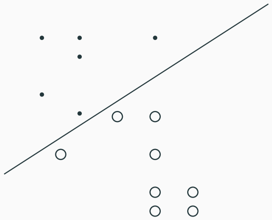
Based on the gradient of the logit function.

Training

Learn an hyperplan $\mathcal{P}$ separating well the two classes.

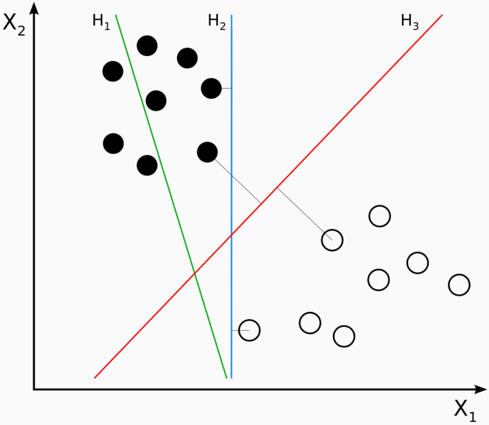
Prediction

What side of the hyperplan $\mathcal{P}$ is the point?



Based on the gradient of the logit function.

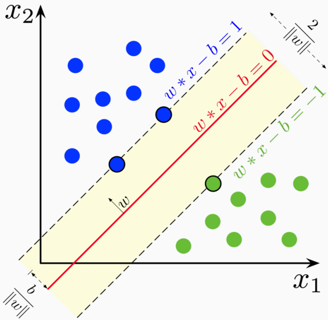
Similarly to logistic regression, we want to find the **best separating hyperplane**



SVM: under the hood

Suppose a **linear separation** exists

$$w^T x + b = 1, w^T x + b = -1$$

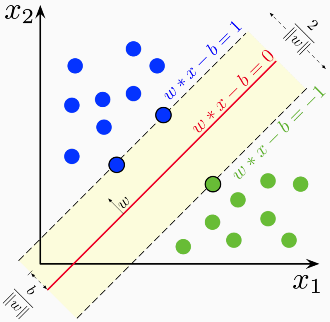


x_i near the margin determine the solution →**support vectors**

SVM: under the hood

Suppose a **linear separation** exists

$$w^T x_i + b \geq 1, \text{ if } y_i = 1, w^T x_i + b \leq -1, \text{ if } y_i = -1$$



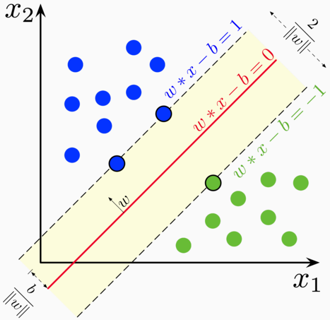
x_i near the margin determine the solution →**support vectors**

SVM: under the hood

Suppose a **linear separation** exists

$$w^T x_i + b \geq 1, \text{ if } y_i = 1, w^T x_i + b \leq -1, \text{ if } y_i = -1$$

$$y_i(w^T x + y) \geq 1$$

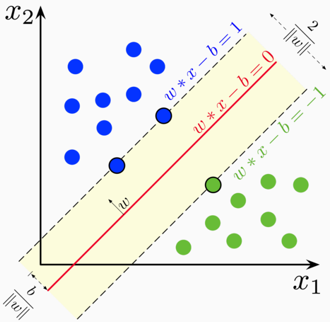


x_i near the margin determine the solution →**support vectors**

SVM: under the hood

Suppose a **linear separation** exists

$$\text{minimize } ||w|| \text{ subject to } y_i(w^T x_i + b) \geq 1, \text{ for } i = 1..n$$



x_i near the margin determine the solution →**support vectors**

Solving SVM

$$\text{minimize } ||w|| \text{ subject to } y_i(w^T x_i + b) \geq 1, \text{ for } i = 1..n$$

$||w||$ is the L_1 norm

The gradient of $||w||$ is $\frac{w}{||w||}$

Let's switch instead to L_2 norm

The gradient of $||w||^2$ is $2w$

$$\text{minimize } \frac{1}{2}||w||^2 \text{ subject to } y_i(w^T x_i + b) \geq 1, \text{ for } i = 1..n$$

This is a **quadratic optimization problem**.

Final thoughts on SVM

- What about non linearly separable data?

  Add misclassification term (number or distance)

- Use hinge function as regularization:
  $\min ||w||^2 + \lambda \left[\frac{1}{n} \sum_i^n \max(0, 1 - y_i(w^T x - b)\right]$
- λ trade-off between increasing margin and ensuring x_i outside of margin
- Only adapted to binary classification
- But you can do ”one-versus-all” classification