Part II: The Road Map of Machine Learning Algorithms

Xijia Liu
xijia.liu.18@gmail.com

Department of mathematics and mathematical statistics
Umeå University

December 2, 2019

Introduction

- Kernel methods: Support vector machine
- Ensemble methods:
    - Random forest
    - Adaboost
    - Artificial Neural Network

Kernel methods

- Feature mapping: $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$

$$(x_1, x_2) \rightarrow (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

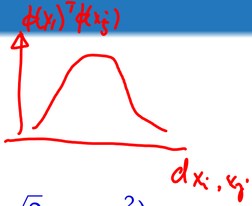
- Inexplicit mapping: ϕ

$x_i^T x_j$

$\phi(x)^T \phi(x)$

$$(x_1^2, \sqrt{2}x_1x_2, x_2^2)\begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix} = \underline{(x_1^2 + x_2^2)}^2$$

Kernel methods



- Feature mapping: $\phi : \mathbb{R}^2 \to \mathbb{R}^3$

$$(x_1, x_2) \to (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

- Inexplicit mapping: ϕ
- Kernel function: $\underline{\kappa_\phi(\mathbf{x}_i, \mathbf{x}_j)} = \cancel{\phi}(\mathbf{x}_i)^T\cancel{\phi}(\mathbf{x}_j)$
- Polynomial kenerl and Gaussian Kernel (RBF).

$$K_\phi(\mathbf{x}_i, \mathbf{x}_j) = (c + \gamma\, \mathbf{x}_i^T \mathbf{x}_j)^M \qquad K_\phi(\mathbf{x}_i, \mathbf{x}_j) = \exp\left\{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}\right\}$$

Kernel methods

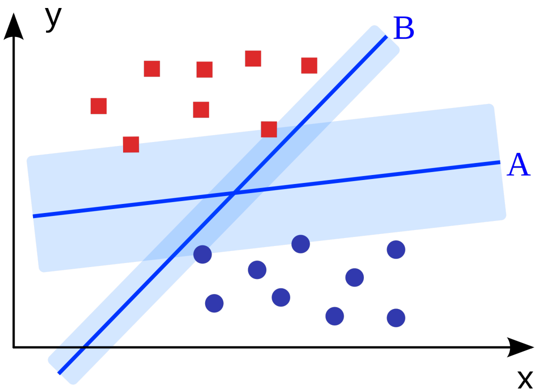
- Feature mapping: $\phi : \mathbb{R}^2 \to \mathbb{R}^3$

$$(x_1, x_2) \to (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

- Inexplicit mapping: ϕ
- Kernel function: $\kappa_\phi(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$
- Polynomial kenerl and Gaussian Kernel (RBF).
- Kernel tricks: obtain a non-linear model by embedding kernel function into different linear algorithms.
- Kernel PCA, Kernel regression, Support vector machine.

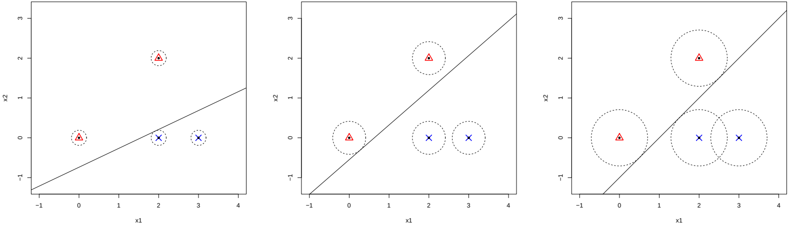
Maximum Margin Classifier

- The margin of the decision boundary should be as large as possible.



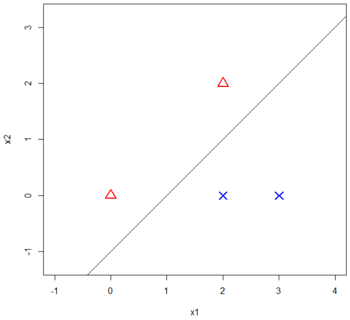
Maximum Margin Classifier

- A statistical illustration of this idea:

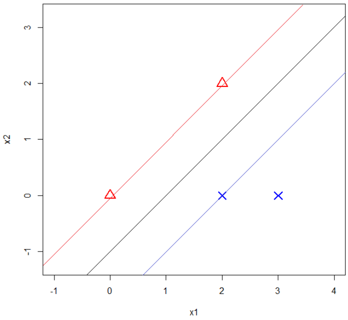


$\varepsilon \sim \mathcal{N}(0, \sigma_A^2)$ $\quad \sigma_B^2 \quad \sigma_C^2$

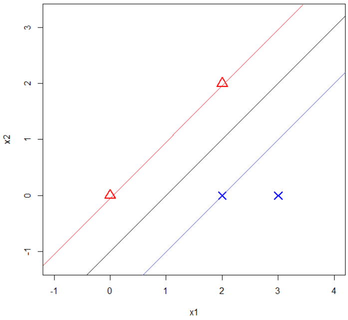
Support Vectors



Support Vectors

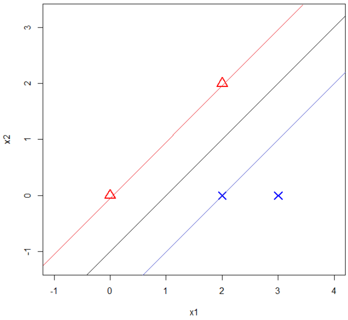


Support Vectors

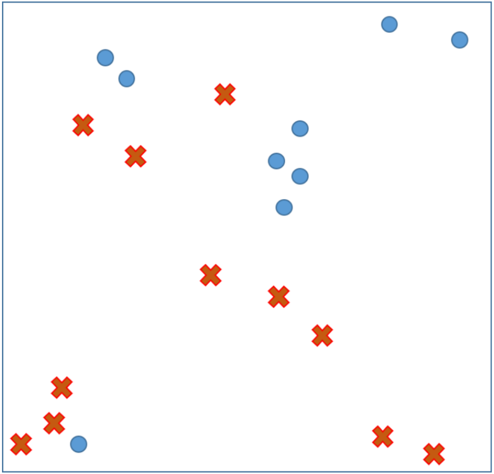


- Only the points on the boundary of margin have contributions to the final estimation results

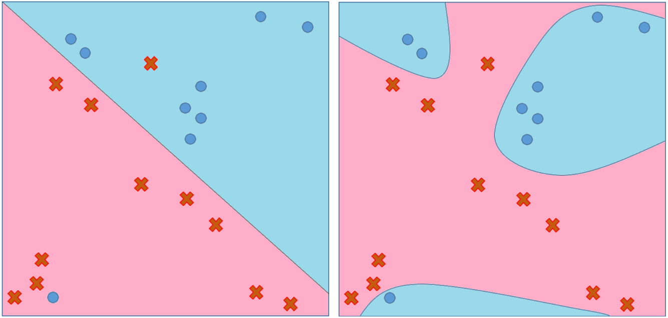
Support Vectors



- Only the points on the boundary of margin have contributions to the final estimation results
- Kernelized MMC $\rightarrow$ support vector machine

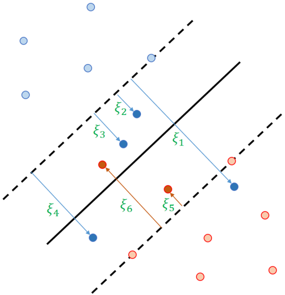


Over fitting in SVM



Soft Margin Classifier

- IDEA: We give up some high noise cases.
- For each observation, we introduce a fuzzy (Slackness) parameter, $\xi_i \geq 0$.
- Hyper-parameter: large C, less noise tolerance, high cost.



Summary of Kernel SVM

- SVM $=$ MMC $+$ Kernel trick.

Summary of Kernel SVM

- SVM $=$ MMC $+$ Kernel trick.
- Soft Margin Classifier $=$ SVM $+$ Cost.

Summary of Kernel SVM

- SVM $=$ MMC $+$ Kernel trick.
- Soft Margin Classifier $=$ SVM $+$ Cost.
- Hyper-parameters: parameters in kernel function and cost C

Summary of Kernel SVM

- SVM $=$ MMC $+$ Kernel trick.
- Soft Margin Classifier $=$ SVM $+$ Cost.
- Hyper-parameters: parameters in kernel function and cost C
- RBF kernel SVM is the most popular one.

Summary of Kernel SVM

- SVM $=$ MMC $+$ Kernel trick.
- Soft Margin Classifier $=$ SVM $+$ Cost.
- Hyper-parameters: parameters in kernel function and cost C
- RBF kernel SVM is the most popular one.
- The kernel parameter σ can be estimated from sample, see Caputo, Furesjo and Smola (2002).

Summary of Kernel SVM

- SVM $=$ MMC $+$ Kernel trick.
- Soft Margin Classifier $=$ SVM $+$ Cost.
- Hyper-parameters: parameters in kernel function and cost C
- RBF kernel SVM is the most popular one.
- The kernel parameter σ can be estimated from sample, see Caputo, Furesjo and Smola (2002).
- Tuning: exponential growing sequesces of C and σ, e.g. $2^{-5}, 2^{-3}, ..., 2^{15}$.

Introduction to ensemble methods

- Whether Swedish national football team can qualify for the Euro Cup in 2020?

Introduction to ensemble methods

- Whether Swedish national football team can qualify for the Euro Cup in 2020?
- Find an expert and follow his/her suggestion.

Introduction to ensemble methods

- Whether Swedish national football team can qualify for the Euro Cup in 2020?
- Find an expert and follow his/her suggestion.
- Aggregate optinions from all of us.

Introduction to ensemble methods

- Whether Swedish national football team can qualify for the Euro Cup in 2020?
- Find an expert and follow his/her suggestion.
- Aggregate optinions from all of us.
- Give more weights to some experts.

Introduction to ensemble methods

- Whether Swedish national football team can qualify for the Euro Cup in 2020?
- Find an expert and follow his/her suggestion.
- Select the "best" model by cross validation.
- Aggregate optinions from all of us.
- Give more weights to some experts.

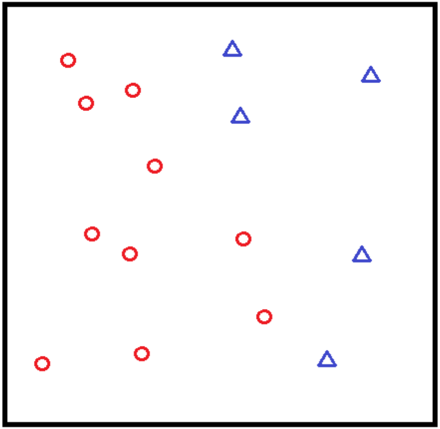
Introduction to ensemble methods

- Whether Swedish national football team can qualify for the Euro Cup in 2020?
- Find an expert and follow his/her suggestion.
- Select the "best" model by cross validation.
- Aggregate optinions from all of us.
- Uniform Aggregation. Bagging/Random forest.
- Give more weights to some experts.

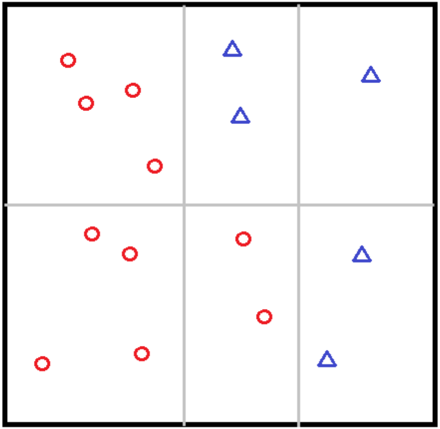
Introduction to ensemble methods

- Whether Swedish national football team can qualify for the Euro Cup in 2020?
- Find an expert and follow his/her suggestion.
- Select the "best" model by cross validation.
- Aggregate optinions from all of us.
- Uniform Aggregation. Bagging/Random forest.
- Give more weights to some experts.
- Non-uniform Aggregation. Adaboost.

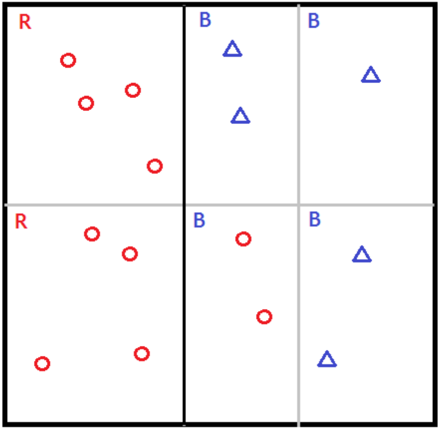
Uniform Aggregation



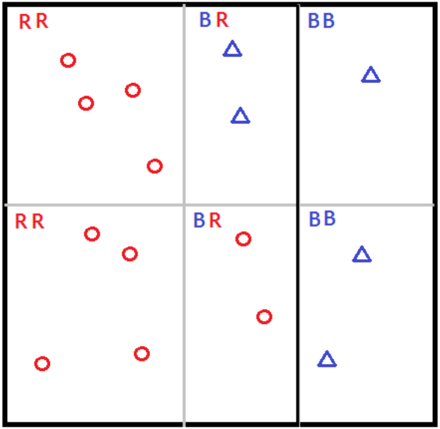
Uniform Aggregation



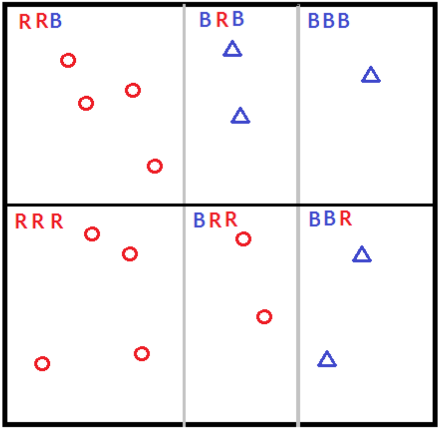
Uniform Aggregation



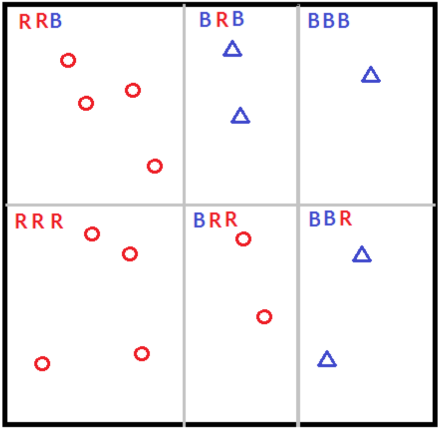
Uniform Aggregation



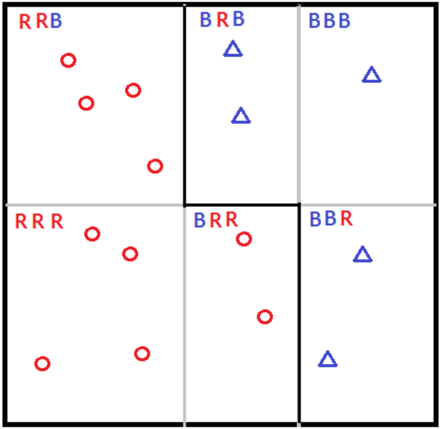
Uniform Aggregation



Uniform Aggregation



Uniform Aggregation



Uniform Aggregation

- Suppose we have learned M models, $f_i(\mathbf{x})$, from the data.

Uniform Aggregation

- Suppose we have learned M models, $f_i(\mathbf{x})$, from the data.
- Uniform Aggregation:

$$F(\mathbf{x}) = sign\left(\sum_{j=1}^{M} f_j(\mathbf{x})\right)$$

Uniform Aggregation

- Suppose we have learned M models, $f_i(\mathbf{x})$, from the data.
- Uniform Aggregation:

$$F(\mathbf{x}) = \frac{1}{M}\sum_{j=1}^{M} f_j(\mathbf{x})$$

Uniform Aggregation

- Suppose we have learned M models, $f_i(\mathbf{x})$, from the data.
- Uniform Aggregation:

$$F(\mathbf{x}) = \frac{1}{M} \sum_{j=1}^{M} f_j(\mathbf{x})$$

- Variance and bias decomposition:

$$E(f_j(\mathbf{x}) - g(\mathbf{x}))^2$$

Uniform Aggregation

- Suppose we have learned M models, $f_i(\mathbf{x})$, from the data.
- Uniform Aggregation:

$$F(\mathbf{x}) = \frac{1}{M}\sum_{j=1}^{M} f_j(\mathbf{x})$$

- Variance and bias decomposition:

$$E(f_j(\mathbf{x}) - g(\mathbf{x}))^2 = E\left(f_j(\mathbf{x}) - E(f_j(\mathbf{x})) + E(f_j(\mathbf{x})) - g(\mathbf{x})\right)^2$$

Uniform Aggregation

- Suppose we have learned M models, $f_i(\mathbf{x})$, from the data.
- Uniform Aggregation:

$$F(\mathbf{x}) = \frac{1}{M}\sum_{j=1}^{M} f_j(\mathbf{x})$$

- Variance and bias decomposition:

$$E(f_j(\mathbf{x}) - g(\mathbf{x}))^2 = E\left(f_j(\mathbf{x}) - E(f_j(\mathbf{x}))\right)^2 + \left(E(f_j(\mathbf{x})) - g(\mathbf{x})\right)^2$$

Uniform Aggregation

- Suppose we have learned M models, $f_i(\mathbf{x})$, from the data.
- Uniform Aggregation:

$$F(\mathbf{x}) = \frac{1}{M}\sum_{j=1}^{M} f_j(\mathbf{x})$$

- Variance and bias decomposition:

$$E(f_j(\mathbf{x}) - g(\mathbf{x}))^2 = E\left(f_j(\mathbf{x}) - E(f_j(\mathbf{x}))\right)^2 + \left(E(f_j(\mathbf{x})) - g(\mathbf{x})\right)^2$$

Uniform Aggregation

- Suppose we have learned M models, $f_i(\mathbf{x})$, from the data.
- Uniform Aggregation:

$$F(\mathbf{x}) = \frac{1}{M}\sum_{j=1}^{M} f_j(\mathbf{x})$$

$Var(X) = E(X - E(X))^2$

- Variance and bias decomposition:

$$E(f_j(\mathbf{x}) - g(\mathbf{x}))^2 = E\left(f_j(\mathbf{x}) - E(f_j(\mathbf{x}))\right)^2 + \left(E(f_j(\mathbf{x})) - g(\mathbf{x})\right)^2$$

Uniform Aggregation

- Suppose we have learned M models, $f_i(\mathbf{x})$, from the data.
- Uniform Aggregation:

$$F(\mathbf{x}) = \frac{1}{M}\sum_{j=1}^{M} f_j(\mathbf{x})$$

- Variance and bias decomposition:

$$E(f_j(\mathbf{x}) - g(\mathbf{x}))^2 = E\left(f_j(\mathbf{x}) - E(f_j(\mathbf{x}))\right)^2 + \left(E(f_j(\mathbf{x})) - g(\mathbf{x})\right)^2$$

- Better performance:

$$\text{avg Err}(f_j(\mathbf{x})) \geq \text{Err}(avg(f_j(\mathbf{x})))$$

Uniform Aggregation

- Suppose we have learned M models, $f_i(\mathbf{x})$, from the data.
- Uniform Aggregation:

$$F(\mathbf{x}) = \frac{1}{M}\sum_{j=1}^{M} f_j(\mathbf{x})$$

- Variance and bias decomposition:

$$E(f_j(\mathbf{x}) - g(\mathbf{x}))^2 = E\left(f_j(\mathbf{x}) - E(f_j(\mathbf{x}))\right)^2 + \left(E(f_j(\mathbf{x})) - g(\mathbf{x})\right)^2$$

- Better performance:

$$\text{avg Err}(f_j(\mathbf{x})) \geq \text{Err}(avg(f_j(\mathbf{x})))$$

- Large diversity among M models.

Uniform Aggregation

- Suppose we have learned M models, $f_i(\mathbf{x})$, from the data.
- Uniform Aggregation:

$$F(\mathbf{x}) = \frac{1}{M}\sum_{j=1}^{M} f_j(\mathbf{x})$$

- Variance and bias decomposition:

$$E(f_j(\mathbf{x}) - g(\mathbf{x}))^2 = E\left(f_j(\mathbf{x}) - E(f_j(\mathbf{x}))\right)^2 + \left(E(f_j(\mathbf{x})) - g(\mathbf{x})\right)^2$$

- Better performance:

$$\text{avg } \mathrm{Err}(f_j(\mathbf{x})) \geq \mathrm{Err}(avg(f_j(\mathbf{x})))$$

- Large diversity among M models.
- **Question**: How to get those M models?

Uniform Aggregation

- Suppose we have learned M models, $f_i(\mathbf{x})$, from the data.
- Uniform Aggregation:

$$F(\mathbf{x}) = \frac{1}{M}\sum_{j=1}^{M} f_j(\mathbf{x})$$

- Variance and bias decomposition:

$$E(f_j(\mathbf{x}) - g(\mathbf{x}))^2 = E\left(f_j(\mathbf{x}) - E(f_j(\mathbf{x}))\right)^2 + \left(E(f_j(\mathbf{x})) - g(\mathbf{x})\right)^2$$

- Better performance:

$$\text{avg Err}(f_j(\mathbf{x})) \geq \text{Err}(avg(f_j(\mathbf{x})))$$

- Large diversity among M models.
- **Question**: How to get those M models? **Bootstrap!**

Bagging (Bootstrap Aggregation)

- Basic idea: Apply Bootstrap resampling technique to generate different bootstrap samples, such that the classifiers learned from each bootstrap sample have large diversity.

Algorithm

Bagging (Bootstrap Aggregation)

- Basic idea: Apply Bootstrap resampling technique to generate different bootstrap samples, such that the classifiers learned from each bootstrap sample have large diversity.

$\underset{\sim}{x}_1 \cdots\cdots - \underset{\sim}{x}_n$

Algorithm

1. Take a random sample with replacement from data set, $\underset{\sim}{\mathbf{X}}^*$
2. Train a classifier from this random sample
3. Repeat step 1 and 2 B times, then perform uniform aggregation

Bagging (Bootstrap Aggregation)

- Basic idea: Apply Bootstrap resampling technique to generate different bootstrap samples, such that the classifiers learned from each bootstrap sample have large diversity.

Algorithm

1. Take a random sample with replacement from data set, $\mathbf{X}^*$
2. Train a classifier from this random sample
3. Repeat step 1 and 2 B times, then perform uniform aggregation

- A strategy to enhance the existing algorithm.

Bagging (Bootstrap Aggregation)

- Basic idea: Apply Bootstrap resampling technique to generate different bootstrap samples, such that the classifiers learned from each bootstrap sample have large diversity.

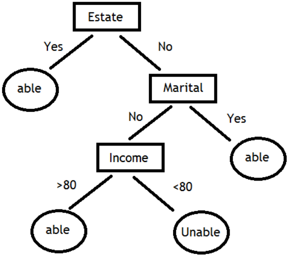
Algorithm

1. Take a random sample with replacement from data set, $\mathbf{X}^*$
2. Train a classifier from this random sample
3. Repeat step 1 and 2 B times, then perform uniform aggregation

- A strategy to enhance the existing algorithm. Any proper algorithm?

Decision Tree

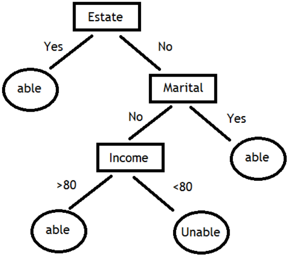
ID	Unable	Estate	Marital	Income
1	0	1	S	125
2	0	0	M	100
3	0	0	S	70
4	0	1	M	120
5	1	0	D	95
6	0	0	M	60
7	0	1	D	220
8	1	0	S	85
9	0	0	M	75
10	1	0	S	90



Decision Tree

Key: Sequentially split the feature space into small pieces.

ID	Unable	Estate	Marital	Income
1	0	1	S	125
2	0	0	M	100
3	0	0	S	70
4	0	1	M	120
5	1	0	D	95
6	0	0	M	60
7	0	1	D	220
8	1	0	S	85
9	0	0	M	75
10	1	0	S	90



Motivations

- The main weaknesses of decision tree is unstable and very sensitive to the data.
- Large variation! Bagging!

Motivations

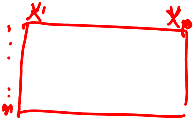
- The main weaknesses of decision tree is unstable and very sensitive to the data.
- Large variation! Bagging!
- Random forest $=$ Decision tree $+$ bagging

Algorithm

1. Draw a bootstrap sample from original data.

Algorithm

1. Draw a bootstrap sample from original data.
    - In order to get a big variation among different trees, random forest algorithm also random select a subset of feature variables for training.



Algorithm

1. Draw a bootstrap sample from original data.
    - In order to get a big variation among different trees, random forest algorithm also random select a subset of feature variables for training.
    - In literature, it is called random subspace, also a kind of feature mapping.

Algorithm

1. Draw a bootstrap sample from original data.
    - In order to get a big variation among different trees, random forest algorithm also random select a subset of feature variables for training.
    - In literature, it is called random subspace, also a kind of feature mapping.
2. Once those information have been decided, we can train a decision tree.

Algorithm

1. Draw a bootstrap sample from original data.
    - In order to get a big variation among different trees, random forest algorithm also random select a subset of feature variables for training.
    - In literature, it is called random subspace, also a kind of feature mapping.
2. Once those information have been decided, we can train a decision tree.
3. Repeat these procedure B times.

Algorithm

1. Draw a bootstrap sample from original data.
   - In order to get a big variation among different trees, random forest algorithm also random select a subset of feature variables for training.
   - In literature, it is called random subspace, also a kind of feature mapping.
2. Once those information have been decided, we can train a decision tree.
3. Repeat these procedure B times.
4. Apply uniform aggregation method on those trees, then we have our random forest model.

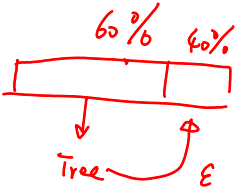
Out of Bagging Errors

- Out of bagging samples: a set of samples which is not included in the bootstrap sample.

Out of Bagging Errors

- Out of bagging samples: a set of samples which is not included in the bootstrap sample.
- The probability of an example is not included in a Bootstrap sample:

$$\left(1-\frac{1}{N}\right)^N \rightarrow \frac{1}{e} \approx 0.37$$



Out of Bagging Errors

- Out of bagging samples: a set of samples which is not included in the bootstrap sample.
- The probability of an example is not included in a Bootstrap sample:

$$\left(1 - \frac{1}{N}\right)^N \to \frac{1}{e} \approx 0.37$$

- Out of bagging classifier: for the bth Bootstrap sample, we can build up a tree, then apply this tree on the corresponding out of bagging samples.
- The error of out of bagging classifier is called out of bagging errors.

Out of Bagging Errors

- Out of bagging samples: a set of samples which is not included in the bootstrap sample.
- The probability of an example is not included in a Bootstrap sample:

$$\left(1-\frac{1}{N}\right)^N \to \frac{1}{e} \approx 0.37$$

- Out of bagging classifier: for the bth Bootstrap sample, we can build up a tree, then apply this tree on the corresponding out of bagging samples.
- The error of out of bagging classifier is called out of bagging errors.
- Self-validation.

Feature Importance

- Measure the importance for each features, $\mathcal{I}(x_i)$

Feature Importance

- Measure the importance for each features, $\mathcal{I}(x_i)$
- Question: how do you evaluate the importance of a football player?

Feature Importance

- Measure the importance for each features, $\mathcal{I}(x_i)$
- Question: how do you evaluate the importance of a football player?
- The number of goals? The number of successful passing lanes? The number of successful blocking? ...

Feature Importance

- Measure the importance for each features, $\mathcal{I}(x_i)$
- Question: how do you evaluate the importance of a football player?
- The number of goals? The number of successful passing lanes? The number of successful blocking? ...
- The best evaluation is if the team still can win the game without this player.

Feature Importance

- Measure the importance for each features, $\mathcal{I}(x_i)$
- Question: how do you evaluate the importance of a football player?
- The number of goals? The number of successful passing lanes? The number of successful blocking? ...
- The best evaluation is if the team still can win the game without this player.
- Replace the feature x_i by permuted x_i^P

Feature Importance

- Measure the importance for each features, $\mathcal{I}(x_i)$
- Question: how do you evaluate the importance of a football player?
- The number of goals? The number of successful passing lanes? The number of successful blocking? ...
- The best evaluation is if the team still can win the game without this player.
- Replace the feature x_i by permuted x_i^P
- Then we can measure the importance of x_i by the difference of performance

$$\mathcal{I}(x_j) = Err_{OOB}(x_j) - Err_{OOB}(x_j^p)$$

Turning Random Forest

- Two hyper-parameters: the number of trees, N_t and the number of features randomly selected on each spliting, N_f.

Turning Random Forest

- Two hyper-parameters: the number of trees, N_t and the number of features randomly selected on each spliting, N_f.
- Suggested tuning procedure:

Turning Random Forest

- Two hyper-parameters: the number of trees, N_t and the number of features randomly selected on each spliting, N_f.
- Suggested tuning procedure:
  - Fix N_f as the squar root of the total number of feature variables, build 5 random forests with different $N_t = 300, 500, ..., 1000$.

Turning Random Forest

- Two hyper-parameters: the number of trees, N_t and the number of features randomly selected on each spliting, N_f.
- Suggested tuning procedure:
  - Fix N_f as the squar root of the total number of feature variables, build 5 random forests with different $N_t = 300, 500, ..., 1000$.
  - Choose the best forest given the OOB errors.

Turning Random Forest

- Two hyper-parameters: the number of trees, N_t and the number of features randomly selected on each spliting, N_f.
- Suggested tuning procedure:
  - Fix N_f as the squar root of the total number of feature variables, build 5 random forests with different $N_t = 300, 500, ..., 1000$.
  - Choose the best forest given the OOB errors.
  - Fix N_t and choose different numbers around the squar root of the total number of feature variables.

Turning Random Forest

- Two hyper-parameters: the number of trees, N_t and the number of features randomly selected on each spliting, N_f.
- Suggested tuning procedure:
  - Fix N_f as the squar root of the total number of feature variables, build 5 random forests with different $N_t = 300, 500, ..., 1000$.
  - Choose the best forest given the OOB errors.
  - Fix N_t and choose different numbers around the squar root of the total number of feature variables.
  - Build up the classifier, then select the best given OOB errors.

Boosting and Adaboosting

- Uniform aggregation: Bagging, Random Forest.

Boosting and Adaboosting

- Uniform aggregation: Bagging, Random Forest.
- Boosting: Non-uniform aggregation.

Boosting and Adaboosting

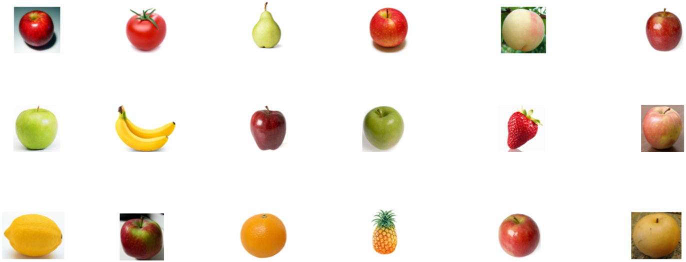
- Uniform aggregation: Bagging, Random Forest.
- Boosting: Non-uniform aggregation.
- Adaboost, Freund Y. and Schapire R.E. (1997):

Boosting and Adaboosting

- Uniform aggregation: Bagging, Random Forest.
- Boosting: Non-uniform aggregation.
- Adaboost, Freund Y. and Schapire R.E. (1997): Learn from failures and mistakes.

Adaboost

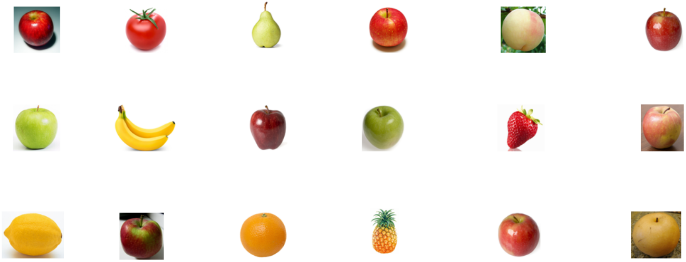
- Teach kids to distinguish apple from other fruits.



Adaboost

- Teach kids to distinguish apple from other fruits.

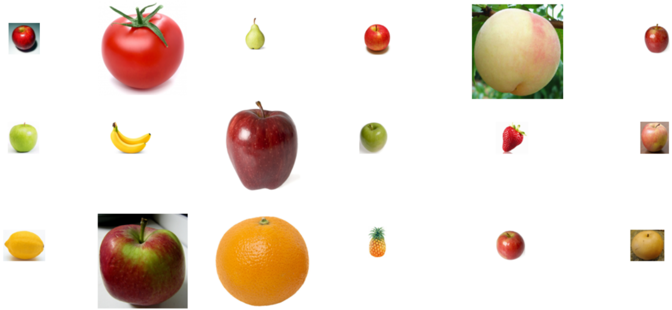
Circle



Adaboost

- Teach kids to distinguish apple from other fruits.

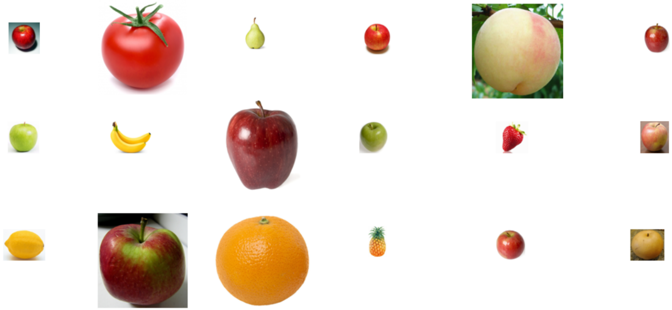
Circle



Adaboost

- Teach kids to distinguish apple from other fruits.

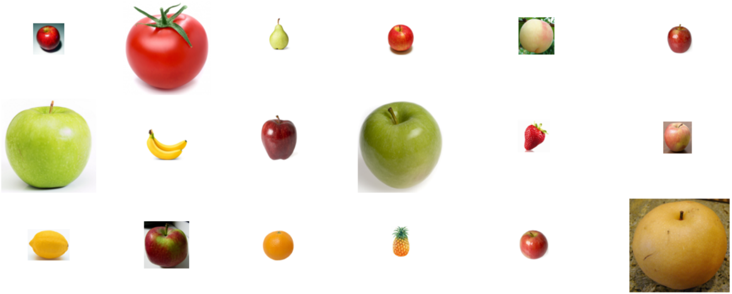
Circle $+$ **Red**



Adaboost

- Teach kids to distinguish apple from other fruits.

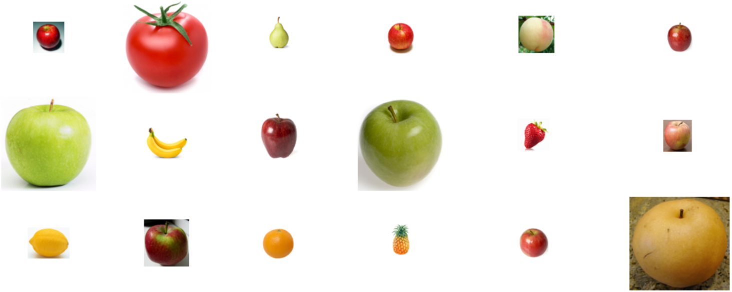
Circle + **Red**



Adaboost

- Teach kids to distinguish apple from other fruits.

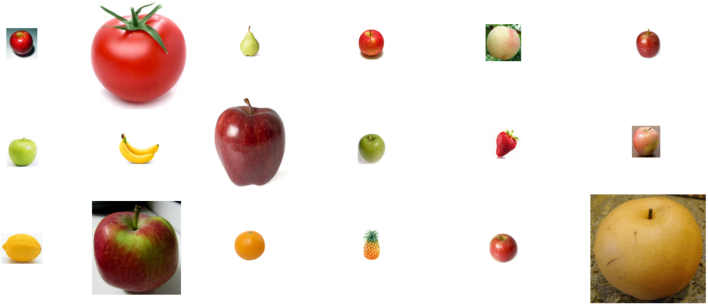
Circle + **Red** + **Green**



Adaboost

- Teach kids to distinguish apple from other fruits.

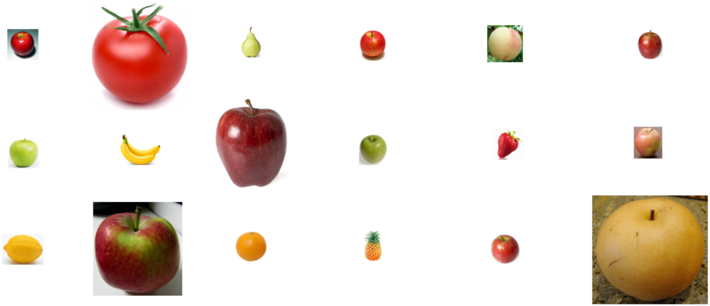
Circle + Red + Green



Adaboost

- Teach kids to distinguish apple from other fruits.

Circle + **Red** + **Green** + **Stem**



Weighted Classifer

- Decision Stump: make a prediction based on the value of a single input feature variable.

Weighted Classifer

- Decision Stump: make a prediction based on the value of a single input feature variable.
- Pay different extents of attention to different cases:

Weighted Classifer

- Decision Stump: make a prediction based on the value of a single input feature variable.
- Pay different extents of attention to different cases: Weighted sample,

Weighted Classifer

- Decision Stump: make a prediction based on the value of a single input feature variable.
- Pay different extents of attention to different cases: Weighted sample, Bootstrap!

Weighted Classifer

- Decision Stump: make a prediction based on the value of a single input feature variable.
- Pay different extents of attention to different cases: Weighted sample, Bootstrap!

Example

$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$

Weighted Classifer

- Decision Stump: make a prediction based on the value of a single input feature variable.
- Pay different extents of attention to different cases: Weighted sample, Bootstrap!

Example

$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$
$\mathcal{D}_B = \{(x_1, y_1), (x_2, y_2), (x_2, y_2), (x_3, y_3)\}$

Weighted Classifer

- Decision Stump: make a prediction based on the value of a single input feature variable.
- Pay different extents of attention to different cases: Weighted sample, Bootstrap!

Example

$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$
$\mathcal{D}_B = \{1(x_1, y_1), 2(x_2, y_2), 1(x_3, y_3), 0(x_4, y_4)\}$

Weighted Classifer

- Decision Stump: make a prediction based on the value of a single input feature variable.
- Pay different extents of attention to different cases: Weighted sample, Bootstrap!

Example

$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$
$\mathcal{D}_B = \{1(x_1, y_1), 2(x_2, y_2), 1(x_3, y_3), 0(x_4, y_4)\}$

- For data $\mathcal{D}$, there are 3 potential cuts. We count each case only once in the calculation of mis-classification errors.

Weighted Classifer

- Decision Stump: make a prediction based on the value of a single input feature variable.
- Pay different extents of attention to different cases: Weighted sample, Bootstrap!

Example

$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$
$\mathcal{D}_B = \{1(x_1, y_1), 2(x_2, y_2), 1(x_3, y_3), 0(x_4, y_4)\}$

- For data $\mathcal{D}$, there are 3 potential cuts. We count each case only once in the calculation of mis-classification errors.
- For data $\mathcal{D}_B$, only 2 potential cuts. We count the case, (x_2, y_2) twice in the calculation of mis-classification errors.

Weighted Classifer

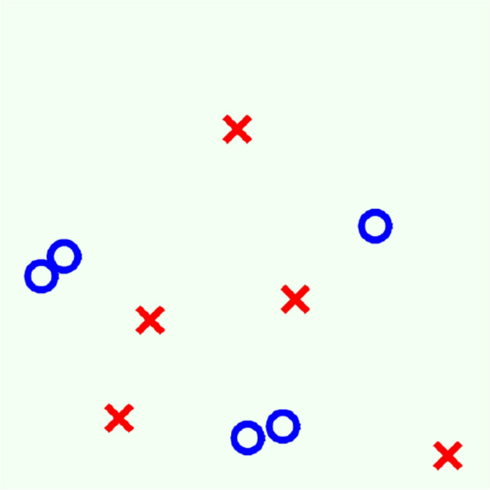
- Decision Stump: make a prediction based on the value of a single input feature variable.
- Pay different extents of attention to different cases: Weighted sample, Bootstrap!

Example

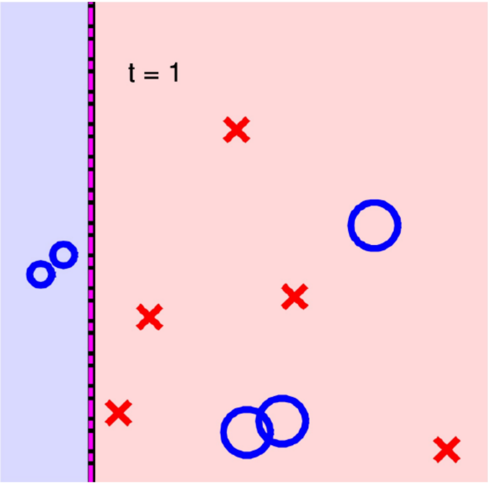
$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$
$\mathcal{D}_B = \{1(x_1, y_1), 2(x_2, y_2), 1(x_3, y_3), 0(x_4, y_4)\}$

- For data $\mathcal{D}$, there are 3 potential cuts. We count each case only once in the calculation of mis-classification errors.
- For data $\mathcal{D}_B$, only 2 potential cuts. We count the case, (x_2, y_2) twice in the calculation of mis-classification errors.
- Adaboost: a reweight scheme, such that the mis-classified examples get more weights in the next round.

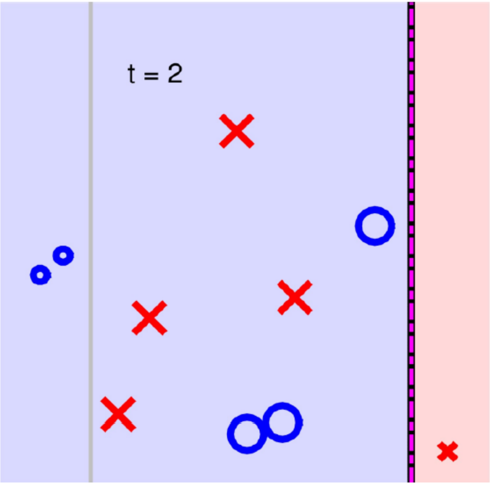
Toy example in Action



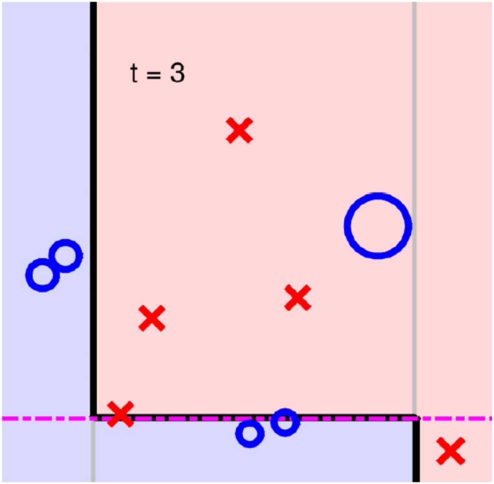
Toy example in Action



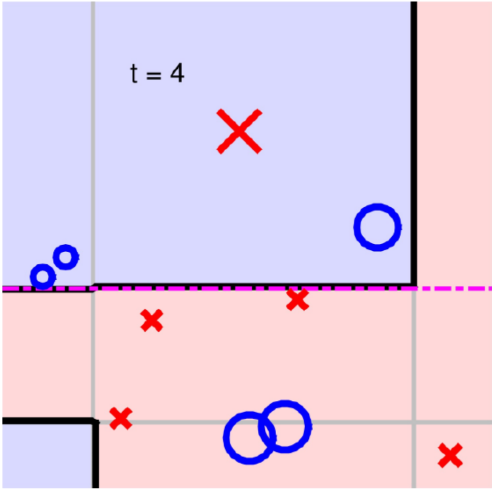
Toy example in Action



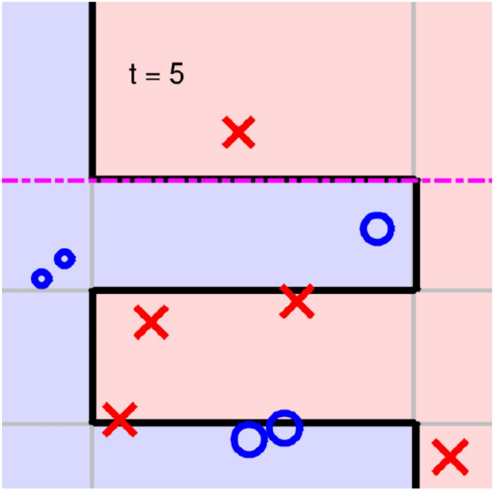
Toy example in Action



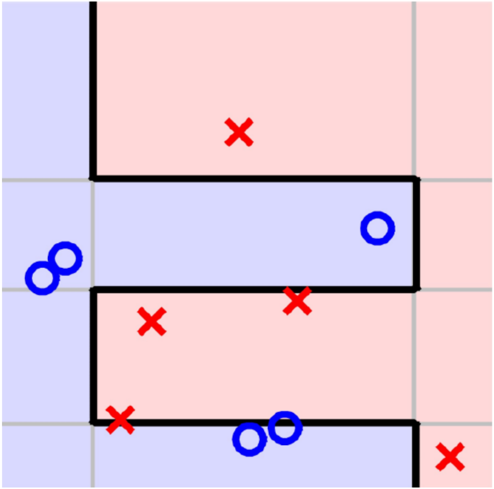
Toy example in Action



Toy example in Action



Toy example in Action



ANN in Ensemble methods

- Uniform aggregation: $F(\mathbf{x}) = Sign\left(\sum_{j=1}^{J} f_j(\mathbf{x})\right)$

ANN in Ensemble methods

- Uniform aggregation: $F(\mathbf{x}) = Sign\left(\sum_{j=1}^{J} f_j(\mathbf{x})\right)$
- Random forest $=$ Bagging $+$ Decision tree

ANN in Ensemble methods

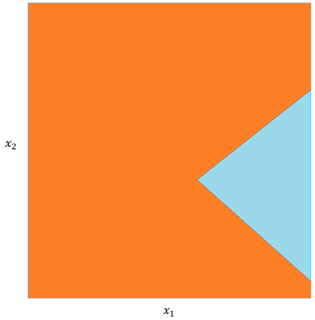
- Uniform aggregation: $F(\mathbf{x}) = Sign\left(\sum_{j=1}^{J} f_j(\mathbf{x})\right)$
- Random forest $=$ Bagging $+$ Decision tree
- **Question**: Can we learn f_j as a perceptron, $f_j = Sign\left(\mathbf{w}^T\mathbf{x} + w_0\right)$?

ANN in Ensemble methods

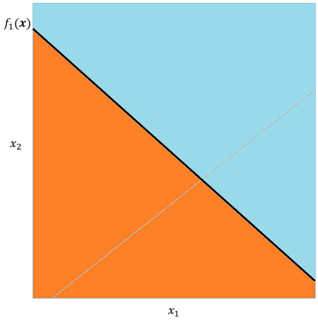
- Uniform aggregation: $F(\mathbf{x}) = Sign\left(\sum_{j=1}^{J} f_j(\mathbf{x})\right)$
- Random forest = Bagging + Decision tree
- **Question**: Can we learn f_j as a perceptron, $f_j = Sign\left(\mathbf{w}^T\mathbf{x} + w_0\right)$?
- **Question**: Can we learn different weights for different perceptrons from the target variable $\mathbf{y}$, such that the final model $F(\mathbf{x})$, is a non-uniform aggregation of those perceptrons?

$$F(\mathbf{x}) = Sign\left(\alpha_0 + \sum_{j=1}^{J} \alpha_j Sign(\mathbf{w}_j^T\mathbf{x} + w_0)\right)$$

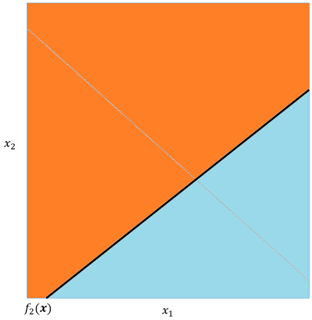
Motivations



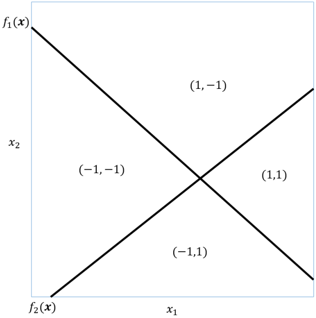
Motivations



Motivations

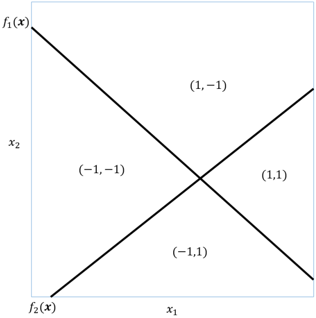


Motivations



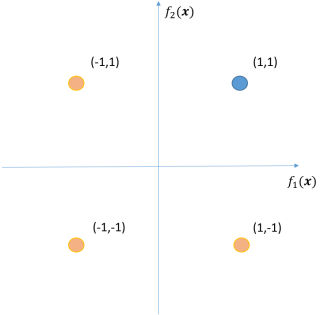
Motivations

- 'AND' operator: $AND(f_1, f_2)$



Motivations

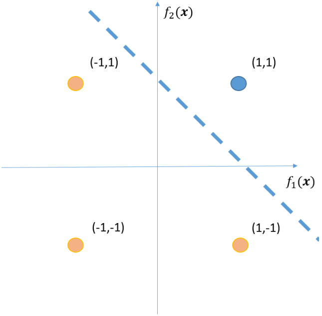
- 'AND' operator: $AND(f_1, f_2)$



Motivations

- 'AND' operator: $AND(f_1, f_2)$
- Find a line split the space.

$$\alpha_0 + \alpha_1 f_1 + \alpha_2 f_2 = 0$$



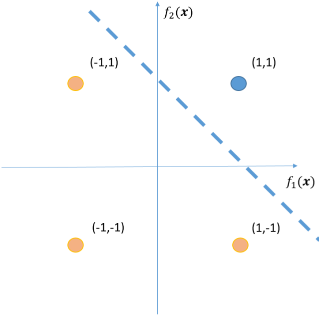
Motivations

- 'AND' operator: $AND(f_1, f_2)$
- Find a line split the space.

$$\alpha_0 + \alpha_1 f_1 + \alpha_2 f_2 = 0$$

- The final classifier is

$$Sign\left(-1 + f_1(\mathbf{x}) + f_2(\mathbf{x})\right)$$



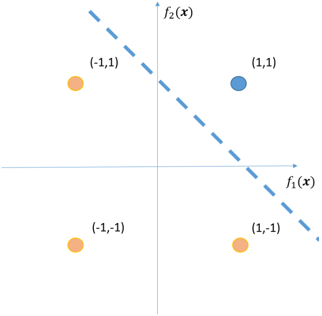
Motivations

- 'AND' operator: $AND(f_1, f_2)$
- Find a line split the space.

$$\alpha_0 + \alpha_1 f_1 + \alpha_2 f_2 = 0$$

- The final classifier is

$$Sign\left(-1 + Sign\left(\mathbf{w}_1^T\mathbf{x} + w_{10}\right) + f_2(\mathbf{x})\right)$$



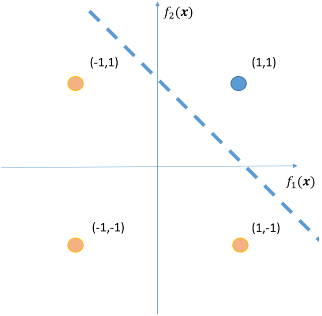
Motivations

- 'AND' operator: $AND(f_1, f_2)$
- Find a line split the space.

$$\alpha_0 + \alpha_1 f_1 + \alpha_2 f_2 = 0$$

- The final classifier is

$$Sign\left(-1 + f_1(\mathbf{x}) + Sign\left(\mathbf{w}_2^T \mathbf{x} + w_{20}\right)\right)$$



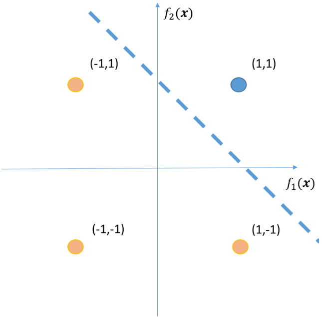
Motivations

- 'AND' operator: $AND(f_1, f_2)$
- Find a line split the space.

$$\alpha_0 + \alpha_1 f_1 + \alpha_2 f_2 = 0$$

- The final classifier is

$$Sign\left(\alpha_0 + \sum_{j=1}^{2} \alpha_j Sign\left(\mathbf{w}_j^T \mathbf{x} + w_{j0}\right)\right)$$



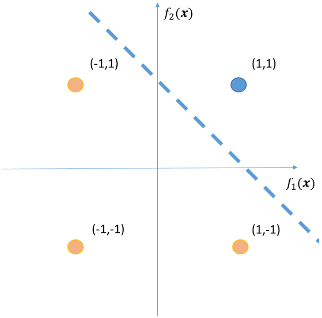
Motivations

- 'AND' operator: $AND(f_1, f_2)$
- Find a line split the space.

$$\alpha_0 + \alpha_1 f_1 + \alpha_2 f_2 = 0$$

- The final classifier is

$$Sign\left(\alpha_0 + \sum_{j=1}^{J} \alpha_j Sign\left(\mathbf{w}_j^T \mathbf{x} + w_{j0}\right)\right)$$



Motivations

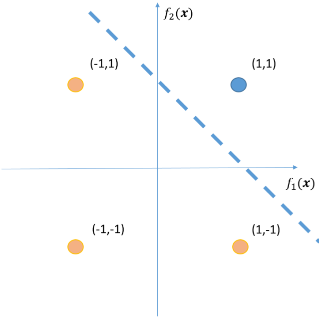
- 'AND' operator: $AND(f_1, f_2)$
- Find a line split the space.

$$\alpha_0 + \alpha_1 f_1 + \alpha_2 f_2 = 0$$

- The final classifier is

$$Sign\left(\alpha_0 + \sum_{j=1}^{J} \alpha_j Sign\left(\mathbf{w}_j^T \mathbf{x} + w_{j0}\right)\right)$$

- 'OR' and 'NOT'



Motivations

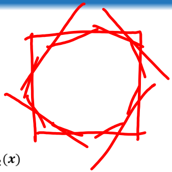
- 'AND' operator: $AND(f_1, f_2)$
- Find a line split the space.

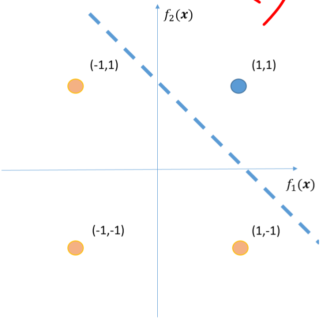
$$\alpha_0 + \alpha_1 f_1 + \alpha_2 f_2 = 0$$

- The final classifier is

$$Sign\left(\alpha_0 + \sum_{j=1}^{J} \alpha_j Sign\left(\mathbf{w}_j^T \mathbf{x} + w_{j0}\right)\right)$$

- 'OR' and 'NOT'
- Any convex set can be approximated by this model if J is large enough.





Non-uniform Aggregation of Perceptrons

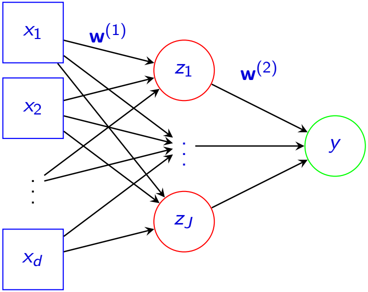
- Non-uniform Aggregation of Perceptrons can be presented as

$$F(\mathbf{x}) = Sign\left(w_0^{(2)} + \sum_{j=1}^{J} w_j^{(2)} Sign\left(\sum_{i=1}^{d} w_{j,i}^{(1)} x_i + w_{j,0}^{(1)}\right)\right)$$

Non-uniform Aggregation of Perceptrons

- Non-uniform Aggregation of Perceptrons can be presented as

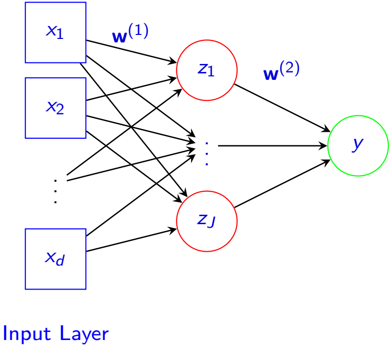
$$F(\mathbf{x}) = Sign\left(w_0^{(2)} + \sum_{j=1}^{J} w_j^{(2)}\, Sign\left(\sum_{i=1}^{d} w_{j,i}^{(1)} x_i + w_{j,0}^{(1)}\right)\right)$$



Non-uniform Aggregation of Perceptrons

- Non-uniform Aggregation of Perceptrons can be presented as

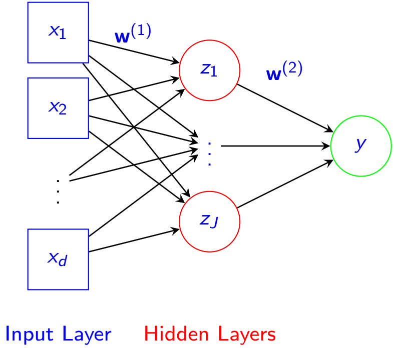
$$F(\mathbf{x}) = Sign\left(w_0^{(2)} + \sum_{j=1}^{J} w_j^{(2)} Sign\left(\sum_{i=1}^{d} w_{j,i}^{(1)} x_i + w_{j,0}^{(1)}\right)\right)$$



Non-uniform Aggregation of Perceptrons

- Non-uniform Aggregation of Perceptrons can be presented as

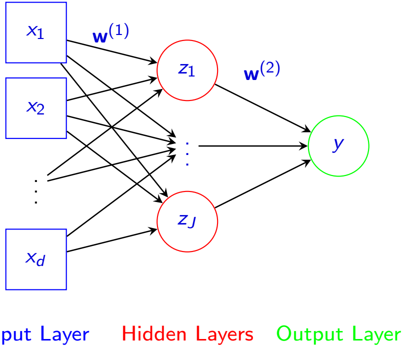
$$F(\mathbf{x}) = Sign\left(w_0^{(2)} + \sum_{j=1}^{J} w_j^{(2)}\, Sign\left(\sum_{i=1}^{d} w_{j,i}^{(1)} x_i + w_{j,0}^{(1)}\right)\right)$$



Non-uniform Aggregation of Perceptrons

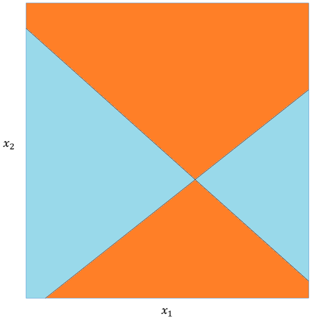
- Non-uniform Aggregation of Perceptrons can be presented as

$$F(\mathbf{x}) = Sign\left(w_0^{(2)} + \sum_{j=1}^{J} w_j^{(2)} Sign\left(\sum_{i=1}^{d} w_{j,i}^{(1)} x_i + w_{j,0}^{(1)}\right)\right)$$



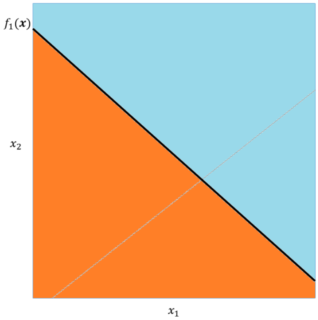
Motivations

- A non-convex set.



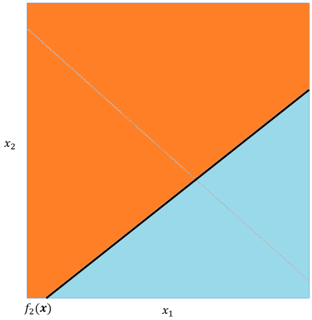
Motivations

- A non-convex set.



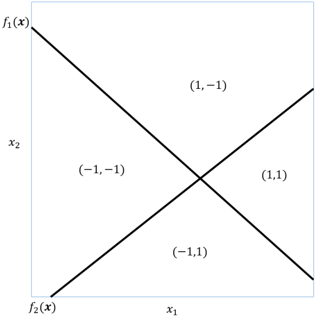
Motivations

- A non-convex set.



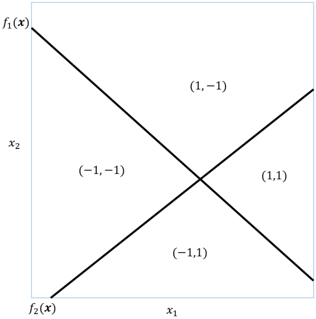
Motivations

- A non-convex set.



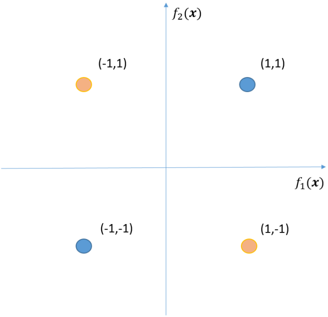
Motivations

- A non-convex set.
- 'XOR' operator.



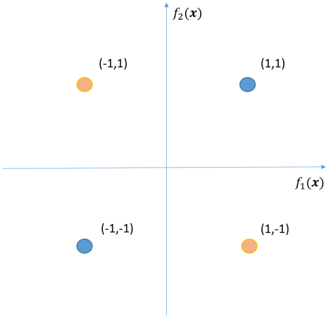
Motivations

- A non-convex set.
- 'XOR' operator.



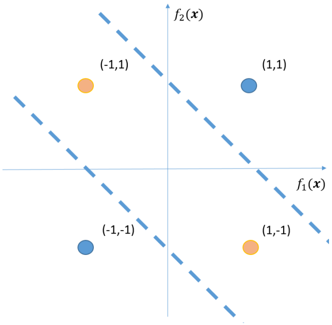
Motivations

- A non-convex set.
- 'XOR' operator.
- Not linear separable... Any idear?



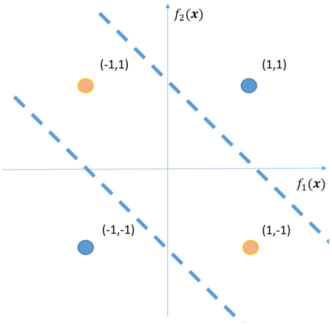
Motivations

- A non-convex set.
- 'XOR' operator.
- Not linear separable... Any idear?



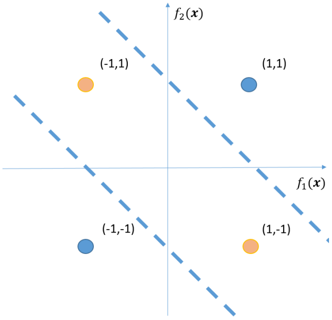
Motivations

- A non-convex set.
- 'XOR' operator.
- Not linear separable... Any idear?
- Multi-Layers Perceptron (MLP).



Motivations

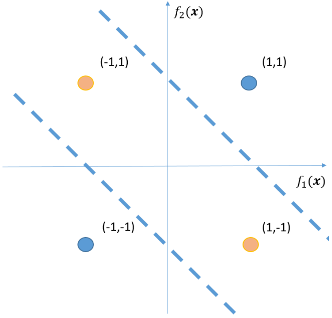
- A non-convex set.
- 'XOR' operator.
- Not linear separable... Any idear?
- Multi-Layers Perceptron (MLP).



There is nothing difficulty cann't be solved by eating a hamburg at MÄX, if so, just eat one more.

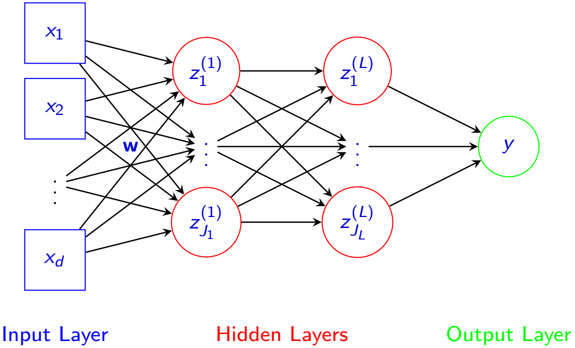
Motivations

- A non-convex set.
- 'XOR' operator.
- Not linear separable... Any idear?
- Multi-Layers Perceptron (MLP).



There is nothing difficulty cann't be solved by Multi-layers perceptron, if so, just add one more layer

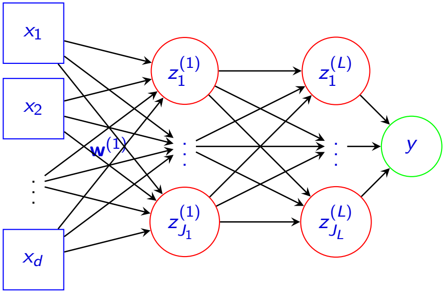
Multi-Layers Perceptron



From MLP to Artificial Neural Network

- The general MLP can be represented as

$$
\begin{aligned}
z_j^{(1)} &= Sign\left(\mathbf{w}_j^{(1)T}\mathbf{x}\right)\\
&\vdots\\
z_j^{(L)} &= Sign\left(\mathbf{w}_j^{(L)T}\mathbf{z}^{(L-1)}\right)\\
y &= Sign\left(\mathbf{w}_j^{(L+1)T}\mathbf{z}^{(L)}\right)
\end{aligned}
$$

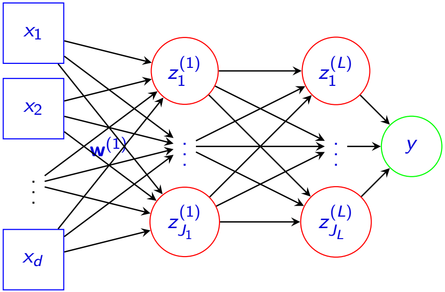


From MLP to Artificial Neural Network

- The general MLP can be represented as

$$\begin{aligned} z_j^{(1)} &= Sign\left(\mathbf{w}_j^{(1)T}\mathbf{x}\right) \\ &\vdots \\ z_j^{(L)} &= Sign\left(\mathbf{w}_j^{(L)T}\mathbf{z}^{(L-1)}\right) \\ y &= Sign\left(\mathbf{w}_j^{(L+1)T}\mathbf{z}^{(L)}\right) \end{aligned}$$

- Neurons: $z_j^{(L)}$, latent variable.



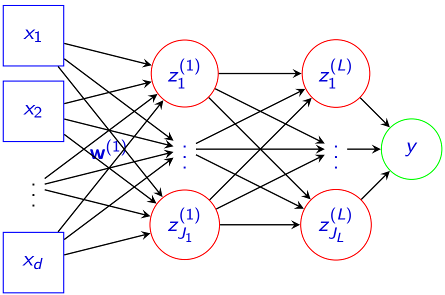
From MLP to Artificial Neural Network

- The general MLP can be represented as

$$
\begin{aligned}
z_j^{(1)} &= Sign\left(\mathbf{w}_j^{(1)T}\mathbf{x}\right)\\
&\vdots\\
z_j^{(L)} &= Sign\left(\mathbf{w}_j^{(L)T}\mathbf{z}^{(L-1)}\right)\\
y &= Sign\left(\mathbf{w}_j^{(L+1)T}\mathbf{z}^{(L)}\right)
\end{aligned}
$$

- Neurons: $z_j^{(L)}$, latent variable.
- Activation function, $\sigma(\cdot)$: $Sign(\cdot)$, Identity function, logit function and so on.

z : S

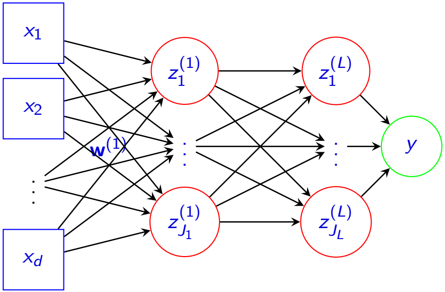


From MLP to Artificial Neural Network

- The general MLP can be represented as

$$
\begin{aligned}
z_j^{(1)} &= \sigma\left(\mathbf{w}_j^{(1)T}\mathbf{x}\right) \\
&\vdots \\
z_{j_k}^{(L)} &= \sigma\left(\mathbf{w}_{j_k}^{(L)T}\mathbf{z}^{(L-1)}\right) \\
y &= \sigma\left(\mathbf{w}_j^{(L+1)T}\mathbf{z}^{(L)}\right)
\end{aligned}
$$

- Neurons: $z_j^{(L)}$, latent variable.
- Activation function, $\sigma(\cdot)$: $Sign(\cdot)$, Identity function, logit function and so on.

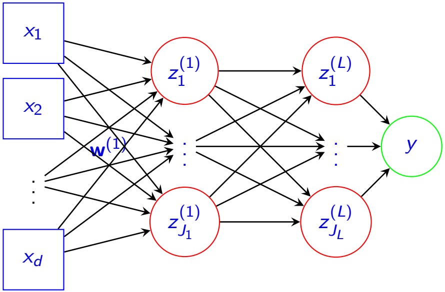


From MLP to Artificial Neural Network

- The general MLP can be represented as

$$
\begin{aligned}
z_j^{(1)} &= \sigma\left(\mathbf{w}_j^{(1)T}\mathbf{x}\right)\\
&\vdots\\
z_{j_k}^{(L)} &= \sigma\left(\mathbf{w}_{j_k}^{(L)T}\mathbf{z}^{(L-1)}\right)\\
y &= \sigma\left(\mathbf{w}_j^{(L+1)T}\mathbf{z}^{(L)}\right)
\end{aligned}
$$

- Neurons: $z_j^{(L)}$, latent variable.
- Activation function, $\sigma(\cdot)$: $Sign(\cdot)$, Identity function, logit function and so on.
- Output layer: y, objective function $\to$ regression or classification.

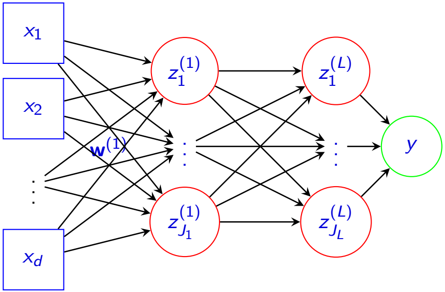


From MLP to Artificial Neural Network

- The general MLP can be represented as

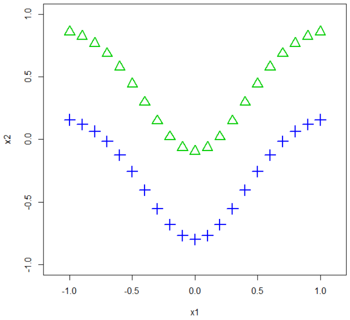
$$
\begin{aligned}
z_j^{(1)} &= \sigma\left(\mathbf{w}_j^{(1)T}\mathbf{x}\right)\\
&\vdots\\
z_{j_k}^{(L)} &= \sigma\left(\mathbf{w}_{j_k}^{(L)T}\mathbf{z}^{(L-1)}\right)\\
y &= \sigma\left(\mathbf{w}_j^{(L+1)T}\mathbf{z}^{(L)}\right)
\end{aligned}
$$

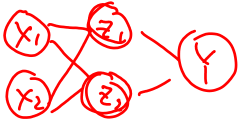
- Neurons: $z_j^{(L)}$, latent variable.
- Activation function, $\sigma(\cdot)$: $Sign(\cdot)$, Identity function, logit function and so on.
- Output layer: y, objective function $\rightarrow$ regression or classification.



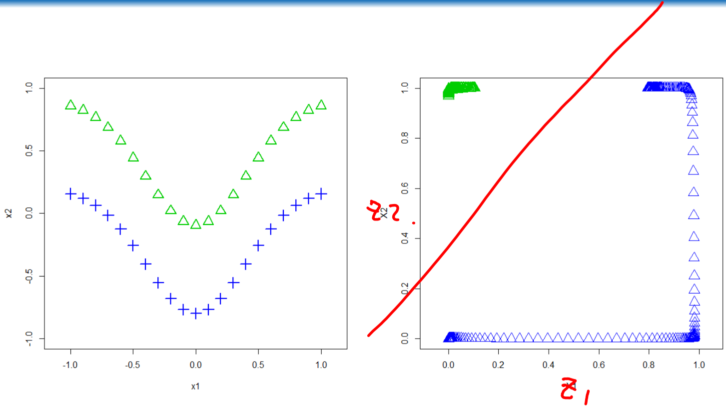
Feed-foward Neural Network

ANN, an integrated learning process

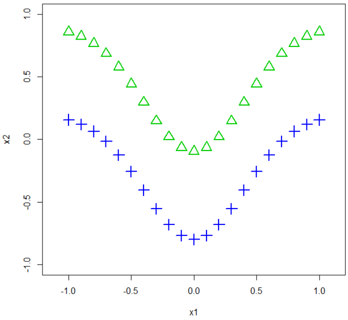


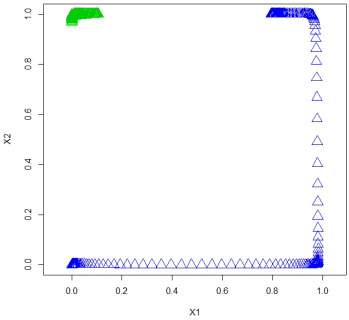


ANN, an integrated learning process



ANN, an integrated learning process





- The last layer of neurons can be viewed as a set of extracted features from the raw data.
- ANN can be viewed as an integrated learning process.