Support Vector Machine (SVM)

Machine Learning

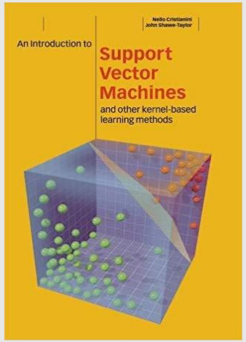
Prof. Sandra Avila
Institute of Computing (IC/Unicamp)

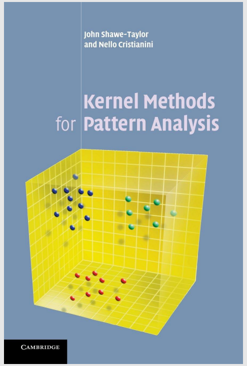
MC886, November 25, 2019

SVMs are among the best “off-the-shelf” supervised learning algorithm.

Andrew Ng

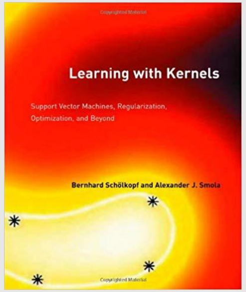
“**An Introduction to Support Vector Machines**: And Other Kernel-based Learning Methods”, Cristianini & Shawe-Taylor, 2000.



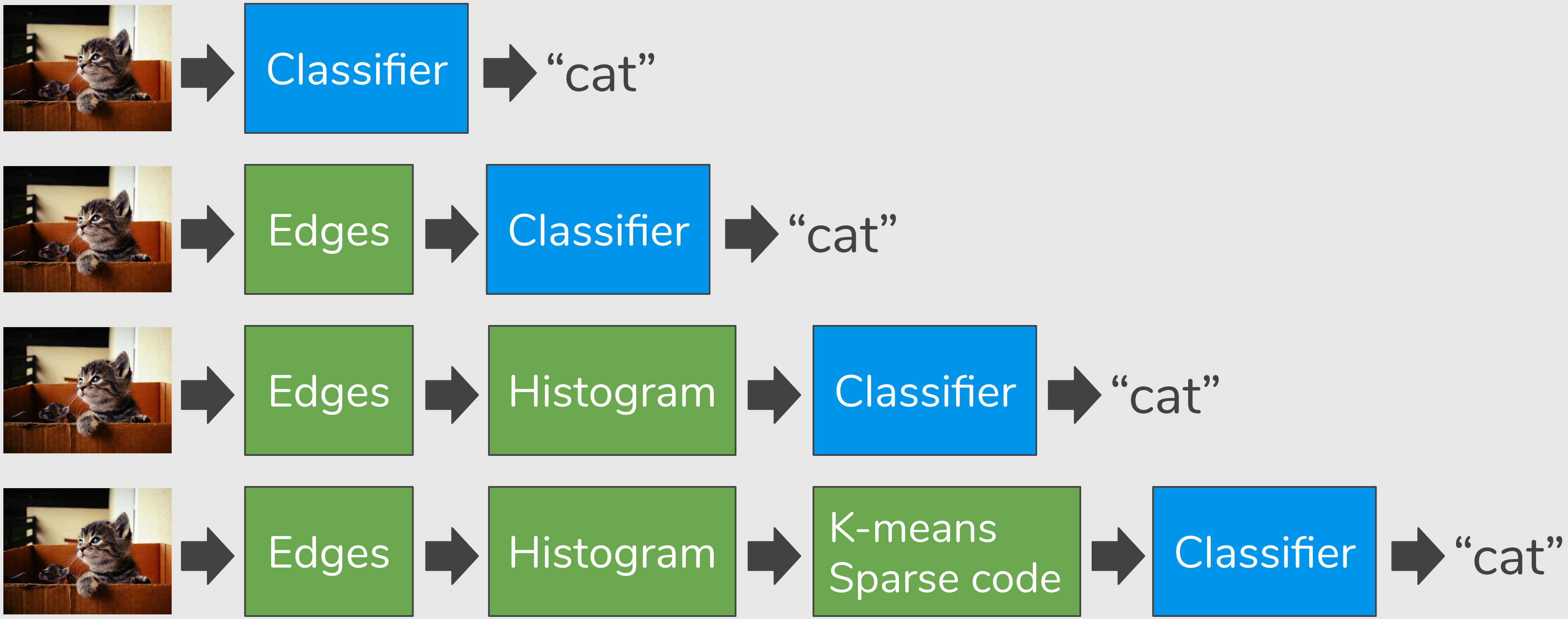


“**Kernel Methods for Pattern Analysis**”, Shawe-Taylor & Cristianini, 2004.

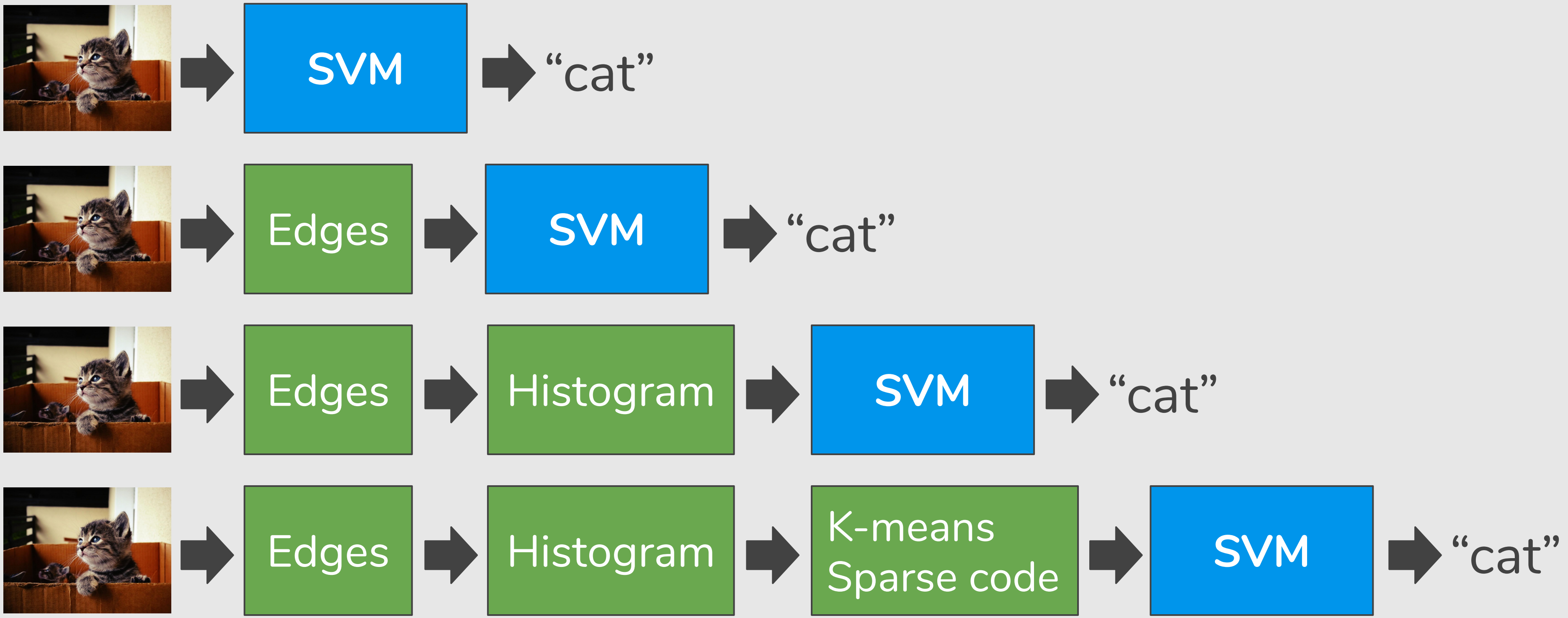
“**Learning with Kernels**: Support Vector Machines, Regularization, Optimization, and Beyond”, Scholkopf & Smola, 2001.



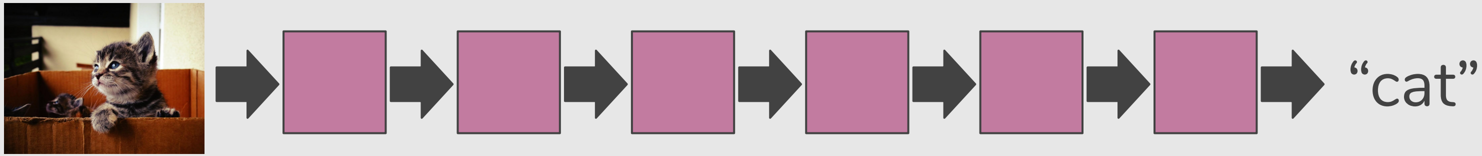
Traditional Recognition



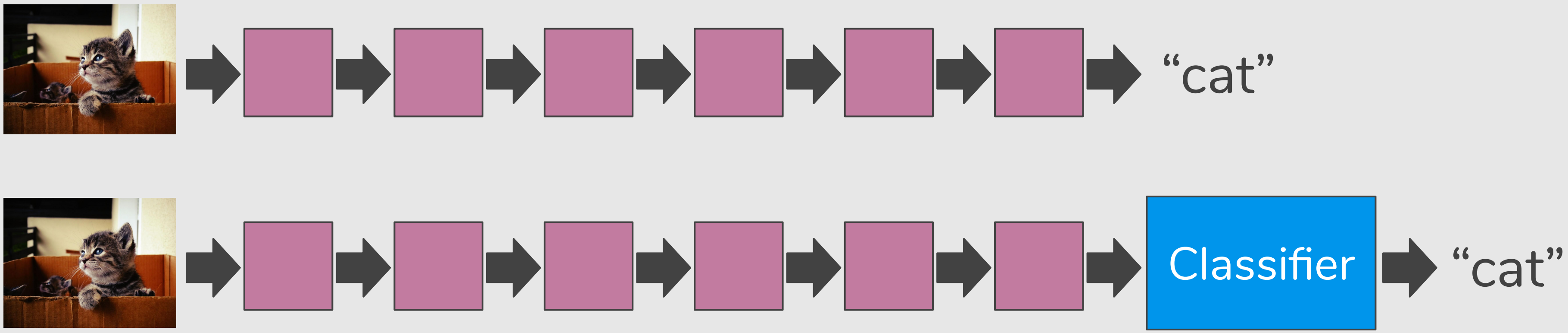
Traditional Recognition



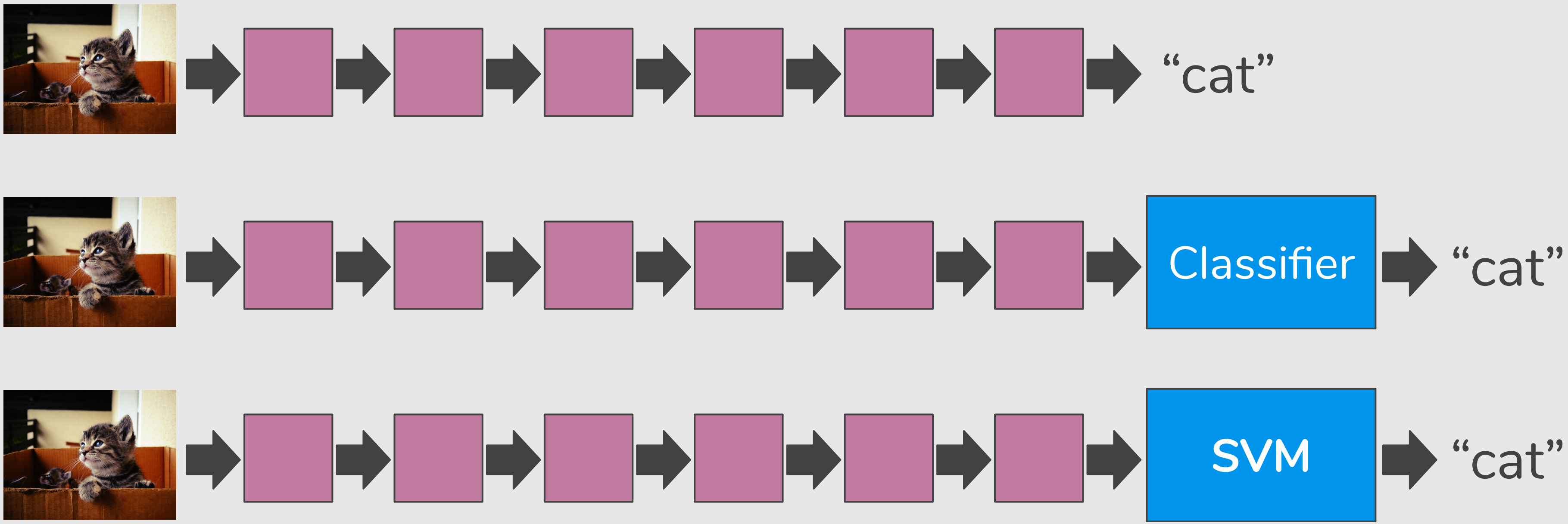
Deep Learning



Deep Learning

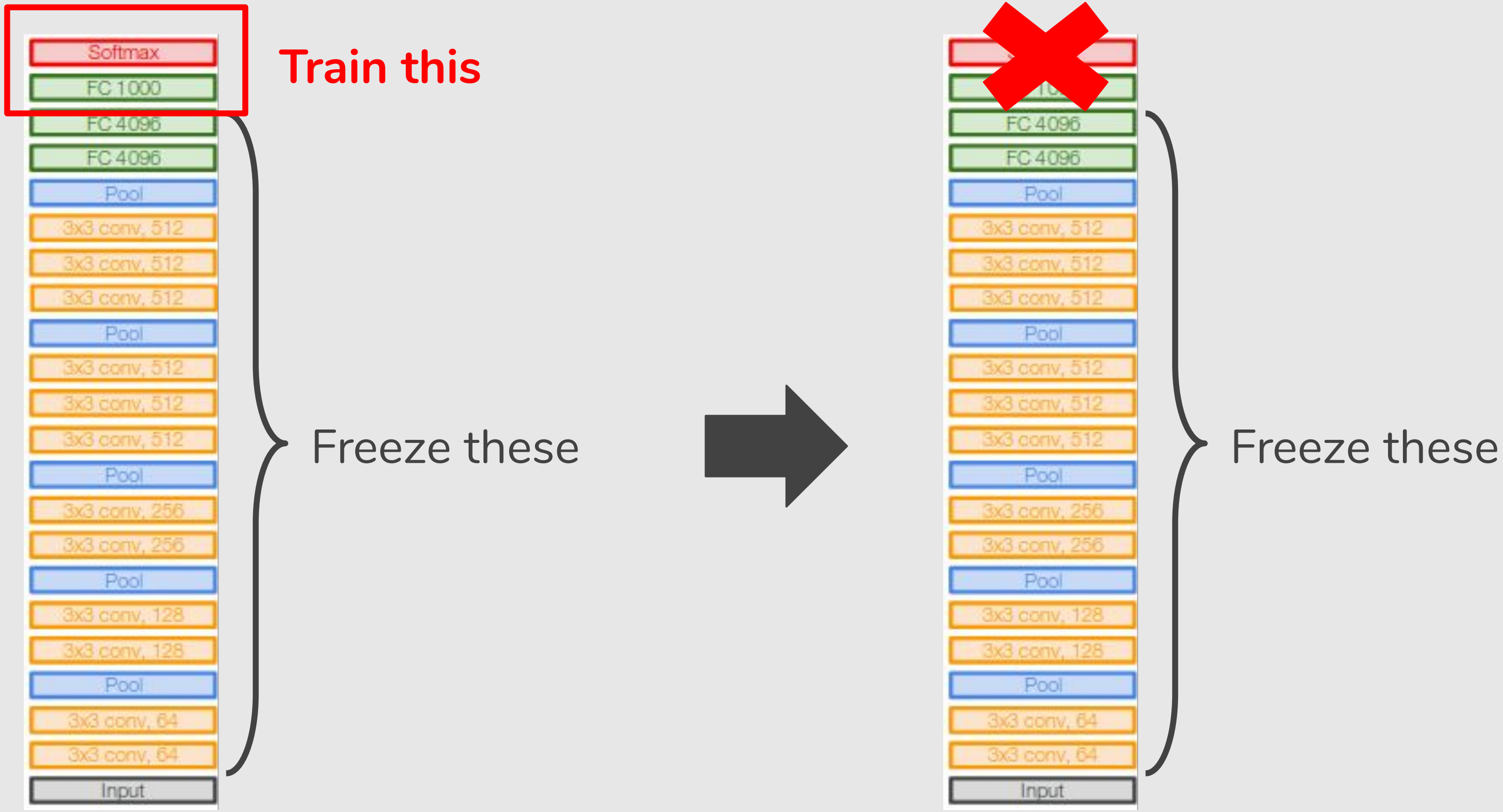


Deep Learning

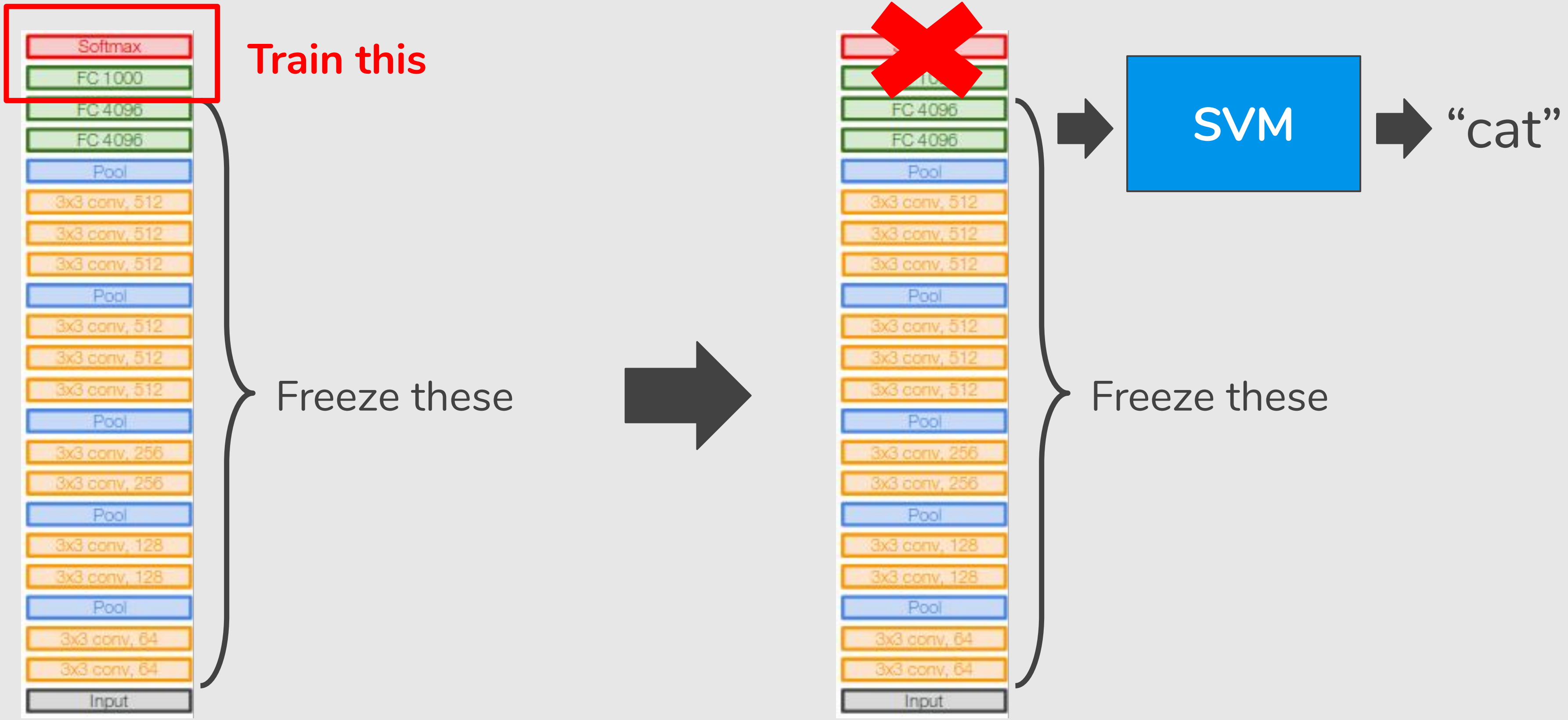


Transfer Learning

Transfer Learning



Transfer Learning

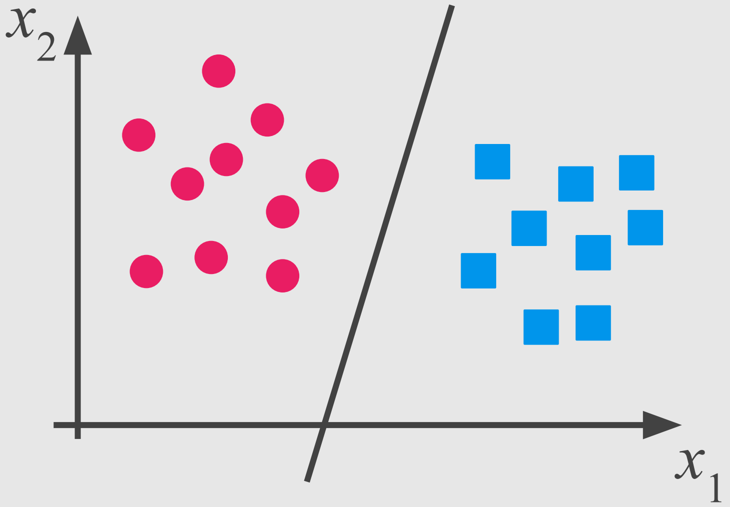


What is Support Vector Machine?

Support Vector Machine (SVM)

[Vapnik and Chervonenkis, 1964; Vapnik, 1982; Vapnik, 1995]

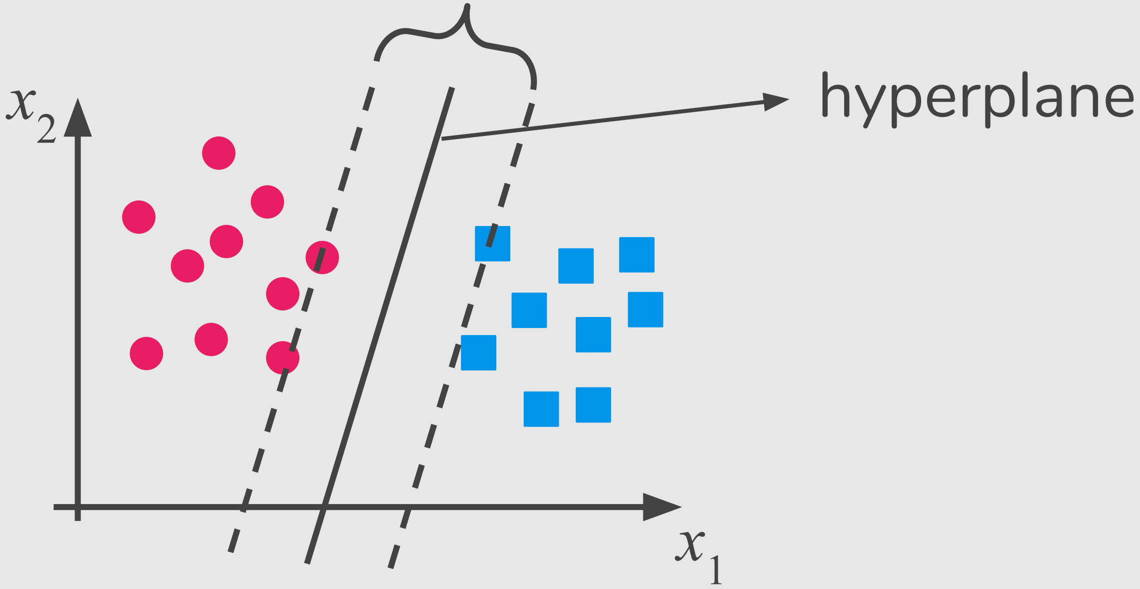
Idea of separating data with **a large "gap"**.



Support Vector Machine (SVM)

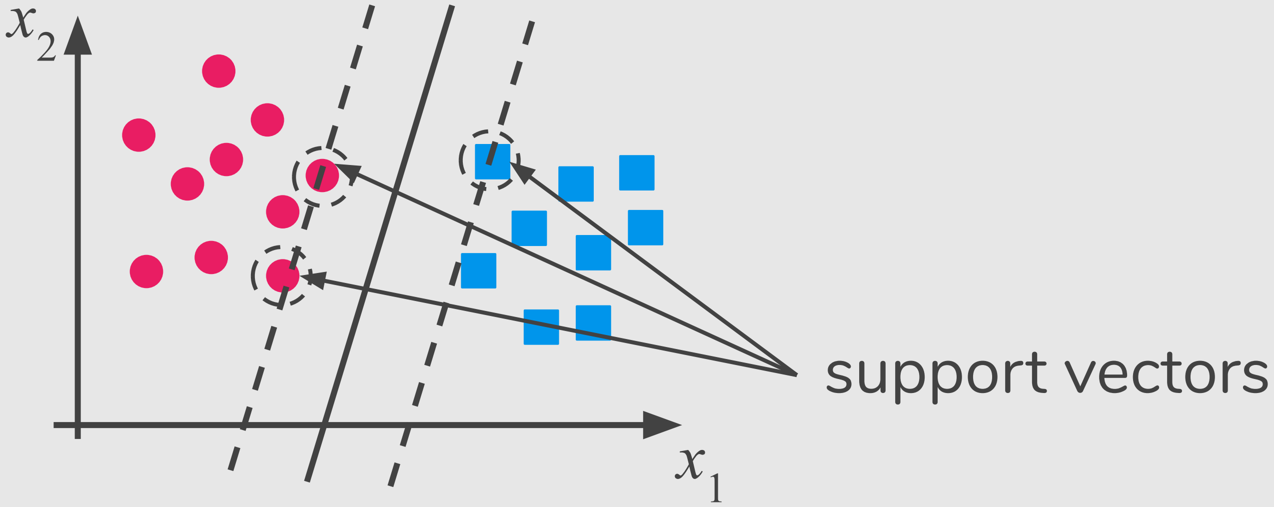
[Vapnik and Chervonenkis, 1964; Vapnik, 1982; Vapnik, 1995]

Idea of separating data with **a large "gap"**.



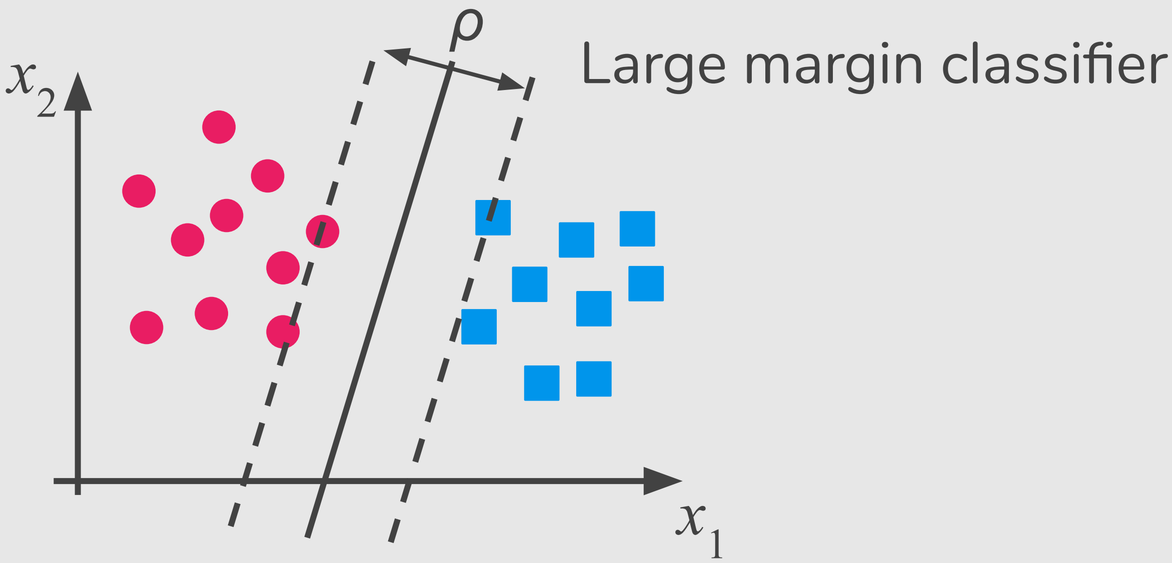
Support Vector Machine (SVM)
[Vapnik and Chervonenkis, 1964; Vapnik, 1982; Vapnik, 1995]

Examples closest to the hyperplane are support vectors.



Support Vector Machine (SVM)
[Vapnik and Chervonenkis, 1964; Vapnik, 1982; Vapnik, 1995]

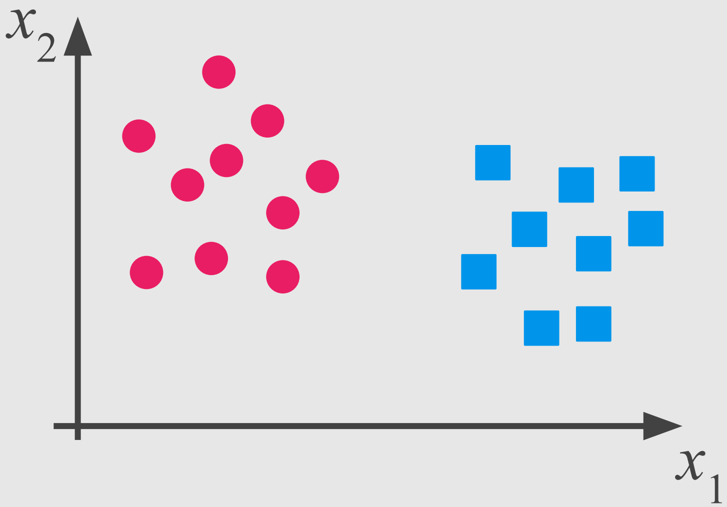
Margin ρ of the separator is the distance between support vectors.



How does SVM work?

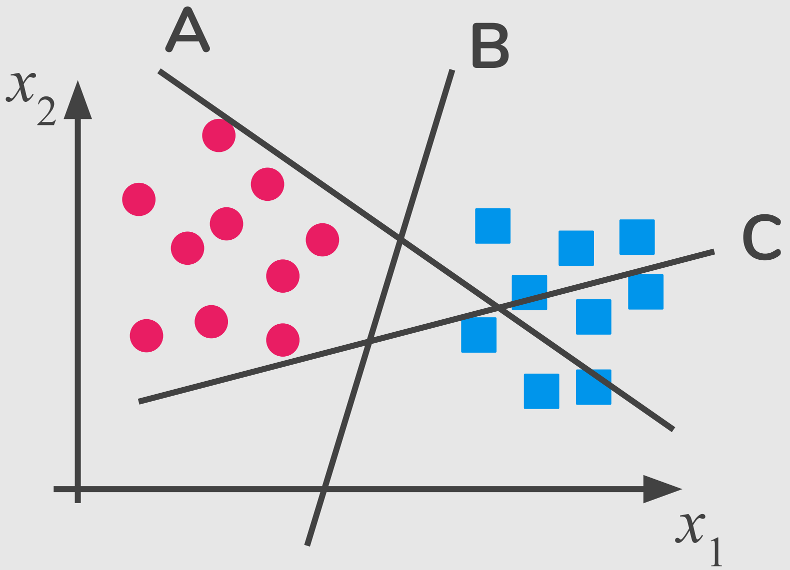
How can we identify the right hyperplane?

Scenario 1



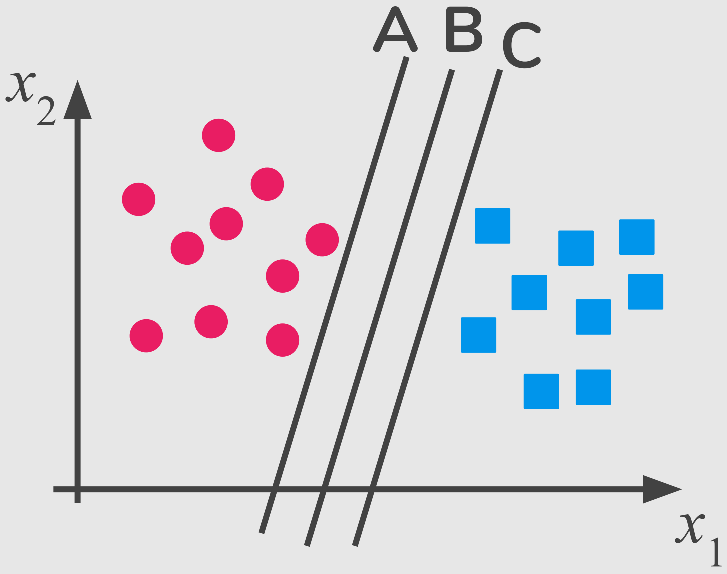
How can we identify the right hyperplane?

Scenario 1



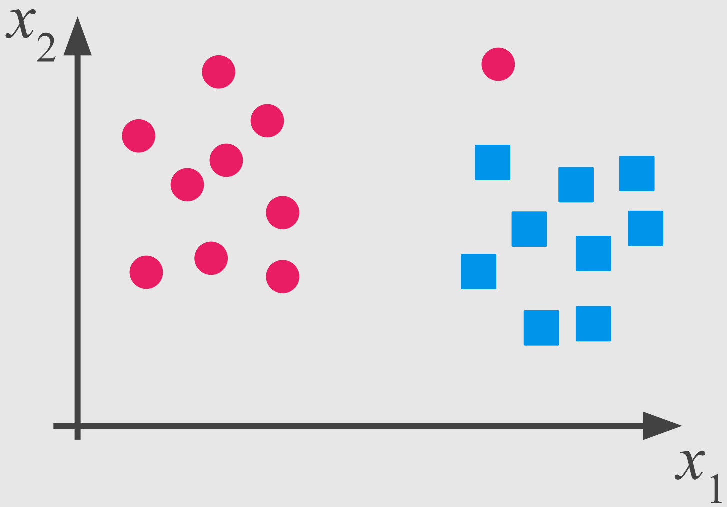
How can we identify the right hyperplane?

Scenario 2



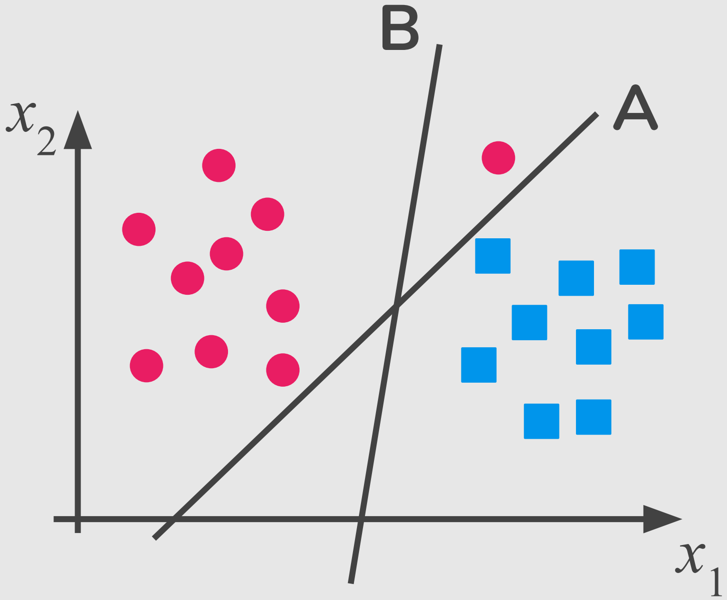
How can we identify the right hyperplane?

Scenario 3



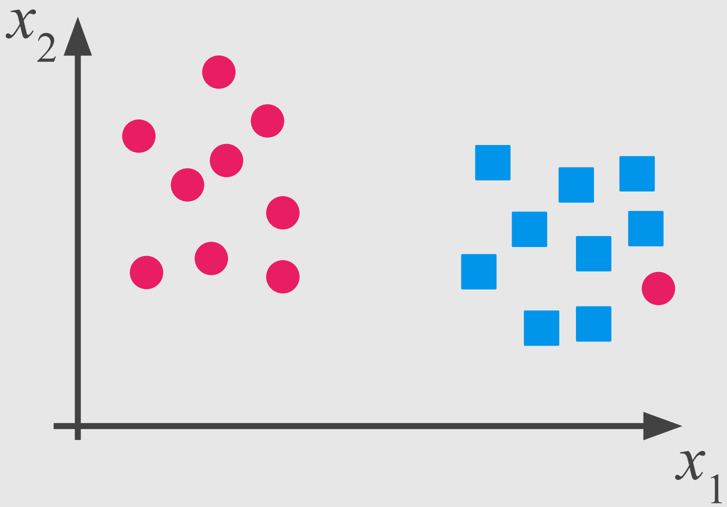
How can we identify the right hyperplane?

Scenario 3



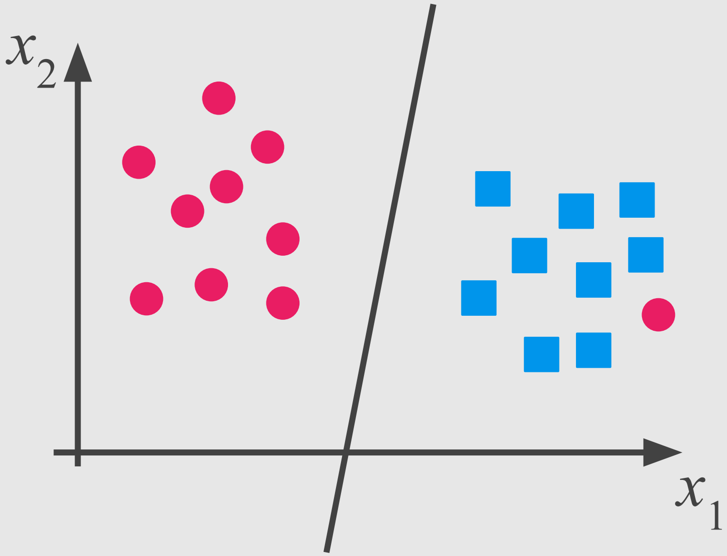
How can we identify the right hyperplane?

Scenario 4



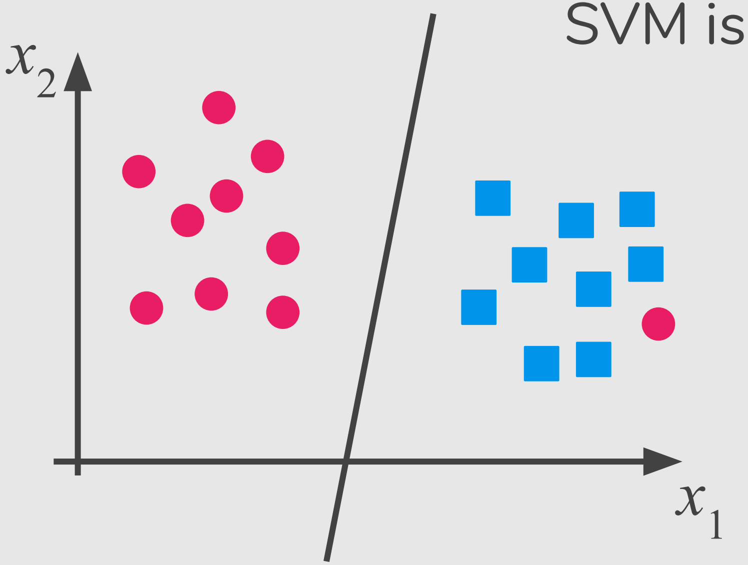
How can we identify the right hyperplane?

Scenario 4



How can we identify the right hyperplane?

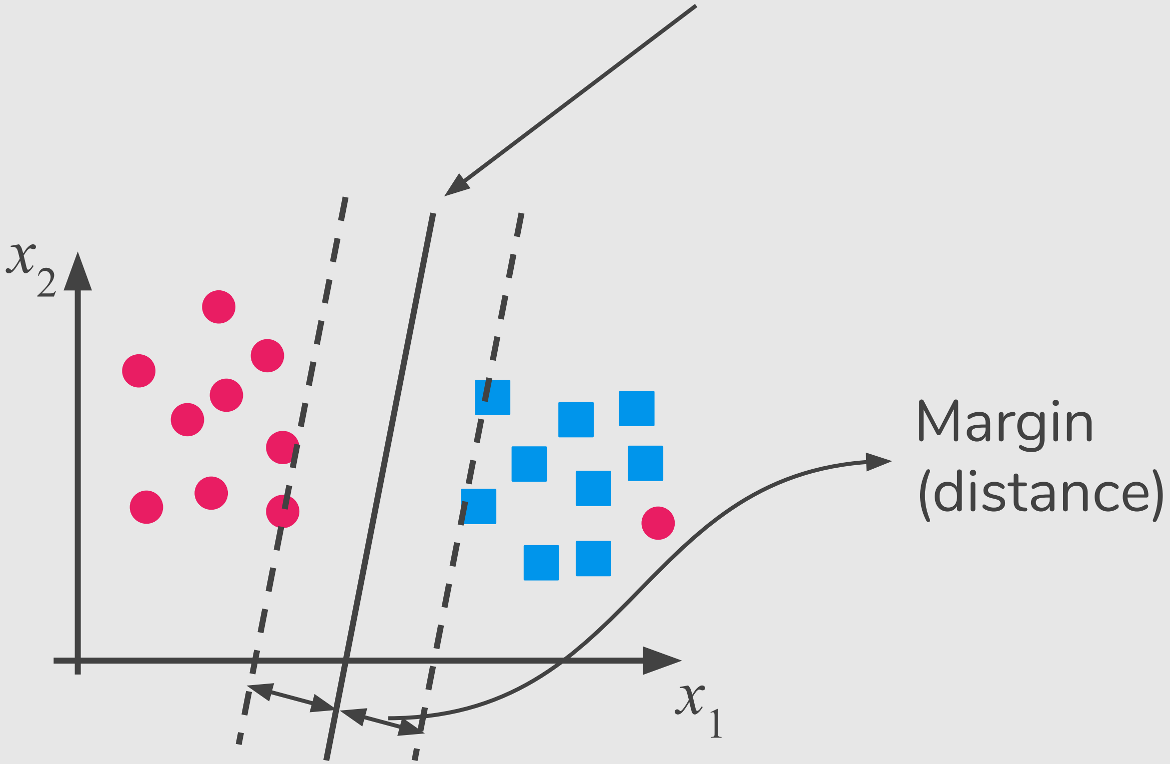
Scenario 4



SVM is robust to outliers

How can we identify the right hyperplane?

Scenario 4



SVM: Notation

We will be considering a **linear classifier for a binary classification** problem with labels y and features x.

SVM: Notation

We will be considering a **linear classifier for a binary classification** problem with labels y and features x.

- Class labels: $y \in \{-1,1\}$ (instead of $\{0,1\}$)
- Parameters: w, b (instead of vector θ)

SVM: Notation

We will be considering a **linear classifier for a binary classification** problem with labels y and features x.

- Class labels: $y \in \{-1,1\}$ (instead of $\{0,1\}$)
- Parameters: w, b (instead of vector θ)
- Classifier: $h_{w,b}(x) = g(w^T x + b)$
  - $g(z) = 1$ if $z \geq 0$, and $g(z) = -1$ otherwise

SVM: The Optimal Hyperplane

Given a training example $(x^{(i)}, y^{(i)})$, we define the margin of (w, b) with respect to the training example:

$$y^{(i)}(w^T x + b) \geq 1,\ i = \{1, ..., m\}.$$

SVM: The Optimal Hyperplane

Let $P(x^{(1)}, y^{(1)})$ be a point and l be a line defined by $ax + by + c = 0$. The distance d from P to l is defined by:

$$d(l,P) = \frac{|ax^{(1)} + by^{(1)} + c|}{\sqrt{a^2 + b^2}}$$

SVM: The Optimal Hyperplane

Let $P(x^{(1)}, y^{(1)})$ be a point and l be a line defined by $ax + by + c = 0$. The distance d from P to l is defined by:

$$d(l,P) = \frac{|ax^{(1)} + by^{(1)} + c|}{\sqrt{a^2 + b^2}}$$

$$d(w,b,x) = \frac{|w^T x + b|}{||w||}$$

SVM: The Optimal Hyperplane

$$d(w,b,x) = \frac{|w^T x + b|}{||w||}$$

$$\min_{w,b} \ \tfrac{1}{2}||w||^2$$

$$\text{s.t. } y^{(i)}(w^T x + b) \geq 1,\ i = \{1, \ldots, m\}$$

SVM: The Optimal Hyperplane

$$d(w,b,x) = \frac{|w^T x + b|}{||w||}$$

http://cs229.stanford.edu/notes/cs229-notes3.pdf

$$\min_{w,b} \tfrac{1}{2}||w||^2$$

$$\text{s.t. } y^{(i)}(w^T x + b) \geq 1,\ i = \{1, \ldots, m\}$$

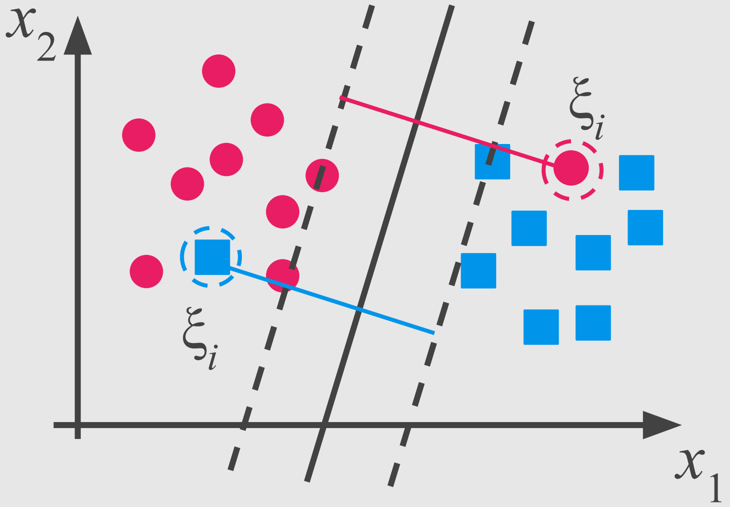
Need to optimize a quadratic function subject to linear constraints.

Soft Margin Classification

What if the training set is not linearly separable?

Soft Margin Classification

Slack variables ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called **soft**.



Soft Margin Classification

Modified formulation incorporates slack variables:

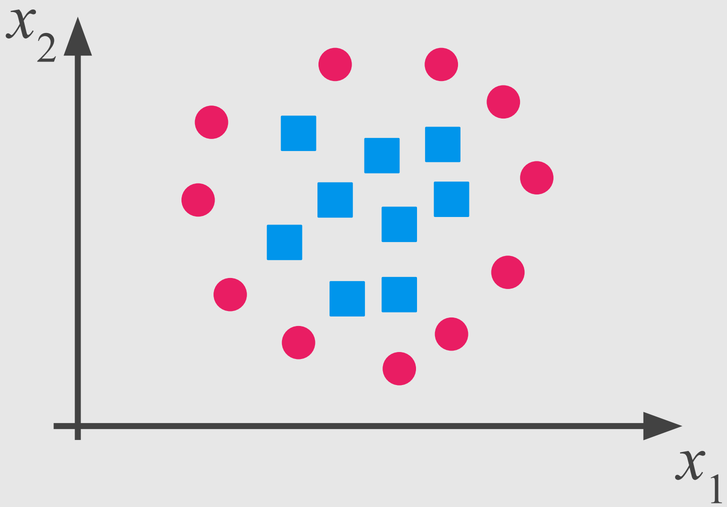
$$\min_{w,b,\xi}\ \tfrac{1}{2}||w||^2 + C\Sigma\xi_i$$
$$\text{s.t. } y_i(w^Tx + b) \geq 1 - \xi_i,\ \xi_i \geq 0,\ i = \{1, \ldots, m\}$$

Parameter C can be viewed as a way to control overfitting:

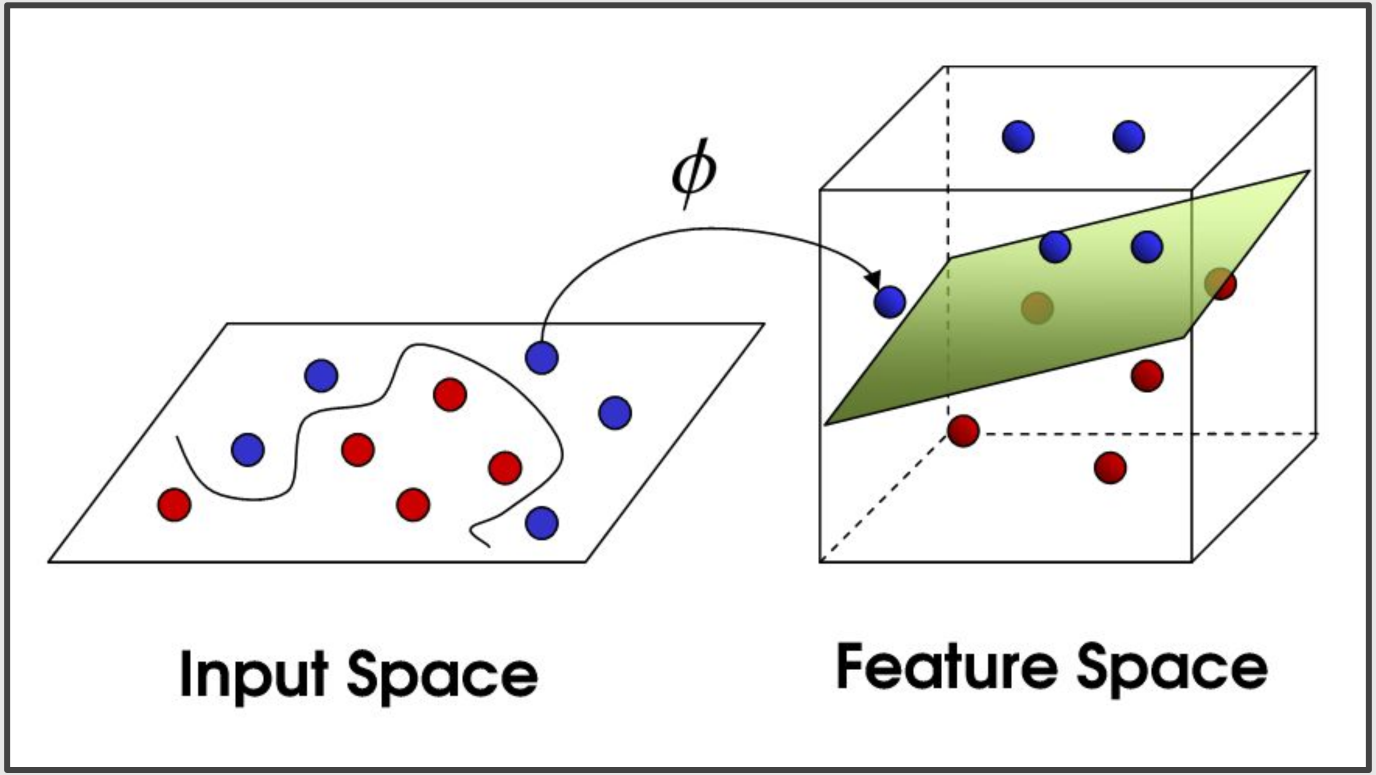
it "trades off" the relative importance of maximizing the margin and fitting the training data.

How can we identify the right hyperplane?

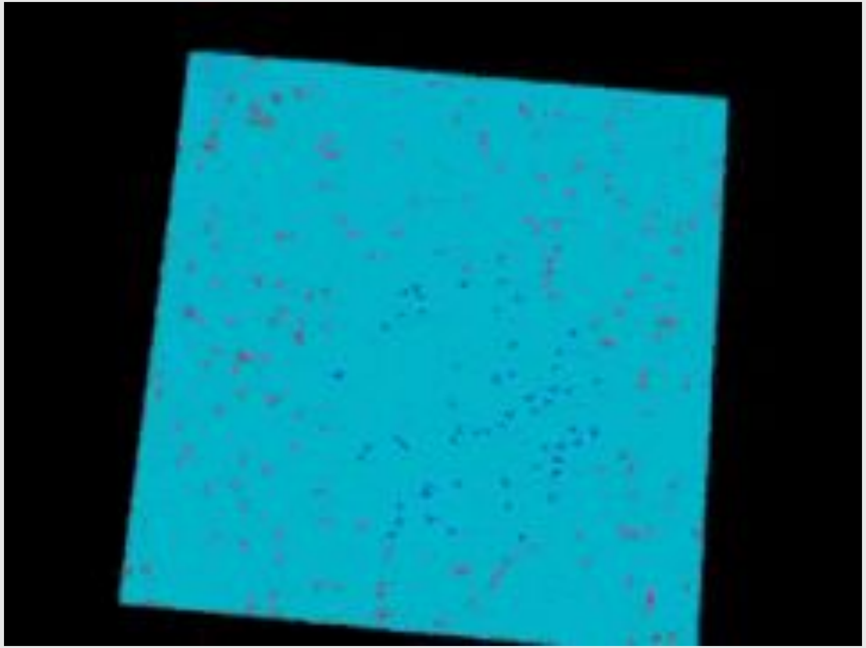
Scenario 5



Kernel Trick



Kernel Trick



Kernel Trick

- Linear SVM: $x_i \cdot x_j$

Kernel Trick

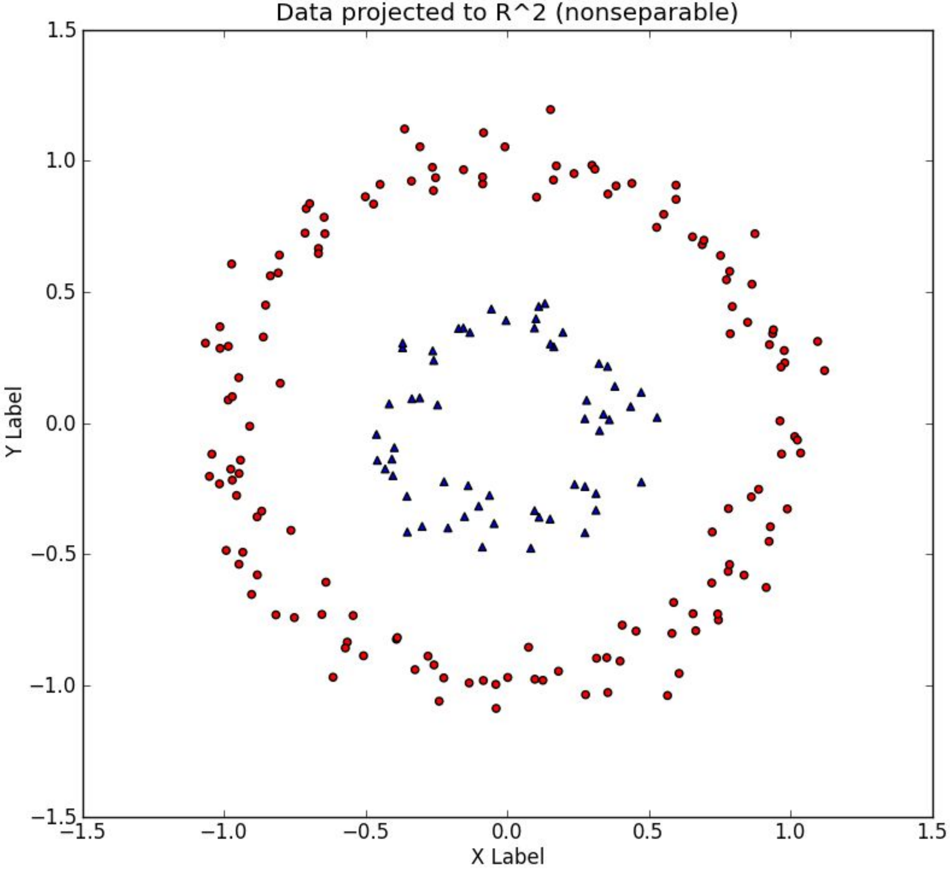
- Linear SVM: $x_i \cdot x_j$
- Nonlinear SVM: $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$, feature mapping ϕ

Kernel Trick

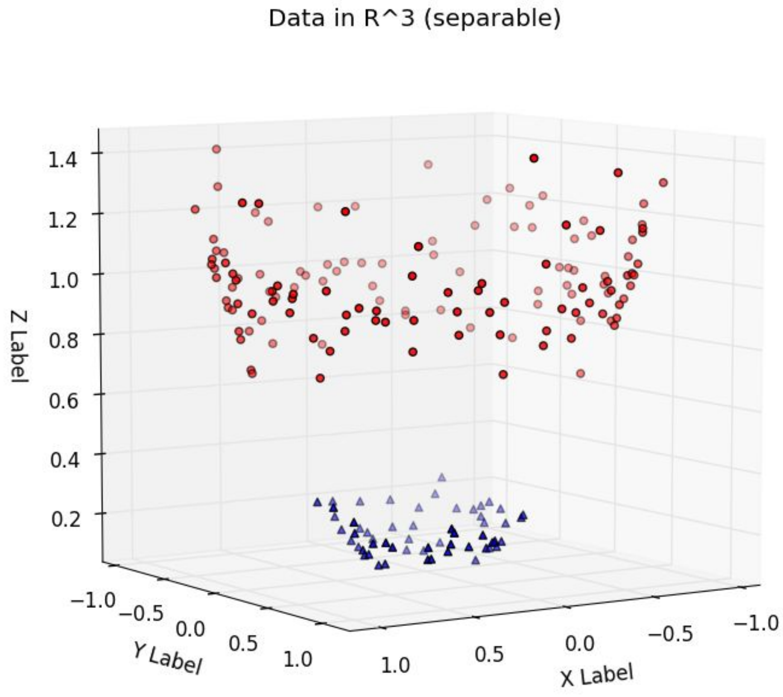
- Linear SVM: $x_i \cdot x_j$
- Nonlinear SVM: $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$, feature mapping ϕ
- Kernel matrix $K_{ij} = K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j) = \phi(x_j) \cdot \phi(x_i) = K_{ji}$

Kernel Trick

- Linear SVM: $x_i \cdot x_j$
- Nonlinear SVM: $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$, feature mapping ϕ
- Kernel matrix $K_{ij} = K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j) = \phi(x_j) \cdot \phi(x_i) = K_{ji}$
- Radial Basis Function (RBF) kernel: $\exp(-\lambda||x_i - x_j||^2)$
- Gaussian kernel: $K(x_i, x_j) = \exp(-||x_i - x_j||^2/(2\sigma^2))$
- Polynomial kernel: $K(x_i, x_j) = (x_i \cdot x_j + 1)^{\mathrm{d}}$, d degree
- Chi-square kernel, histogram intersection kernel, string kernel, ….



Data projected to R^2 (nonseparable)
Y Label
X Label
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5



Data in R^3 (separable)
Z Label
1.4
1.2
1.0
0.8
0.6
0.4
0.2
Y Label
−1.0
−0.5
0.0
0.5
1.0
X Label
1.0
0.5
0.0
−0.5
−1.0

Important Parameters

Important parameters having higher impact on model performance, “kernel”, “gamma” and “C”.

Important Parameters

Important parameters having higher impact on model performance, “kernel”, “gamma” and “C”.

C: Penalty parameter C of the error term. It also controls the trade off between smooth decision boundary and classifying the training points correctly.

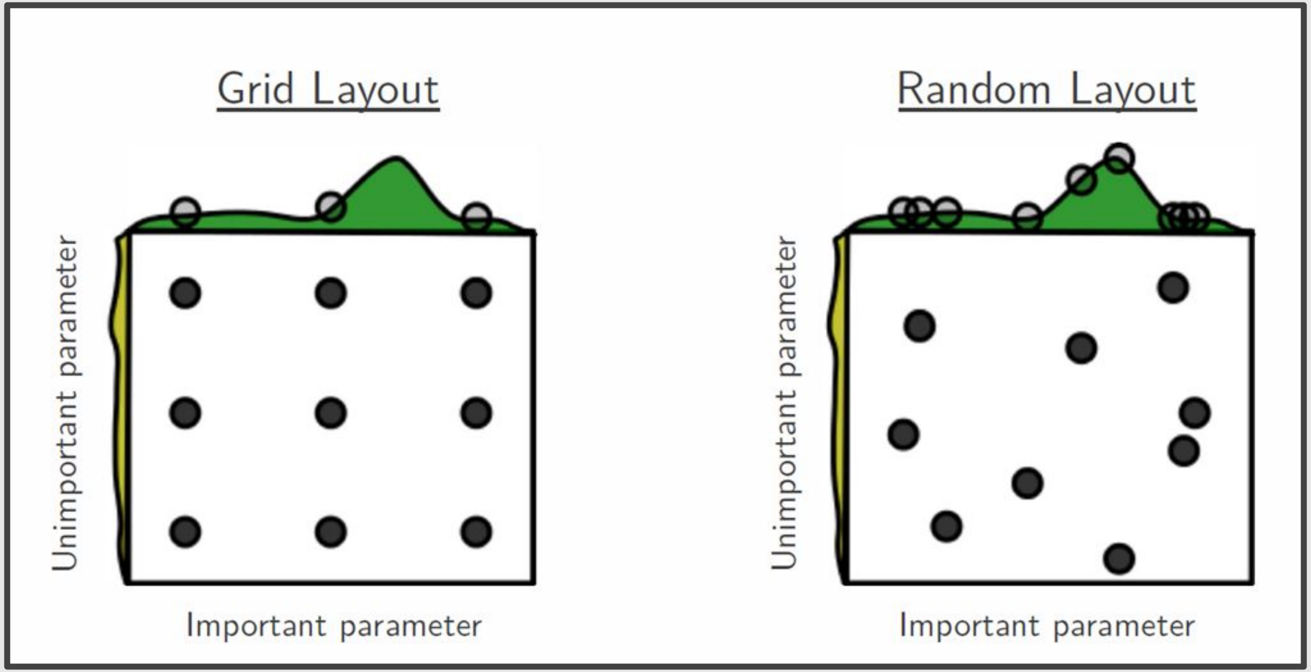
Important Parameters

Important parameters having higher impact on model performance, “kernel”, “gamma” and “C”.

C: Penalty parameter C of the error term. It also controls the trade off between smooth decision boundary and classifying the training points correctly.

The parameters can be tuned using grid-search.

Grid Search



“Random Search for Hyper-Parameter Optimization”: http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

Libraries

- Scikit-learn: https://scikit-learn.org/stable/modules/svm.html
- LIBSVM: https://www.csie.ntu.edu.tw/~cjlin/libsvm
- LIBLINEAR: https://www.csie.ntu.edu.tw/~cjlin/liblinear
- PmSVM: https://sites.google.com/site/wujx2001/home/power-mean-svm

References

Machine Learning Books

- Hands-On Machine Learning with Scikit-Learn and TensorFlow, Chap. 5
- Pattern Recognition and Machine Learning, Chap. 6 & 7

Machine Learning Courses

- https://www.coursera.org/learn/machine-learning, Week 7
- http://cs229.stanford.edu/syllabus.html,
  http://cs229.stanford.edu/notes/cs229-notes3.pdf

Random Forests

Machine Learning

Prof. Sandra Avila
Institute of Computing (IC/Unicamp)

MC886, November 25, 2019

Decision Tree

Decision Tree & Random Forest

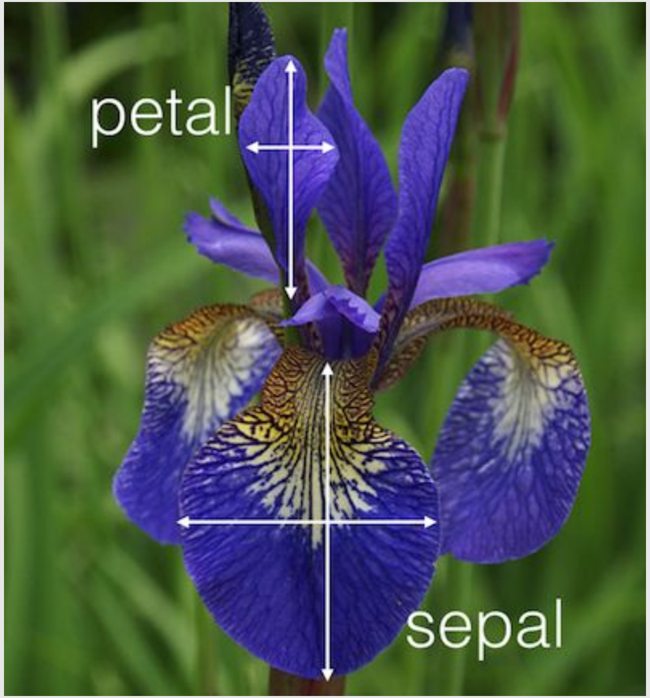
- **Decision Trees** are versatile Machine Learning algorithms that can perform both classification and regression tasks, and even multi-output tasks.

Decision Tree & Random Forest

- Decision Trees are versatile Machine Learning algorithms that can perform both classification and regression tasks, and even multi-output tasks.

- **Random Forest is an ensemble of Decision Trees**, generally trained using the Bagging method (or sometimes Pasting).

Decision Tree: Iris Dataset



http://sebastianraschka.com/Articles/2014_python_lda.html

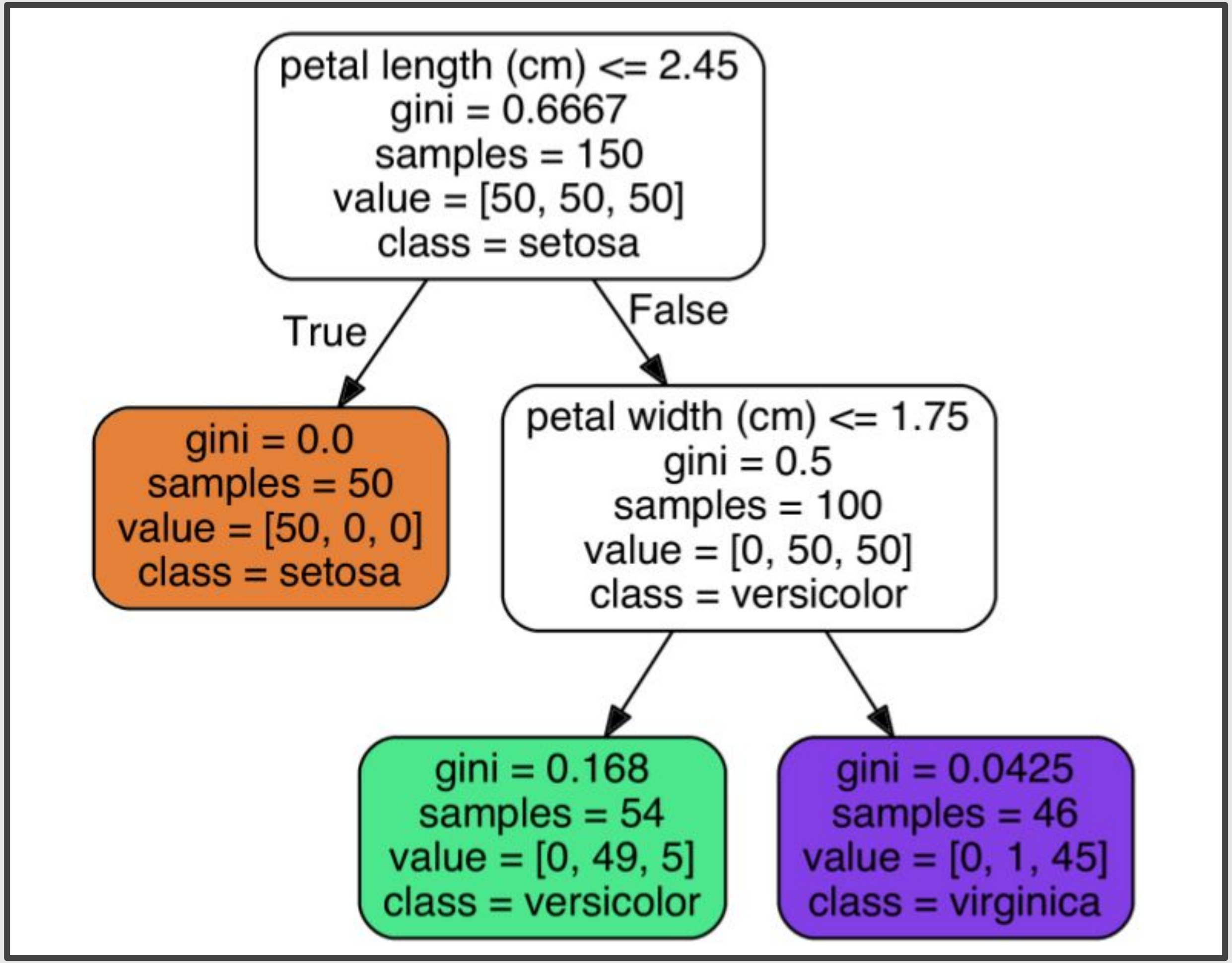
150 iris flowers from three different species.

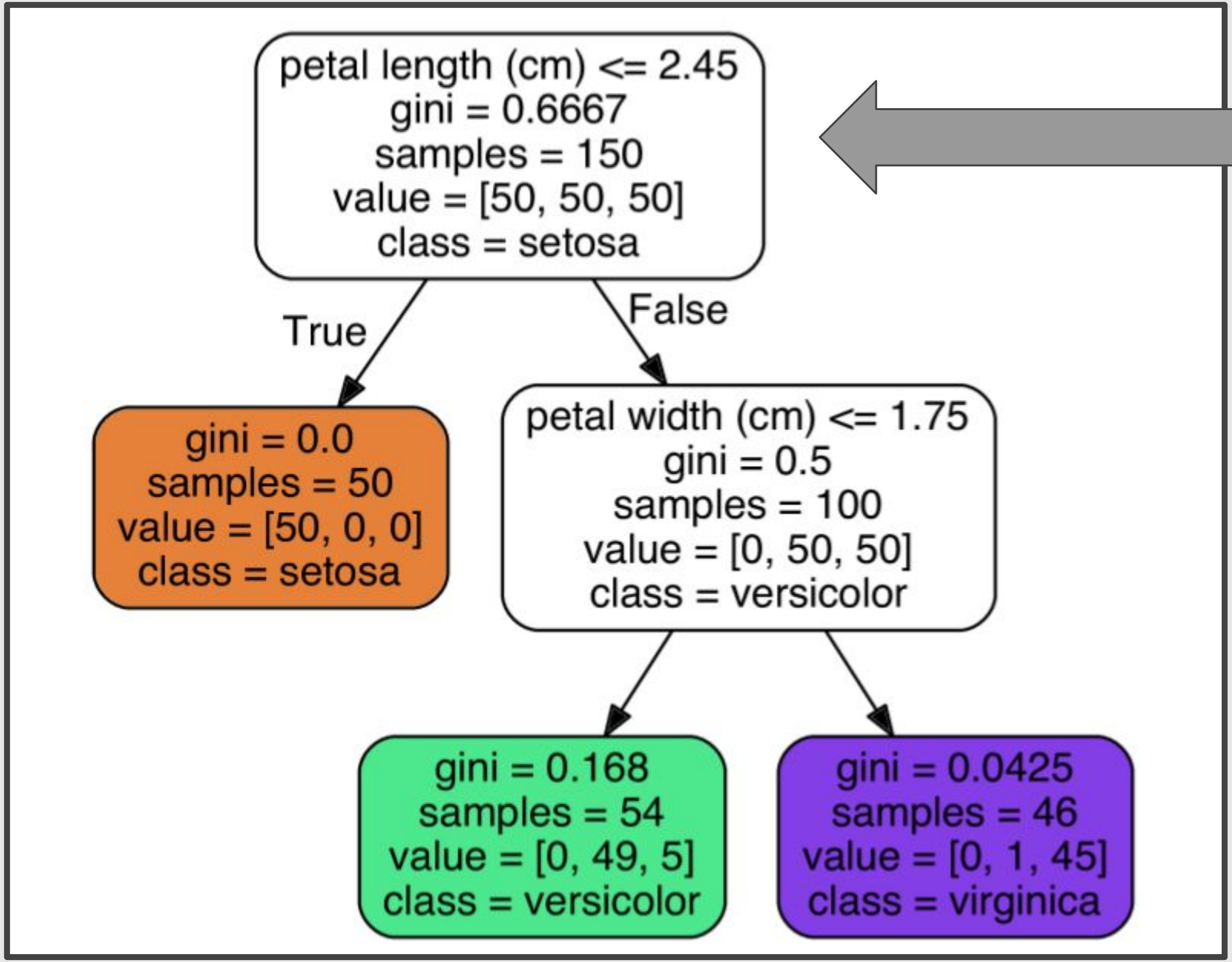
The three classes in the Iris dataset:

1. Iris-setosa (n=50)
2. Iris-versicolor (n=50)
3. Iris-virginica (n=50)

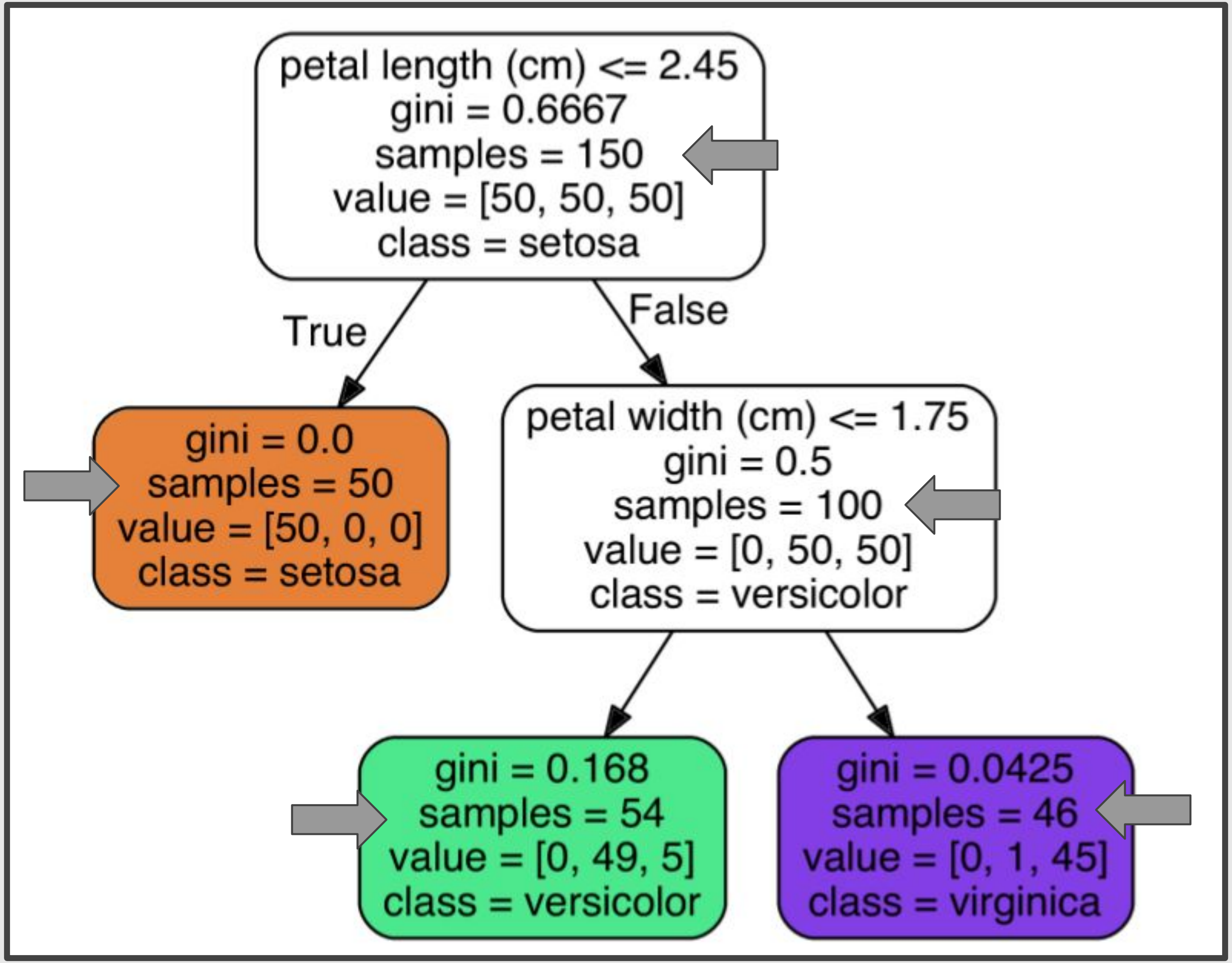
The four features of the Iris dataset:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm

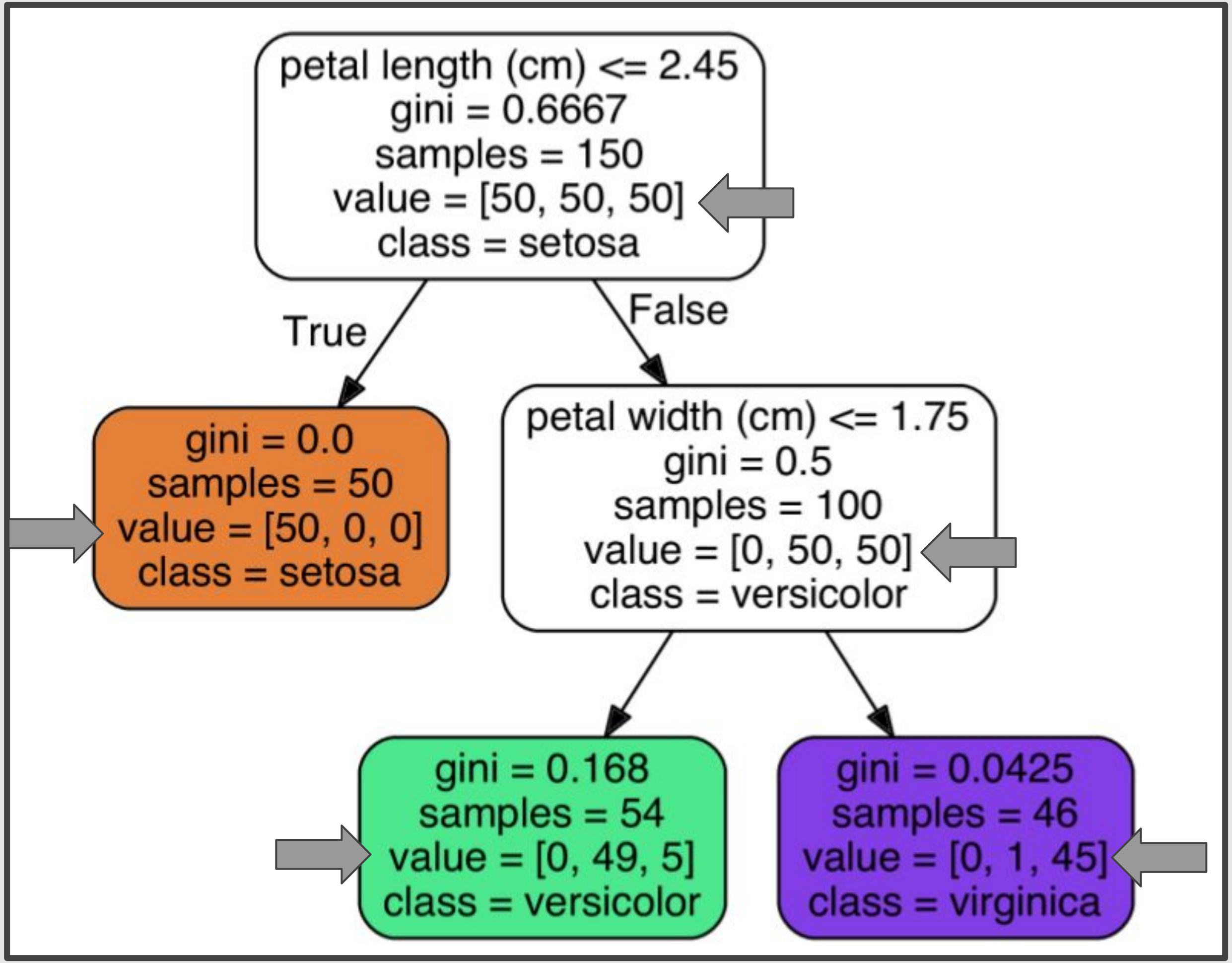

petal length (cm) <= 2.45
gini = 0.6667
samples = 150
value = [50, 50, 50]
class = setosa
True
False
gini = 0.0
samples = 50
value = [50, 0, 0]
class = setosa
petal width (cm) <= 1.75
gini = 0.5
samples = 100
value = [0, 50, 50]
class = versicolor
gini = 0.168
samples = 54
value = [0, 49, 5]
class = versicolor
gini = 0.0425
samples = 46
value = [0, 1, 45]
class = virginica



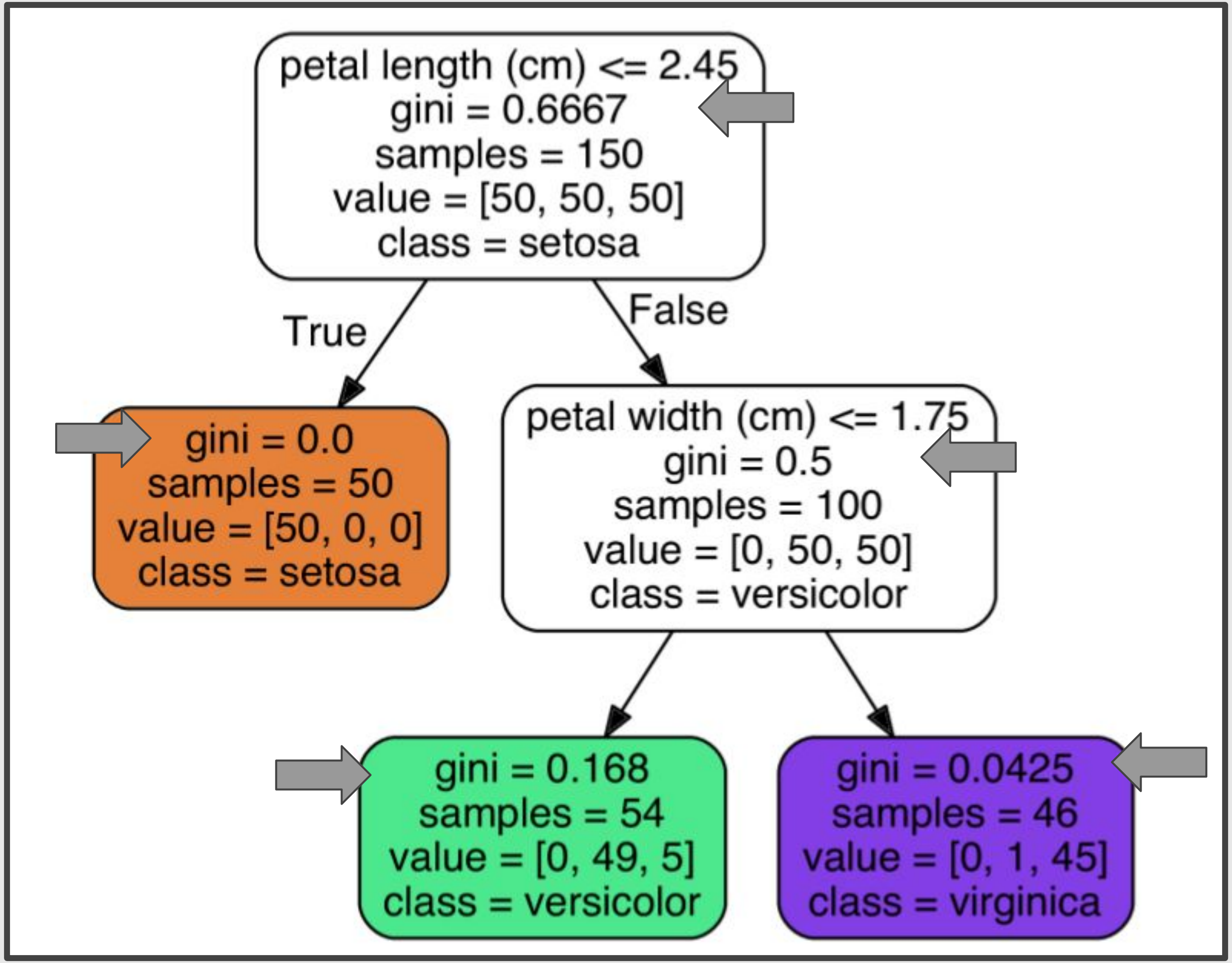
This node asks whether the flower's petal length is smaller than 2.45 cm



A **node's samples** attribute counts how many training instances it applies to.



A **node's value** attribute tells you how many training instances of each class this node applies to.



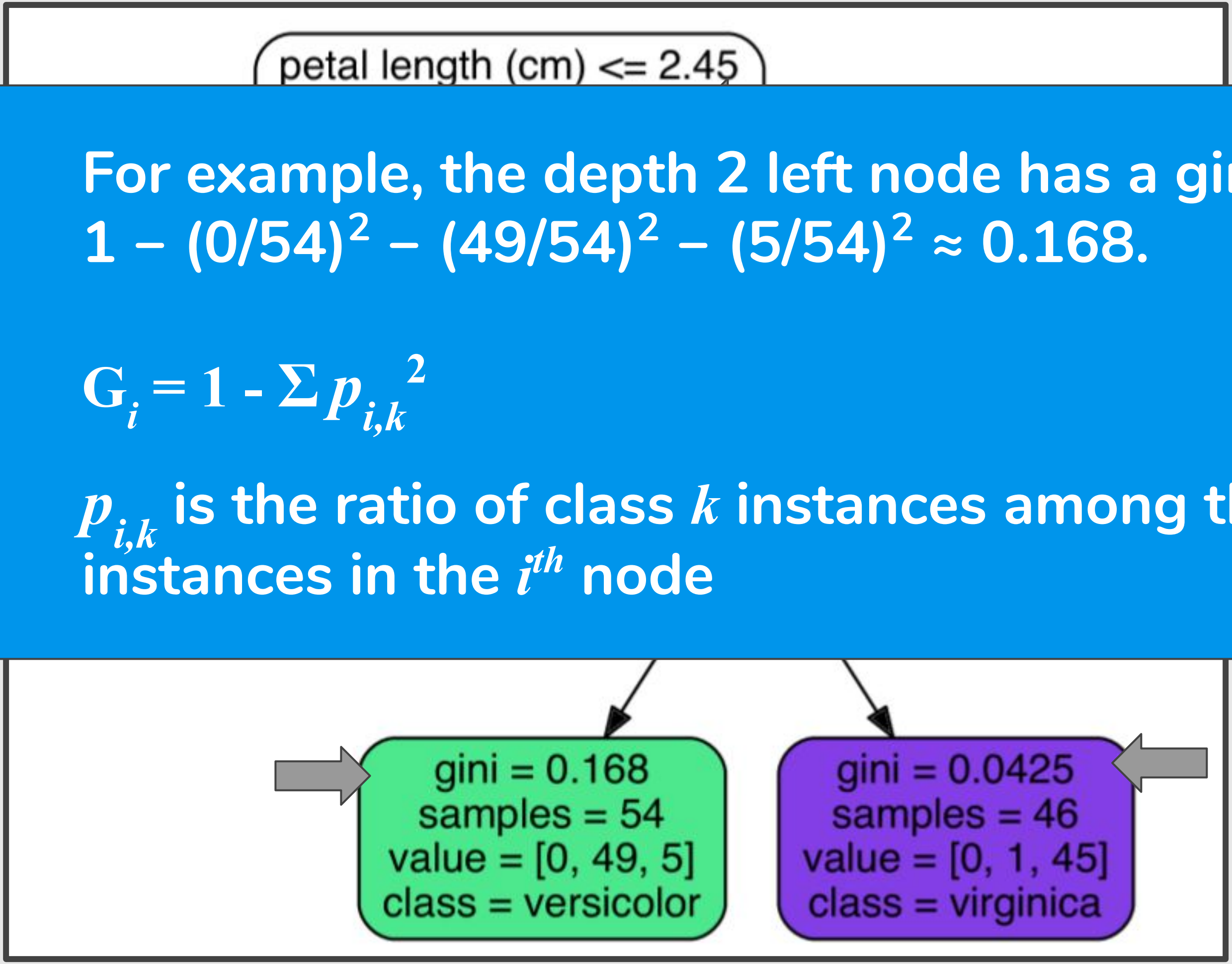
A **node's gini** attribute measures its impurity.

"pure" (gini=0): all training instances belong to the same class.

For example, the depth 2 left node has a gini score equal to $1 - (0/54)^2 - (49/54)^2 - (5/54)^2 \approx 0.168$.

$$G_i = 1 - \Sigma \, p_{i,k}{}^2$$

$p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node

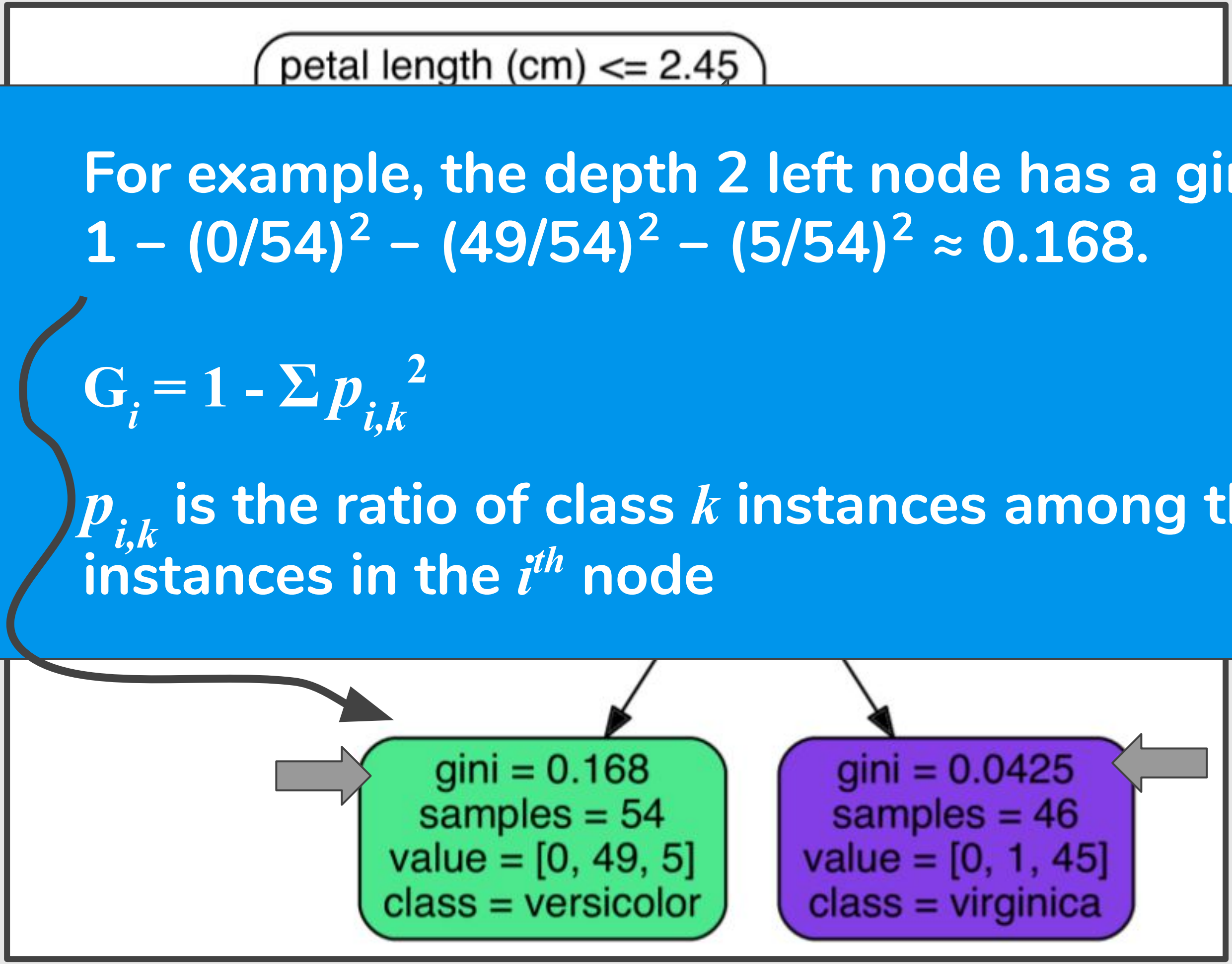


same class.

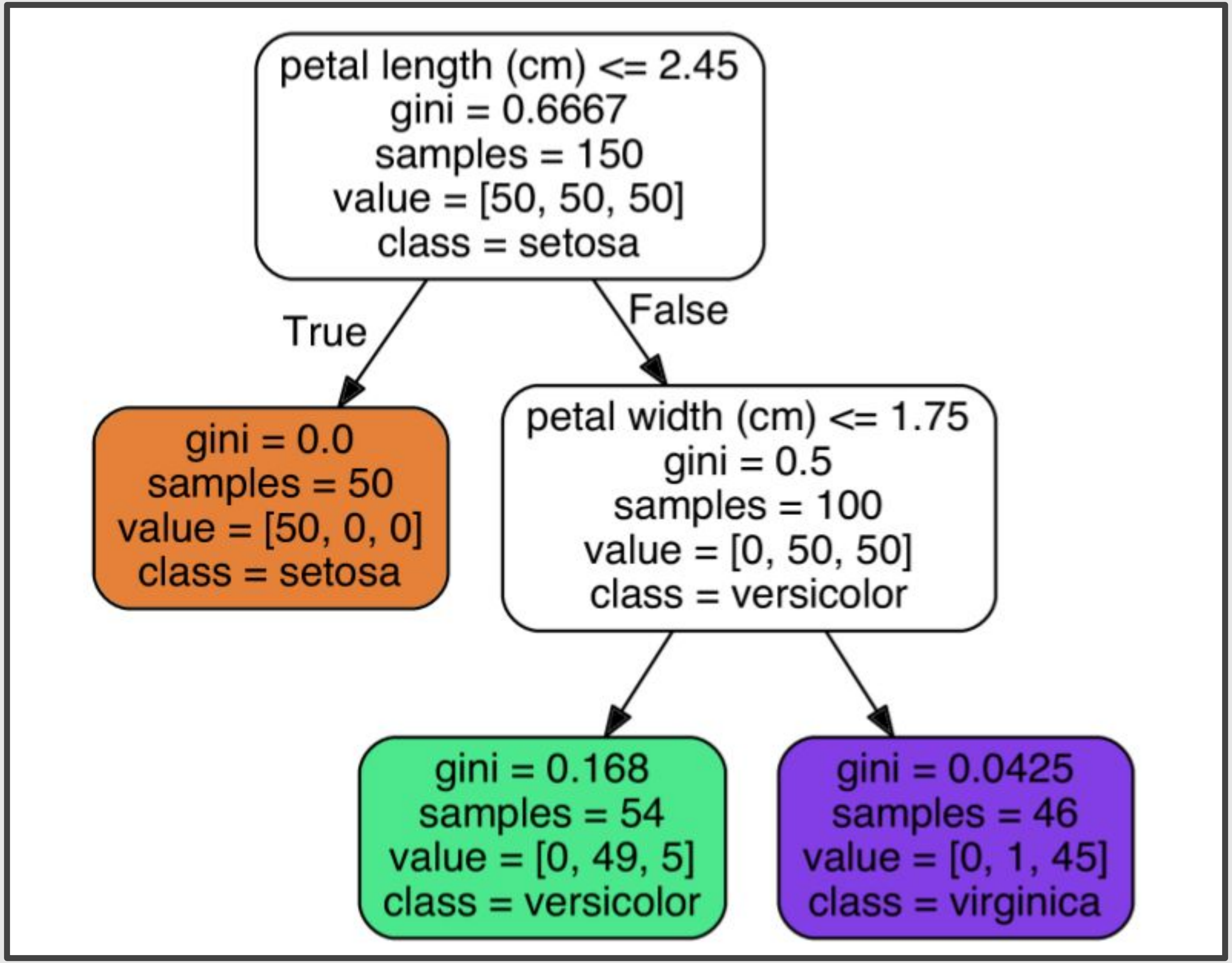
For example, the depth 2 left node has a gini score equal to $1 - (0/54)^2 - (49/54)^2 - (5/54)^2 \approx 0.168$.

$$G_i = 1 - \sum p_{i,k}^{\;2}$$

$p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node

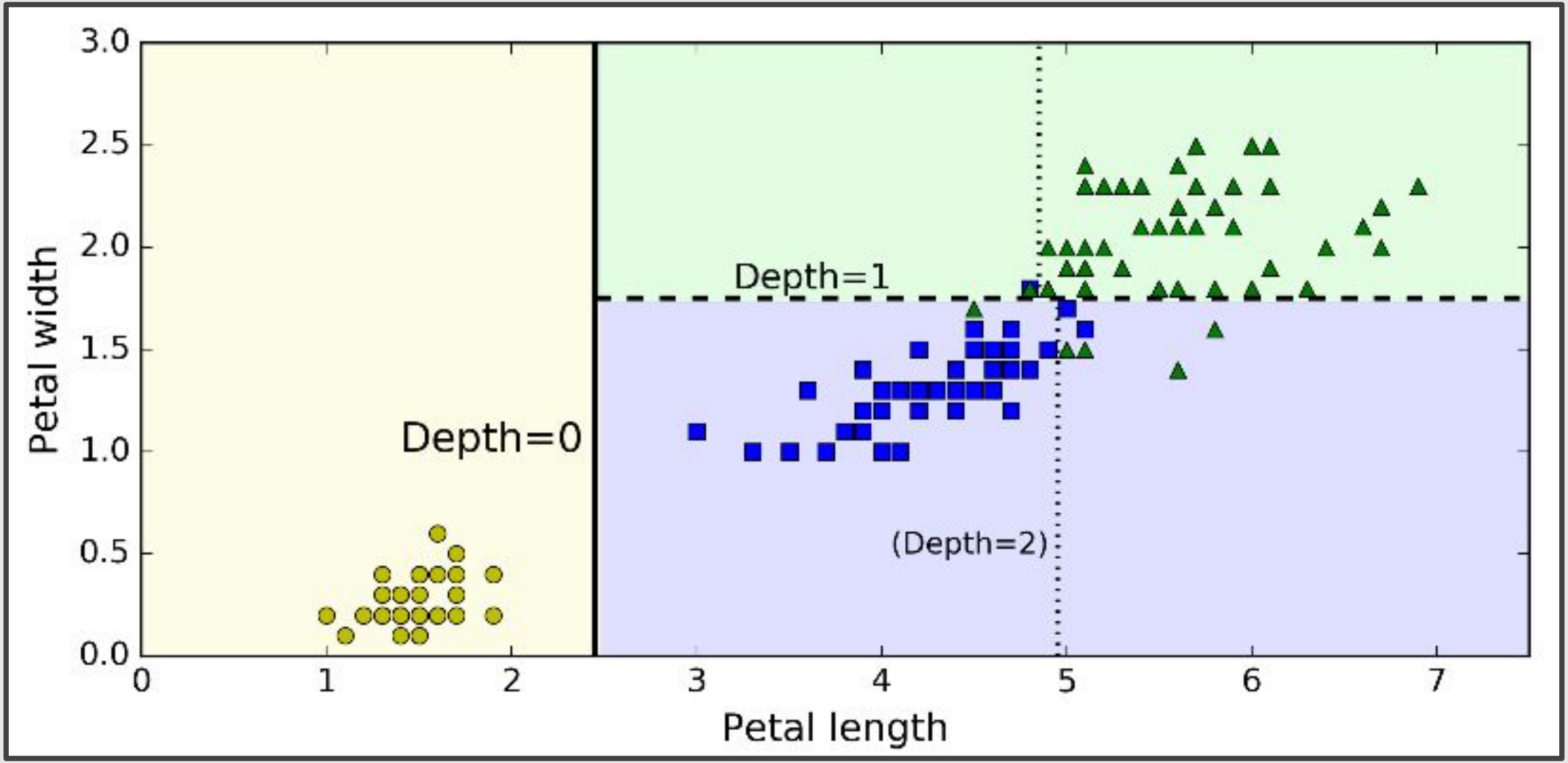


same class.



A **node's gini** attribute measures its impurity.

"pure" (gini=0): all training instances belong to the same class.



Depth=0
Depth=1
(Depth=2)
Petal width
Petal length
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
1
2
3
4
5
6
7

The CART Algorithm

- Classification And Regression Tree (CART) algorithm.

The CART Algorithm

- Classification And Regression Tree (CART) algorithm.
- The idea is really quite simple: the algorithm first splits the training set in two subsets using a single feature k and a threshold t_k (e.g. “petal length ≤ 2.45 cm”).

The CART Algorithm

- Classification And Regression Tree (CART) algorithm.
- The idea is really quite simple: the algorithm first splits the training set in two subsets using a single feature k and a threshold t_k (e.g. "petal length ≤ 2.45 cm").
- How does it choose k and t_k?

The CART Algorithm

- Classification And Regression Tree (CART) algorithm.
- The idea is really quite simple: the algorithm first splits the training set in two subsets using a single feature k and a threshold t_k (e.g. “petal length ≤ 2.45 cm”).
- How does it choose k and t_k?

  It searches for the pair (k, t_k) that produces the purest subsets (weighted by their size).

The CART Algorithm

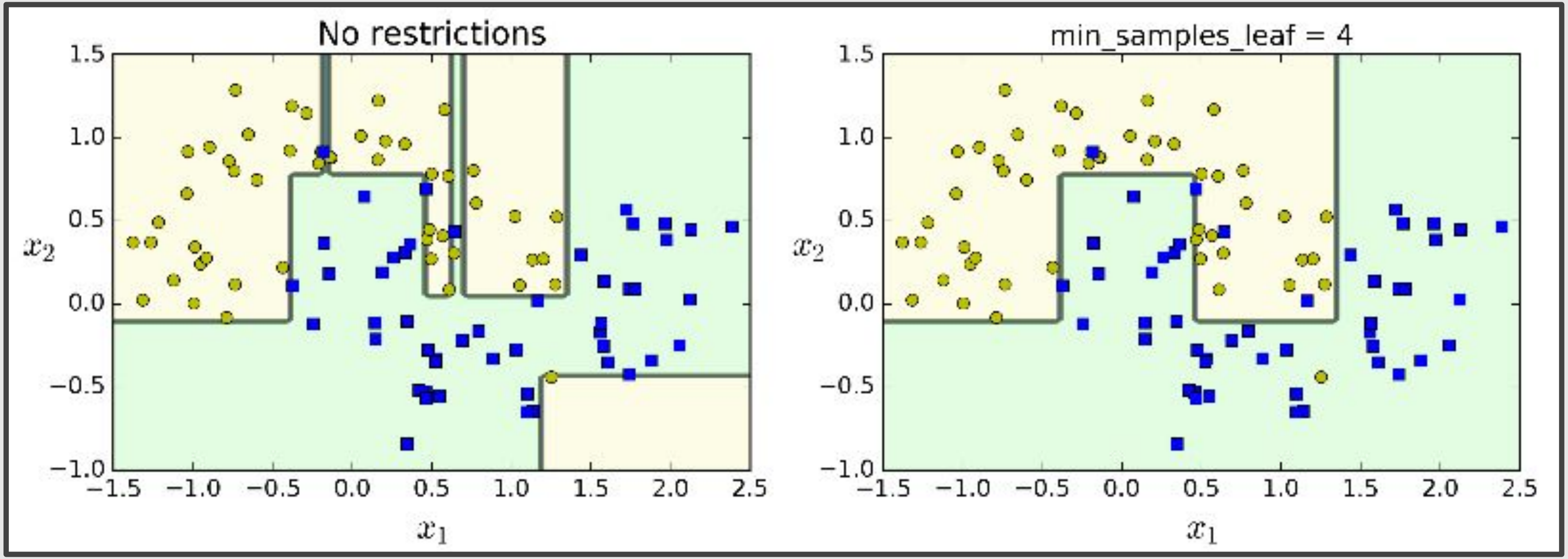
$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

$$\text{where } \begin{cases} G_{\text{left/right}} \text{ measures the impurity of the left/right subset,} \\ m_{\text{left/right}} \text{ is the number of instances in the left/right subset.} \end{cases}$$

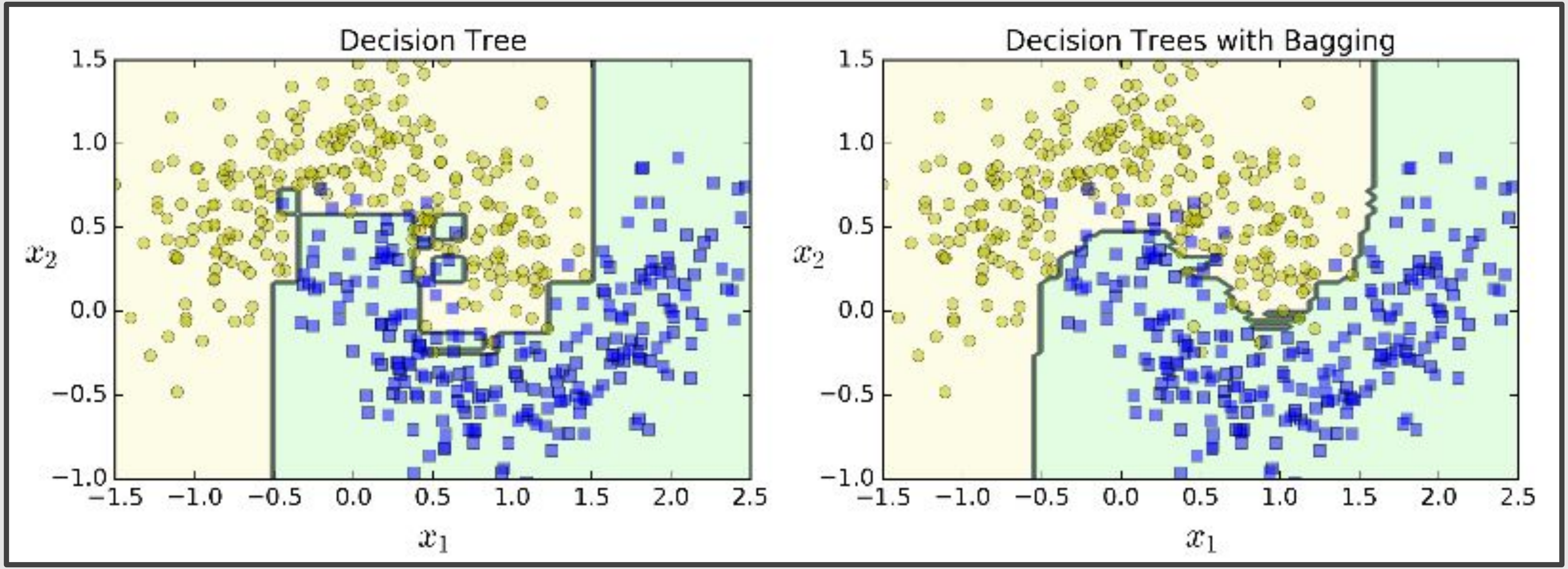
CART cost function for classification

It stops recursing once it reaches the maximum depth (hyperparameter), or if it cannot find a split that will reduce impurity.

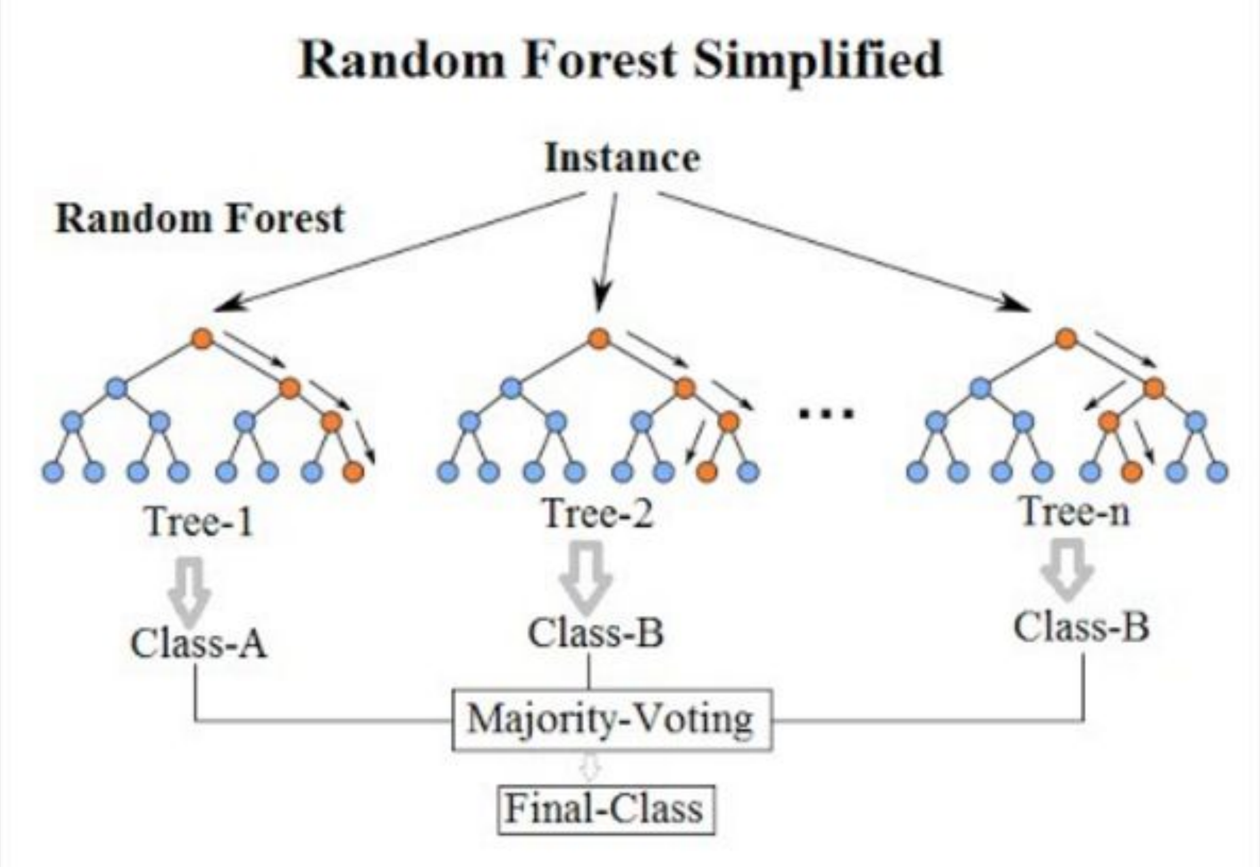
Regularization



Regularization



Random Forest



https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d

Random Forest [Ho, 1995]

- Random Forest is an ensemble of Decision Trees, generally trained using the Bagging method.

“Random Decision Forests”, Tin Kam Ho (1995): http://goo.gl/zVOGQ1

Random Forest [Ho, 1995]

- Random Forest is an ensemble of Decision Trees, generally trained using the Bagging method.

- **Extra randomness when growing trees**:
    - Instead of searching for the very best feature when splitting a node, it searches for the best feature among a **random subset of features**.

“Random Decision Forests”, Tin Kam Ho (1995): http://goo.gl/zVOGQ1

Random Forest [Ho, 1995]

1. Assume number of cases in the training set is *N*. Then, sample of these *N* cases is taken at random but with replacement.

“Random Decision Forests”, Tin Kam Ho (1995): http://goo.gl/zVOGQ1

Random Forest [Ho, 1995]

2. If there are *M* input variables, a number $m<M$ is specified such that at each node, *m* variables are selected at random out of the *M*.

   The best split on these *m* is used to split the node. The value of *m* is held constant while we grow the forest.

“Random Decision Forests”, Tin Kam Ho (1995): http://goo.gl/zVOGQ1

Random Forest [Ho, 1995]

3. Each tree is grown to the largest extent possible and there is no pruning.

4. Predict new data by aggregating the predictions of the ntree trees (i.e., majority votes for classification, average for regression).

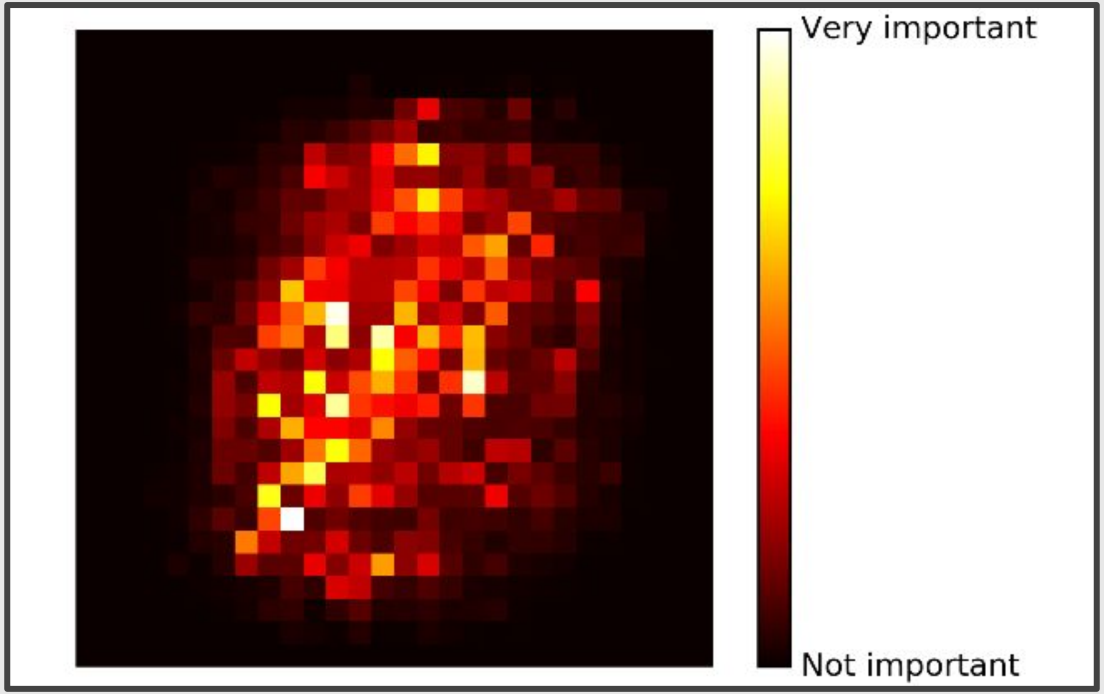
“Random Decision Forests”, Tin Kam Ho (1995): http://goo.gl/zVOGQ1

Random Forest: Feature Importance

```
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
iris = load_iris()
rnd_clf = RandomForestClassifier(n_estimators=500, n_jobs=-1)
rnd_clf.fit(iris["data"], iris["target"])
for name, score in zip(iris["feature_names"], rnd_clf.feature_importances_):
    print(name, score)

sepal length (cm) 0.112492250999
sepal width (cm) 0.0231192882825
petal length (cm) 0.441030464364
petal width (cm) 0.423357996355
```

Random Forest: Feature Importance



Forward Thinking: Building Deep Random Forests

Kevin Miller, Chris Hettinger, Jeffrey Humpherys, Tyler Jarvis, and David
Department of Mathematics
Brigham Young University
Provo, Utah 84602
millerk5@byu.edu, hettinger@math.byu.edu, jeffh@math.byu.
jarvis@math.byu.edu, david.kartchner@math.byu.edu

Abstract

Distributed Deep Forest and its Application to Automatic Detection of Cash-out Fraud

Ya-Lin Zhang†, Jun Zhou‡, Wenhao Zheng†, Ji Feng†, Longfei Li‡, Ziqi Liu‡, Ming Li†, Zhiqiang Zhang‡, Chaochao Chen‡, Xiaolong Li‡, Zhi-Hua Zhou†
†National Key Lab for Novel Software Technology, Nanjing University, China
†{zhangyl, zhengwh, fengj, lim, zhouzh}@lamda.nju.edu.cn
‡Ant Financial Services Group, China
‡{jun.zhoujun, longyao.llf, ziqiliu, lingyao.zzq, chaochao.ccc, xl.li}@antfin.com

Training Big Random Forests with Little Resources

Fabian Gieseke
Department of Computer Science
University of Copenhagen
Copenhagen, Denmark
fabian.gieseke@di.ku.dk

Christi
Department of C
University of
Copenhage
igel@di

ABSTRACT

Without access to large compute clusters, building random forests on large datasets is still a challenging problem. This is, in particular, the case if fully-grown trees are desired. We propose a simple yet effective framework that allows to efficiently construct ensembles of huge trees for hundreds of millions or even billions of training instances using a cheap desktop computer with commodity hardware. The basic idea is to consider a multi-level construction scheme, which builds top trees for small random subsets of the available data and which subsequently distributes all training instances to the top trees' leaves for further processing. While being conceptually simple, the overall efficiency crucially depends on the particular implementation of the different phases. The practical merits of our

ensembles in a parallel or distr dividual compute nodes (e.g., by node). While this can significa such frameworks naturally req ing environments. Further, the e might cause problems in case t large to fit into the main memo

In this work, we propose a scheme for building random fo scale. The main idea is to build phases: Starting with a top tree the data, one subsequently dist leaves of that tree. For each lea

Deep Forest: Towards an Alternative to Deep Neural Networks*

Zhi-Hua Zhou and **Ji Feng**
National Key Lab for Novel Software Technology, Nanjing University, Nanjing 210023, China
{zhouzh, fengj}@lamda.nju.edu.cn

Abstract

In this paper, we propose gcForest, a decision tree

ample, even when several authors all use convolutional neural networks [LeCun *et al.*, 1998; Krizhenvsky *et al.*, 2012; Simonyan and Zisserman, 2014], they are actually using dif-

References

Machine Learning Books

- Hands-On Machine Learning with Scikit-Learn and TensorFlow, Chap. 6 & 7
- Pattern Recognition and Machine Learning, Chap. 14
- Pattern Classification, Chap 8 & 9 (Sec. 9.5)

- https://towardsdatascience.com/random-forest-in-python-24d0893d51c0