Support Vector Machine (SVM)

Machine Learning and Pattern Recognition

Prof. Sandra Avila

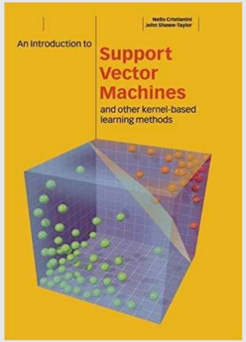
Institute of Computing (IC/Unicamp)

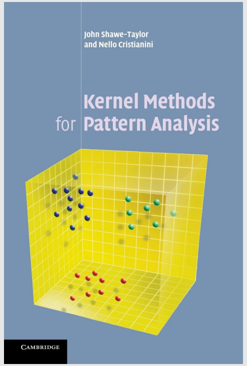
MC886/MO444, November 13, 2018

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

Andrew Ng

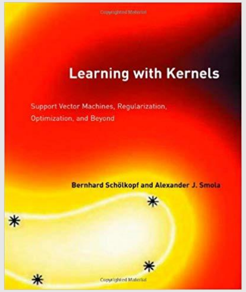
"**An Introduction to Support Vector Machines**: And Other Kernel-based Learning Methods", Cristianini & Shawe-Taylor, 2000.



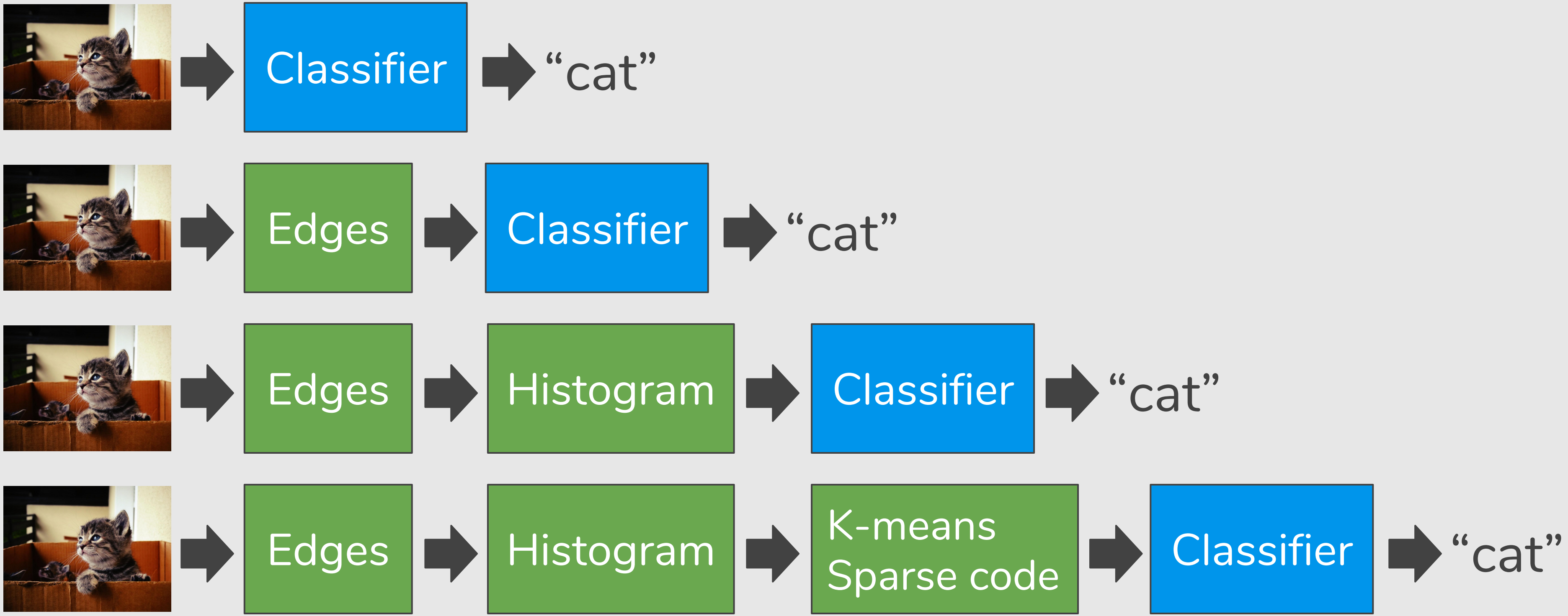


"**Kernel Methods for Pattern Analysis**", Shawe-Taylor & Cristianini, 2004.

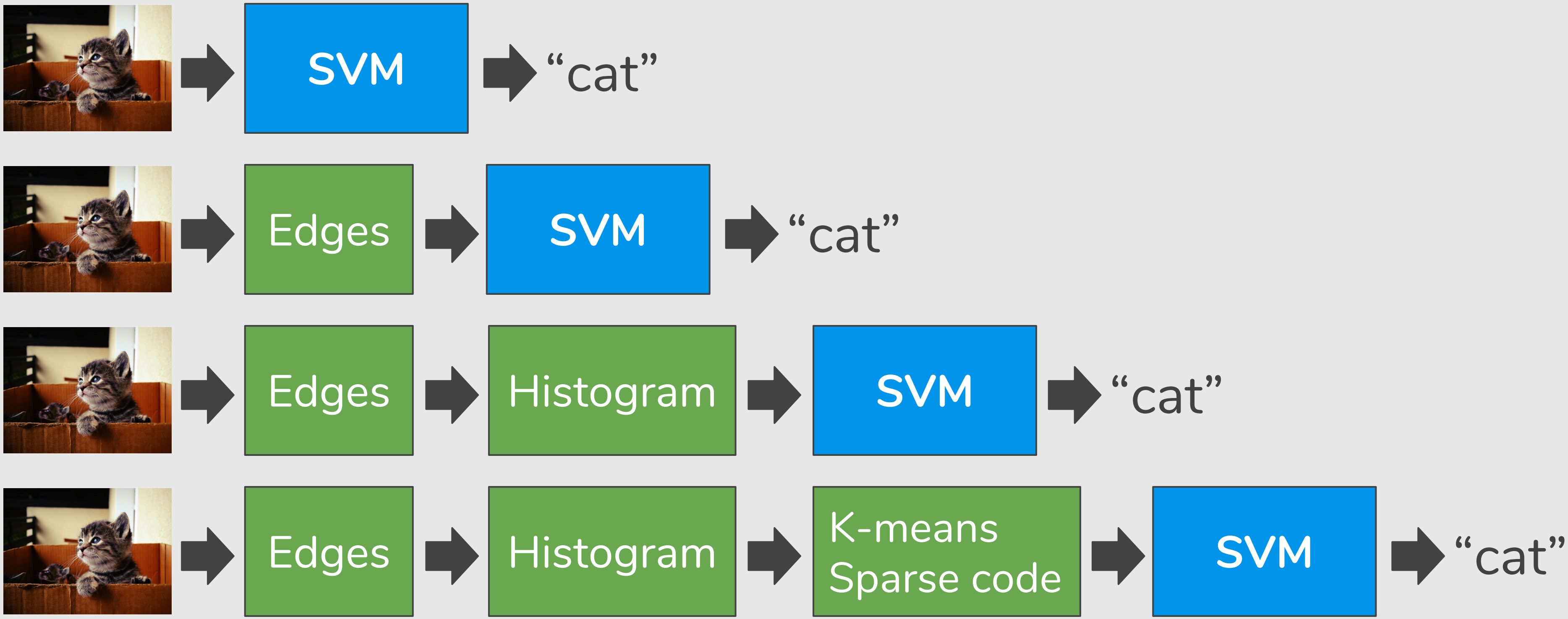
"**Learning with Kernels**: Support Vector Machines, Regularization, Optimization, and Beyond", Scholkopf & Smola, 2001.



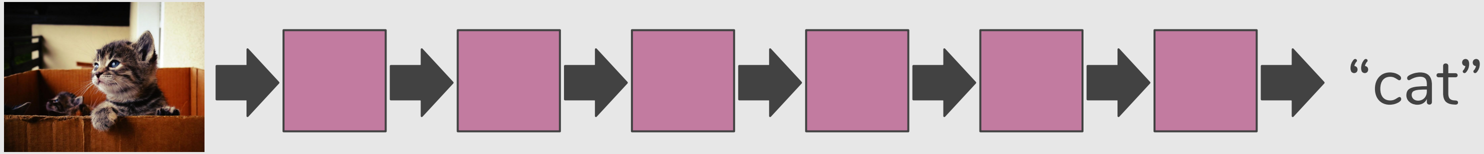
Traditional Recognition



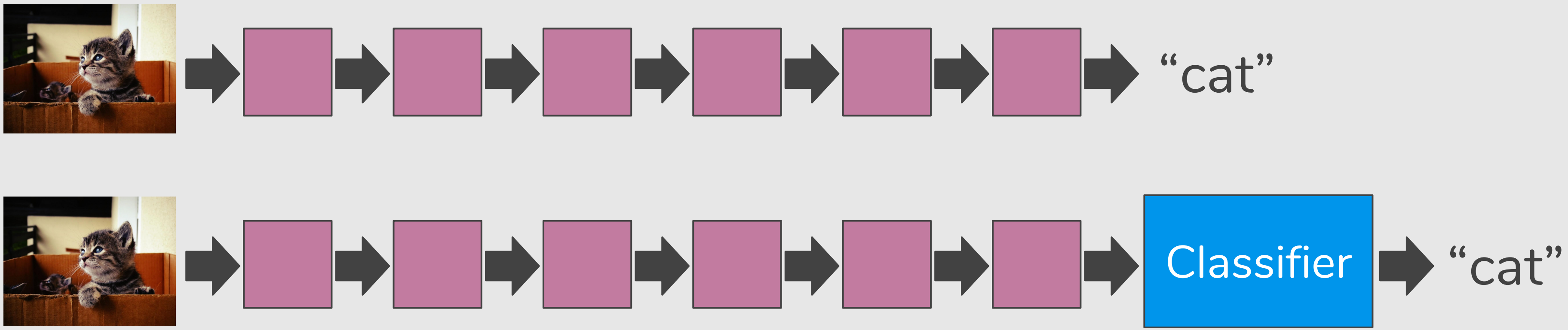
Traditional Recognition



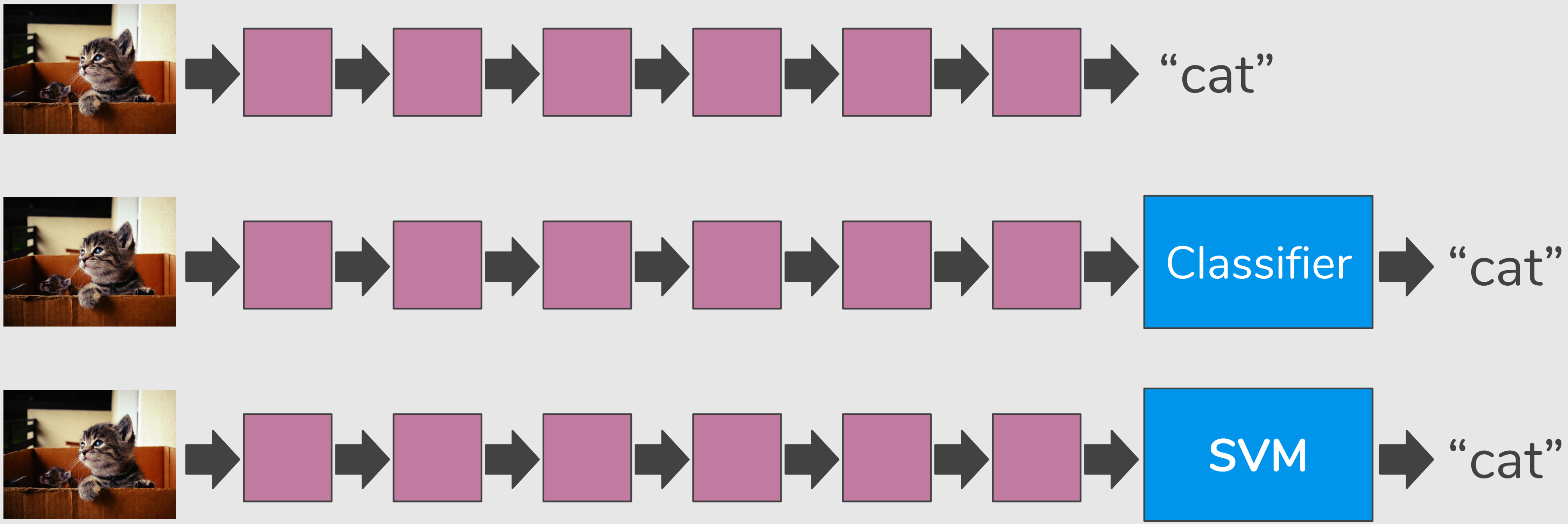
Deep Learning



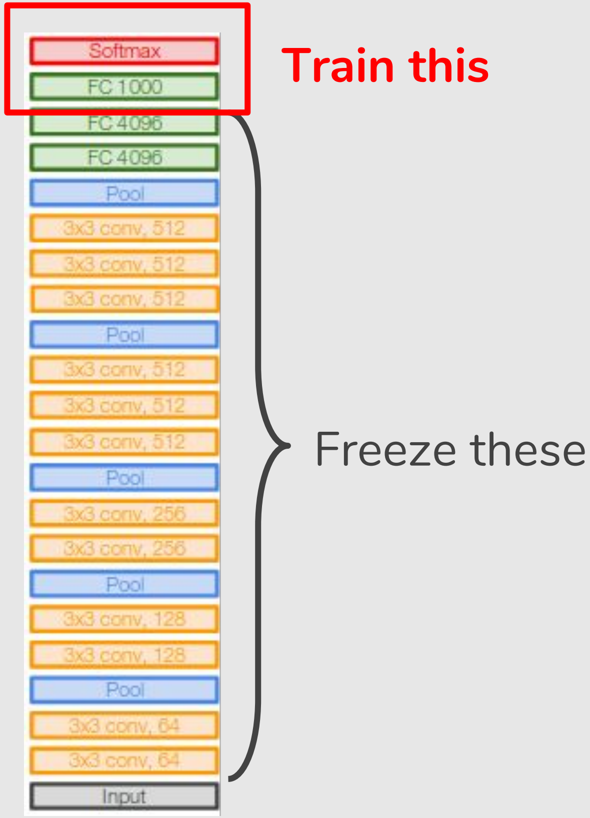
Deep Learning



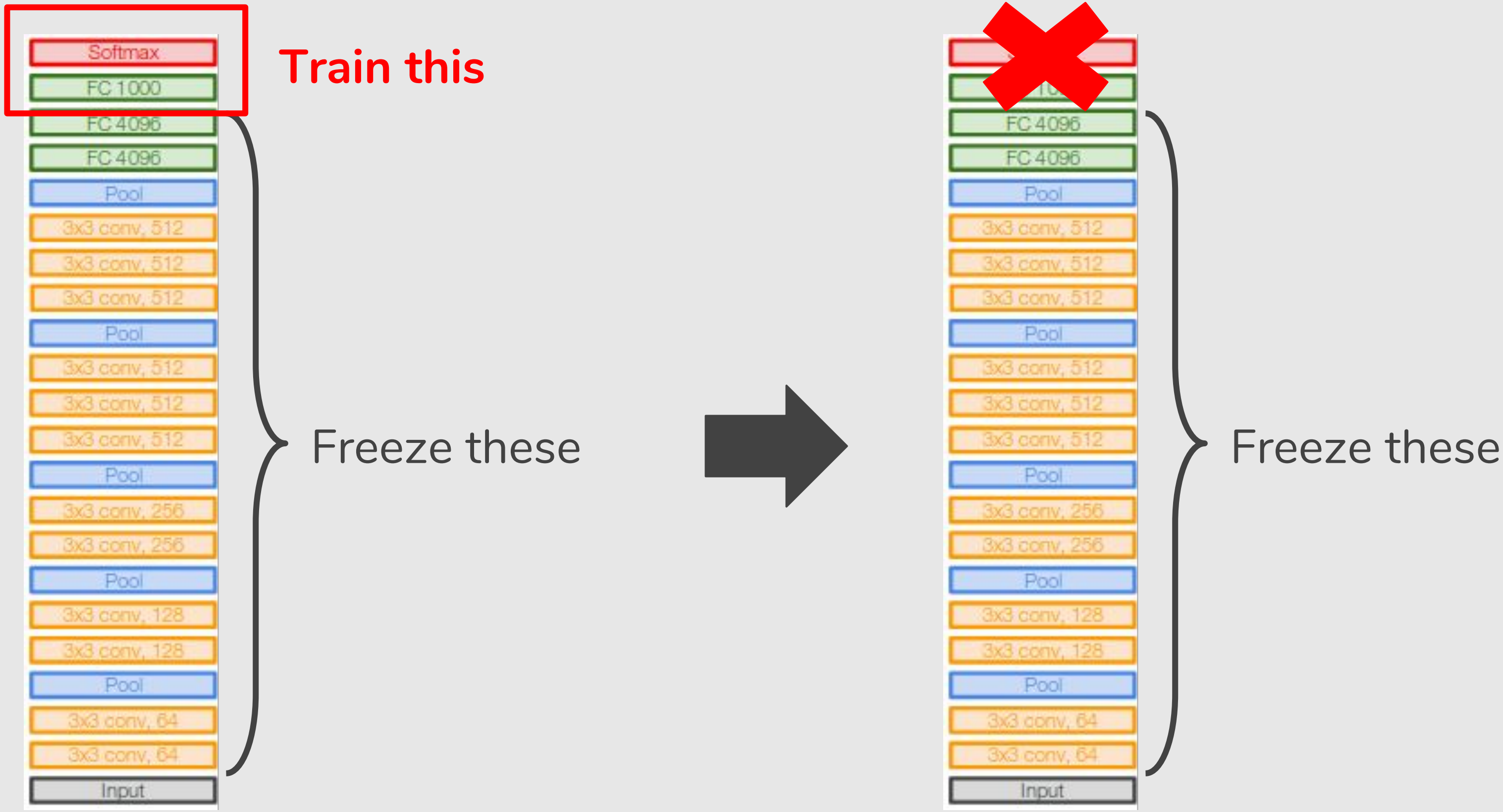
Deep Learning



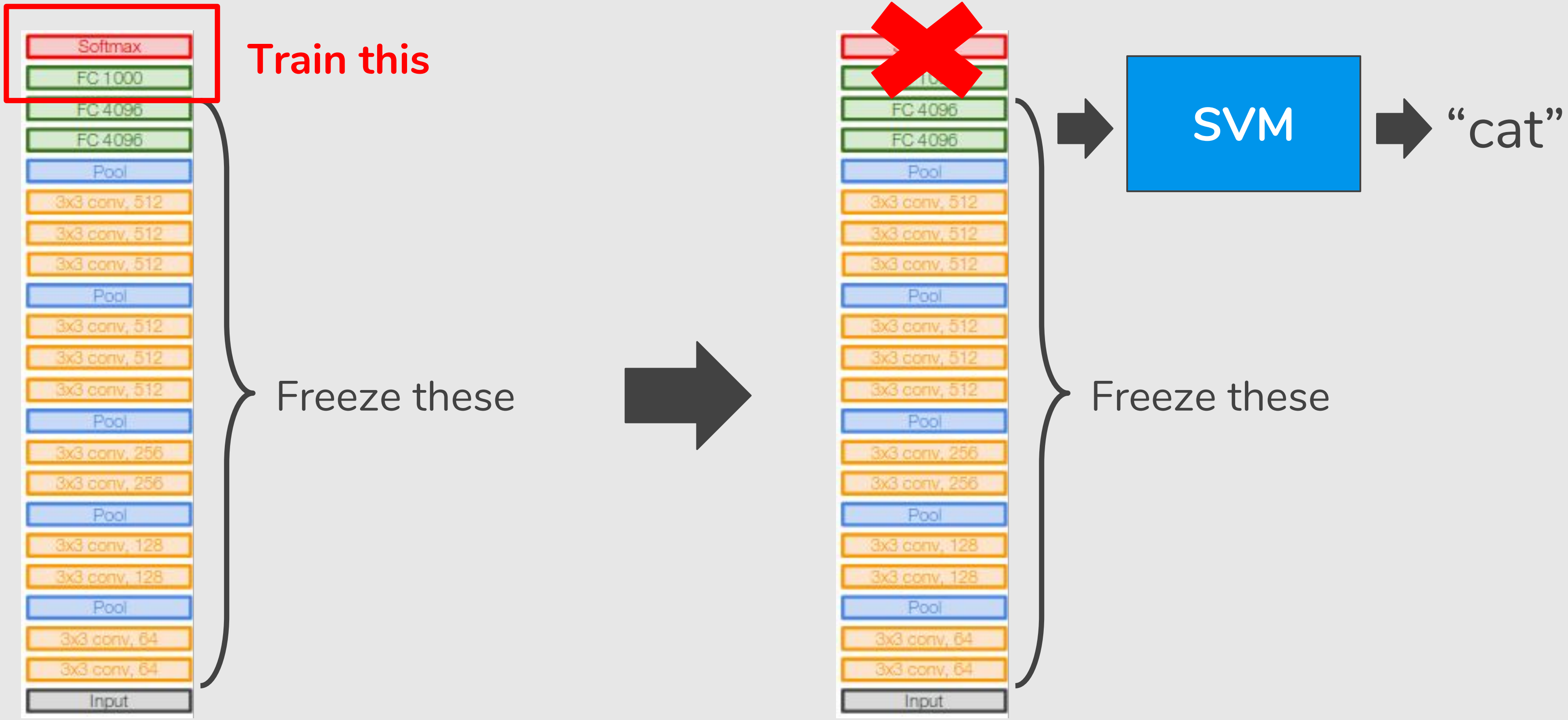
Transfer Learning



Transfer Learning



Transfer Learning

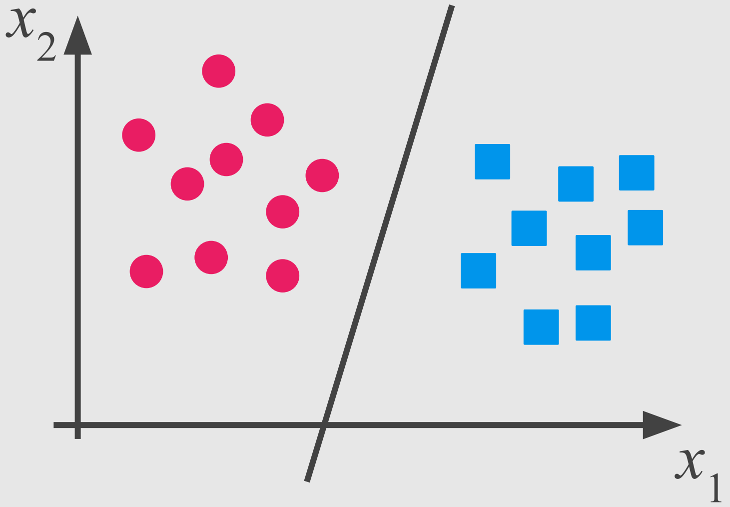


What is Support Vector Machine?

Support Vector Machine (SVM)

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

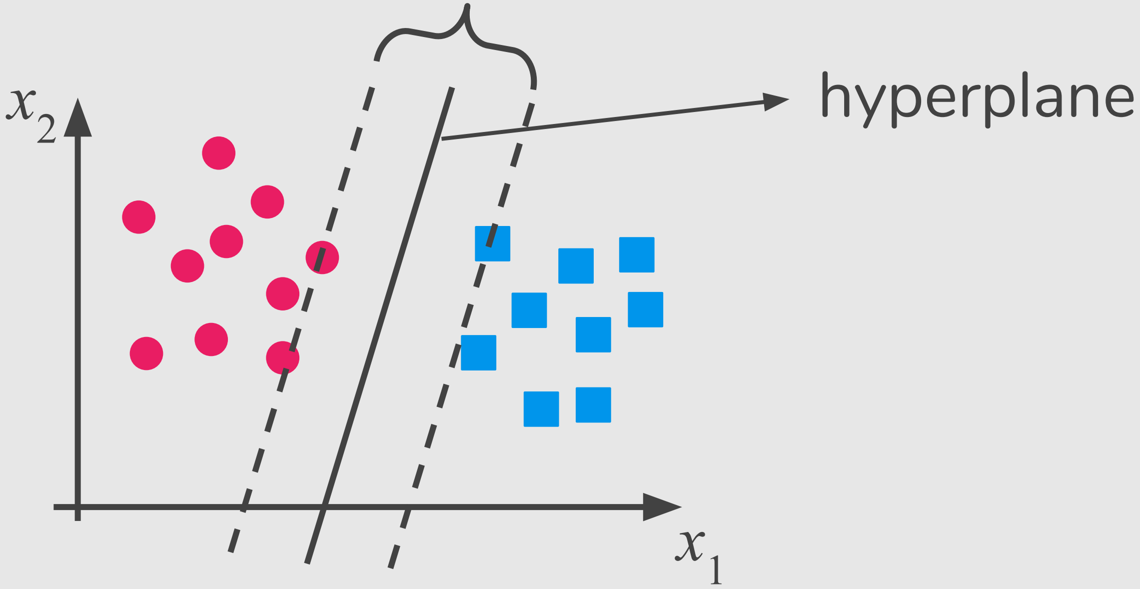
Idea of separating data with **a large "gap"**.



Support Vector Machine (SVM)

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

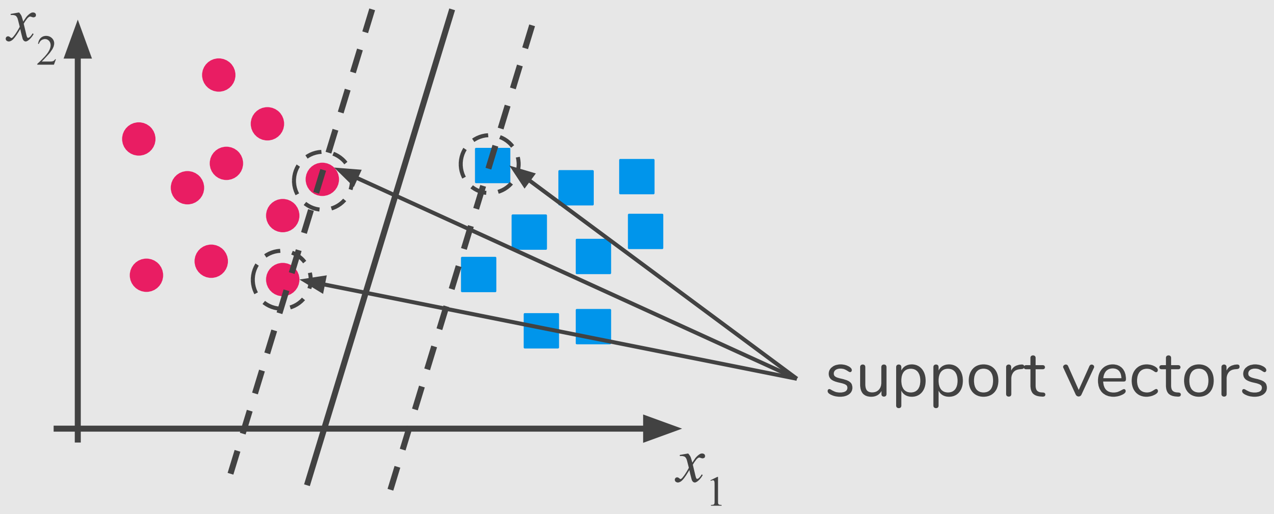
Idea of separating data with **a large "gap"**.



Support Vector Machine (SVM)

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

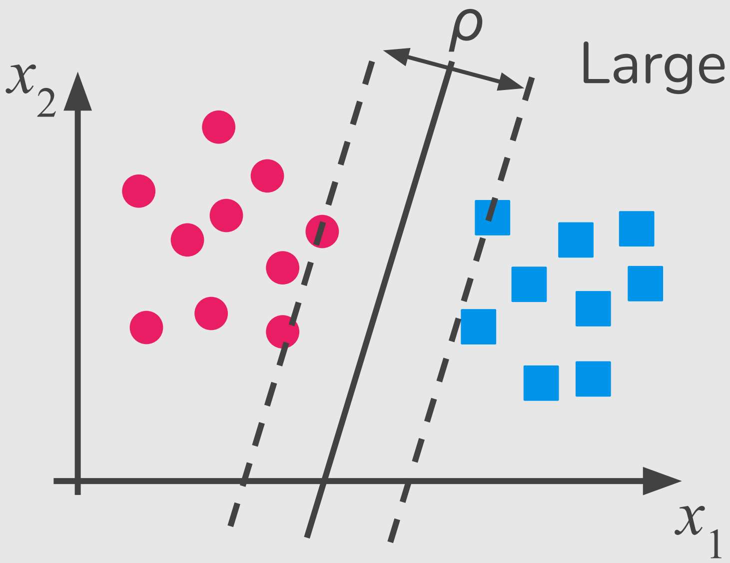
Examples closest to the hyperplane are support vectors.



Support Vector Machine (SVM)

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

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

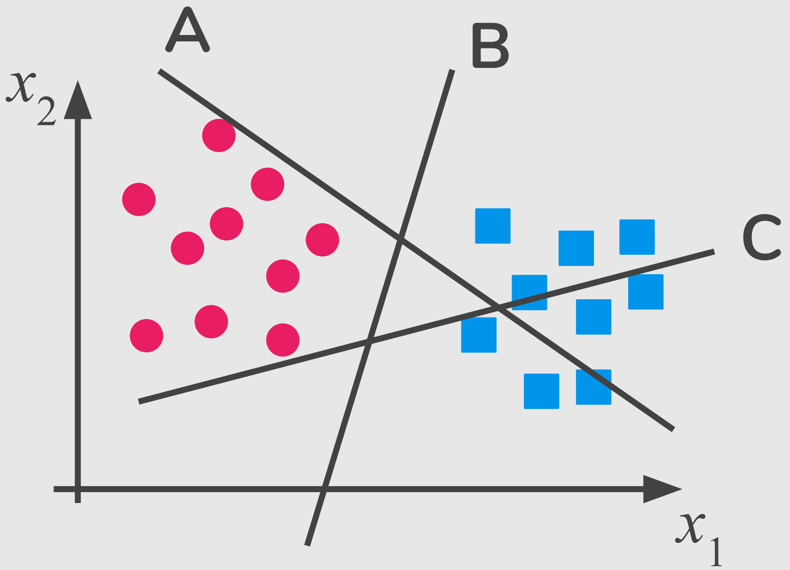


Large margin classifier

How does SVM work?

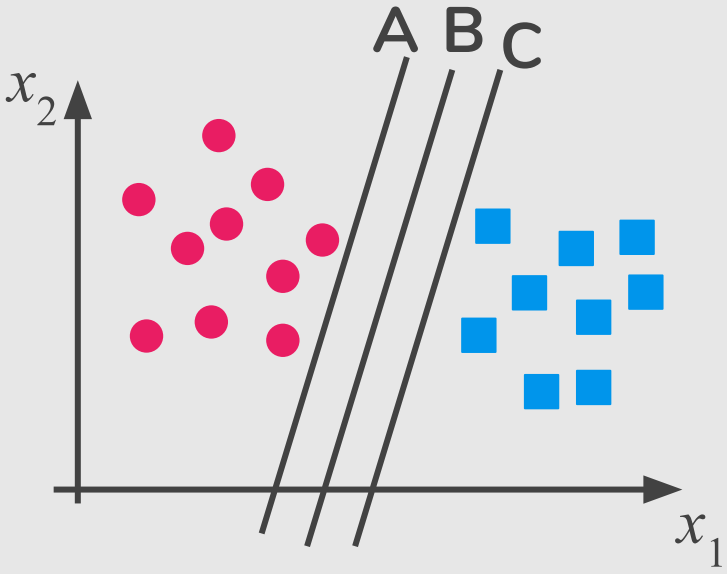
How can we identify the right hyperplane?

Scenario 1



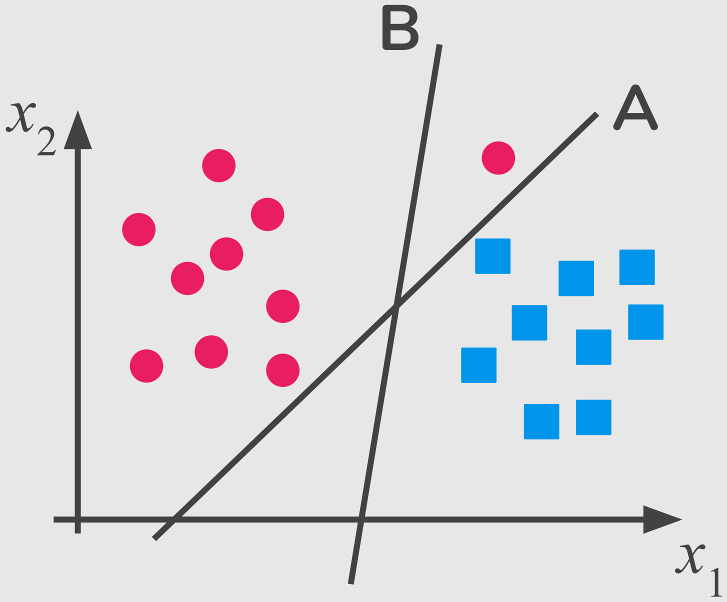
How can we identify the right hyperplane?

Scenario 2



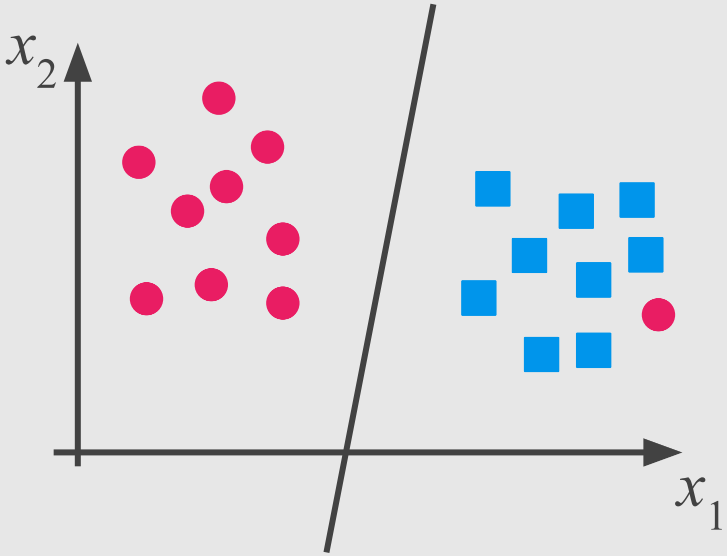
How can we identify the right hyperplane?

Scenario 3



How can we identify the right hyperplane?

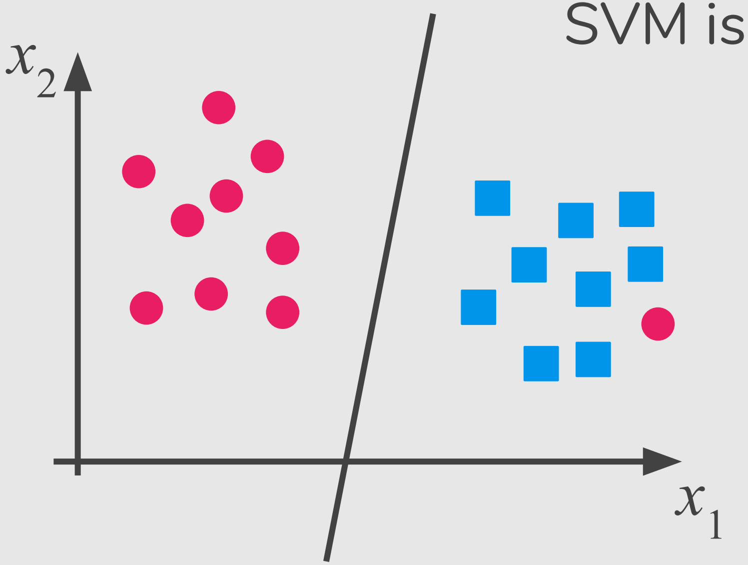
Scenario 4



How can we identify the right hyperplane?

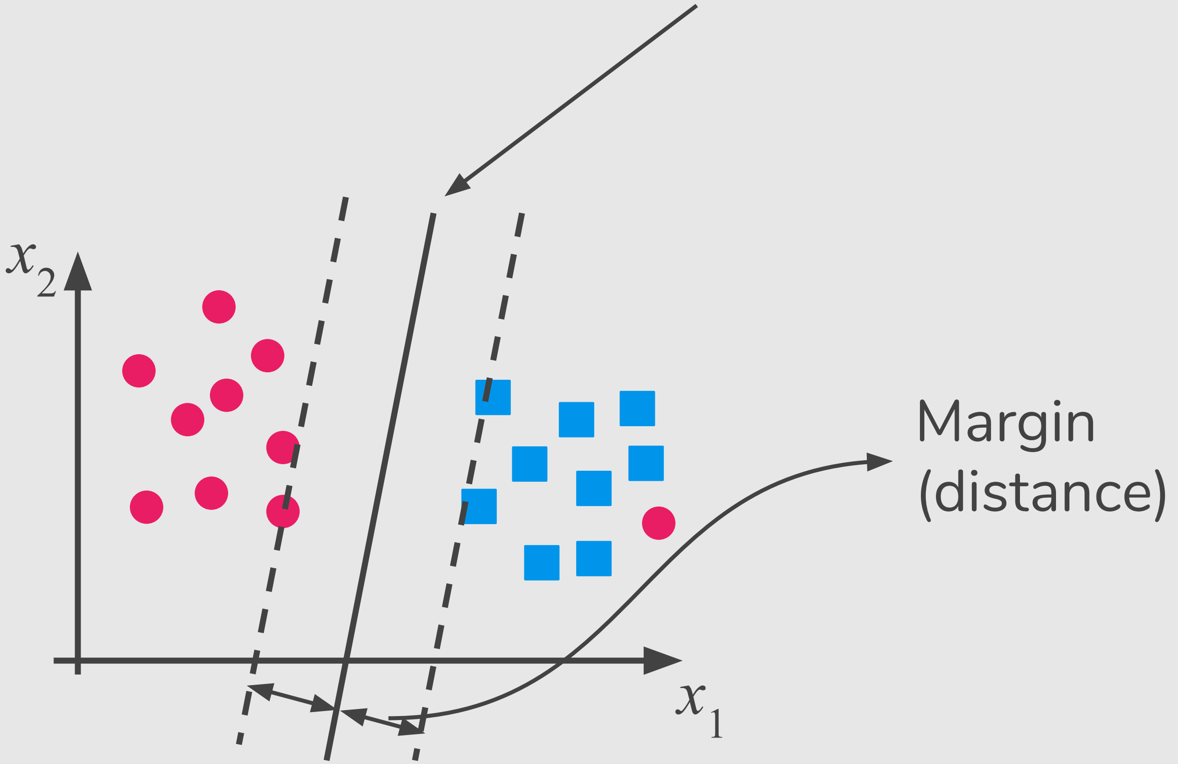
Scenario 4

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:

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

$$\Downarrow$$

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

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

$$\min_{w,b} \ \tfrac{1}{2}||w||^2$$
$$\text{s.t. } y^{(i)}(w^T x + b) \geq 1,\ i = \{1, \ldots, m\}$$

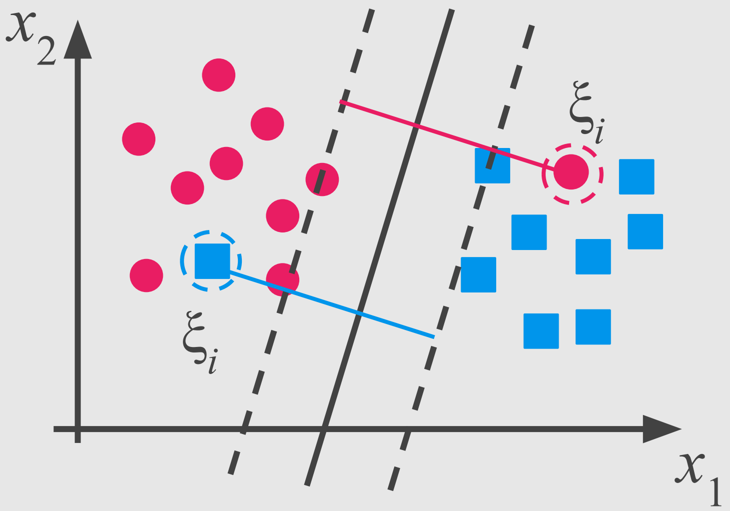
Need to optimize a quadratic function subject to linear constraints.

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

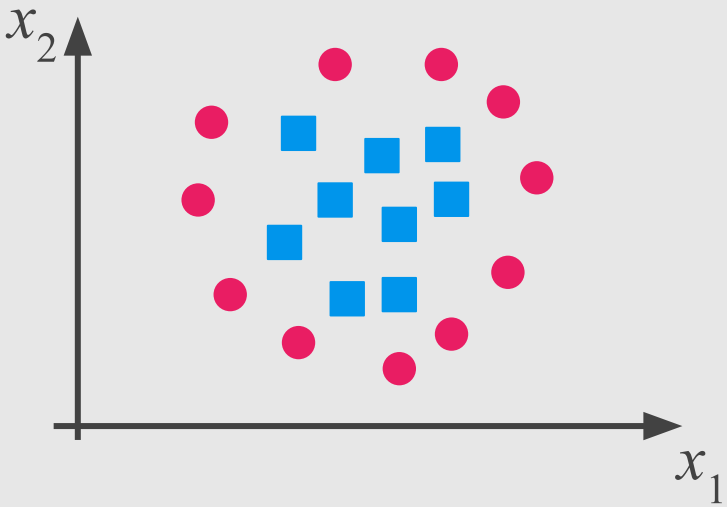
Modified formulation incorporates slack variables:

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

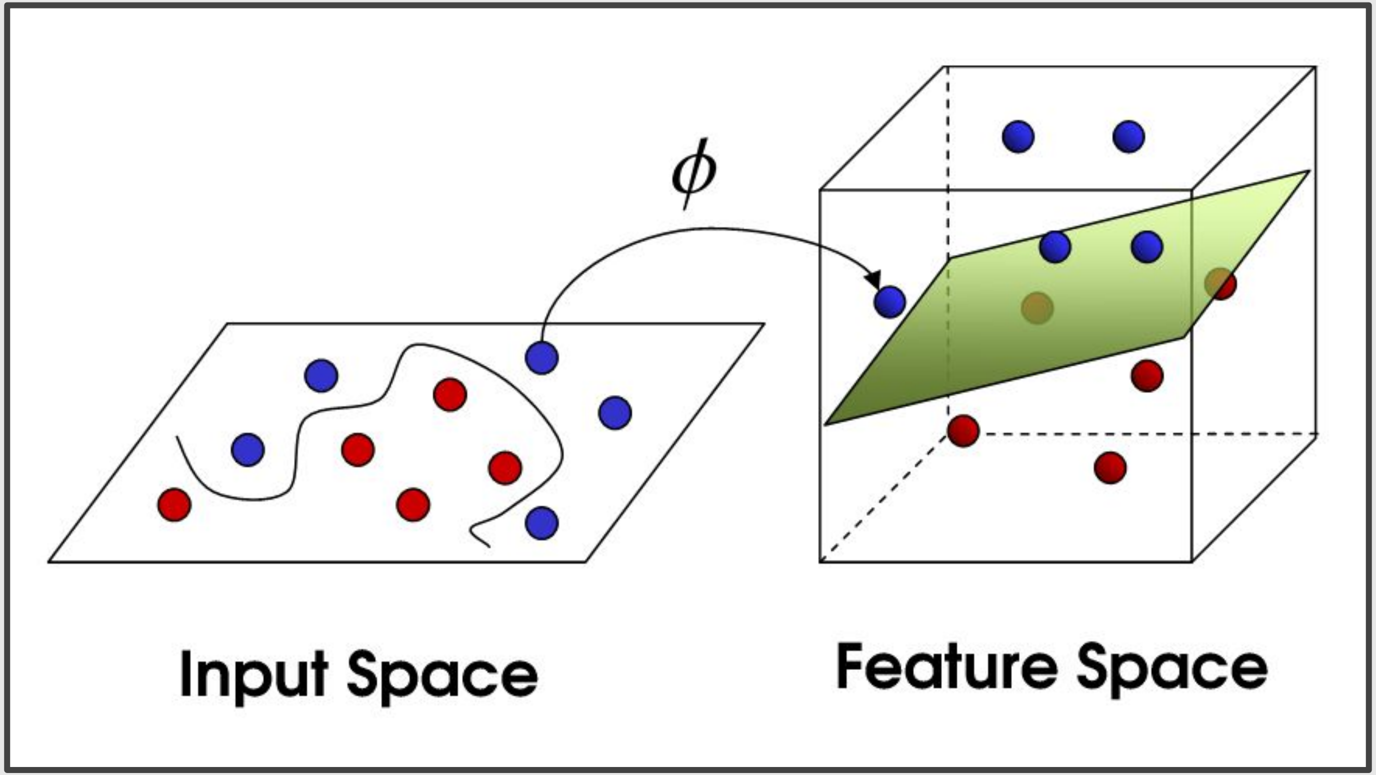
Parameter C can be viewed as a way to control overfitting:
it "trades off" the relative importance of maximizing the margin and fitting the training data.

How can we identify the right hyperplane?

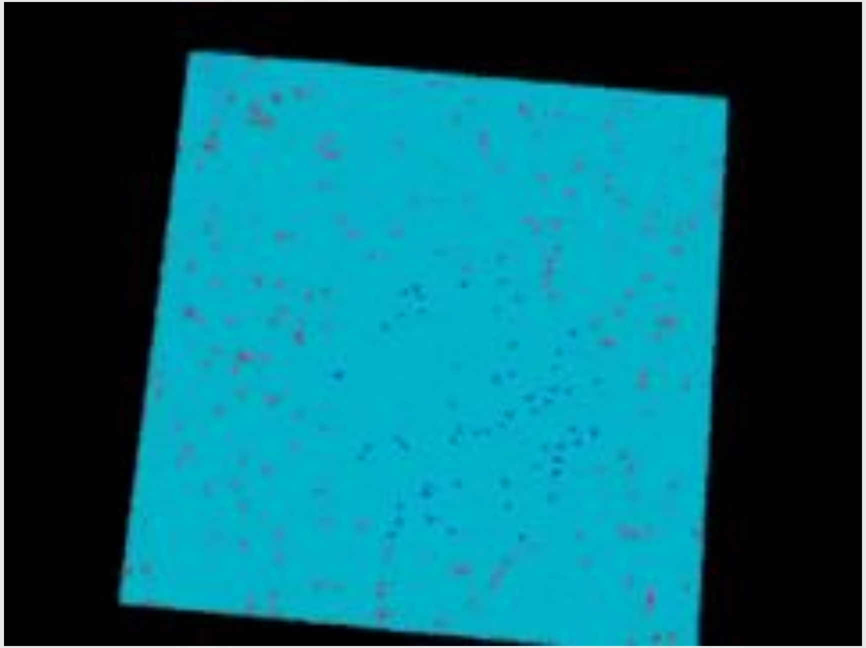
Scenario 5



Kernel Trick



Kernel Trick



Kernelized SVM

Suppose you want to apply a 2nd degree polynomial transformation to a 2-dimensional training set, then train a linear SVM classifier on the transformed training set.

$$\varphi(x) = \varphi((x_1\ x_2)) = (x_1{}^2\ \ \sqrt{2x_1x_2}\ \ x_2{}^2)$$

Kernelized SVM

Suppose you want to apply a 2nd degree polynomial transformation to a 2-dimensional training set, then train a linear SVM classifier on the transformed training set.

$$\varphi(x) = \varphi((x_1\ x_2)) = (x_1{}^2\ \ \sqrt{2x_1x_2}\ \ x_2{}^2)$$

The transformed vector is 3-dimensional instead of 2-dimensional.

Kernelized SVM

Let's look at what happens to a couple 2-dimensional vectors **a** and **b** if we apply this 2nd degree polynomial mapping then compute the dot product of the transformed vectors.

$$\varphi(\mathbf{a})^T \varphi(\mathbf{b}) =$$

Kernelized SVM

Let's look at what happens to a couple 2-dimensional vectors **a** and **b** if we apply this 2nd degree polynomial mapping then compute the dot product of the transformed vectors.

$$\varphi(\mathbf{a})^T\varphi(\mathbf{b}) = (a_1{}^2 \;\; \sqrt{2a_1a_2} \;\; a_2{}^2)^T(b_1{}^2 \;\; \sqrt{2b_1b_2} \;\; b_2{}^2) =$$

Kernelized SVM

Let’s look at what happens to a couple 2-dimensional vectors **a** and **b** if we apply this 2nd degree polynomial mapping then compute the dot product of the transformed vectors.

$$
\begin{aligned}
\varphi(\mathbf{a})^{\mathrm{T}}\varphi(\mathbf{b}) &= (a_1{}^2 \;\; \sqrt{2}a_1a_2 \;\; a_2{}^2)^{\mathrm{T}}(b_1{}^2 \;\; \sqrt{2}b_1b_2 \;\; b_2{}^2) = \\
&= a_1{}^2b_1{}^2 + 2a_1a_2b_1b_2 + a_2{}^2b_2{}^2 = \\
&= (a_1b_1 + a_2b_2)^2 = \\
&= ((a_1 \; a_2)^{\mathrm{T}}(b_1 \; b_2))^2 = \\
&= (\mathbf{a}^{\mathrm{T}} \cdot \mathbf{b})^2
\end{aligned}
$$

Kernelized SVM

The dot product of the transformed vectors is equal to the square of the dot product of the original vectors: $\varphi(\mathbf{a})^{T}\varphi(\mathbf{b}) = (\mathbf{a}^{T} \cdot \mathbf{b})^{2}$

Kernelized SVM

The dot product of the transformed vectors is equal to the square of the dot product of the original vectors: $\varphi(\mathbf{a})^{\mathrm{T}}\varphi(\mathbf{b}) = (\mathbf{a}^{\mathrm{T}} \cdot \mathbf{b})^2$

So **you don't actually need to transform the training instances** at all: just replace the dot product by its square.

This is the essence of the kernel trick.

Kernelized SVM

In Machine Learning, a ***kernel*** is a function capable of computing the dot product $\varphi(\mathbf{a})^T\varphi(\mathbf{b})$ based only on the original vectors **a** and **b**, without having to compute (or even to know about) the transformation φ.

Kernel Trick

- Linear SVM: $x_i \cdot x_j$

Kernel Trick

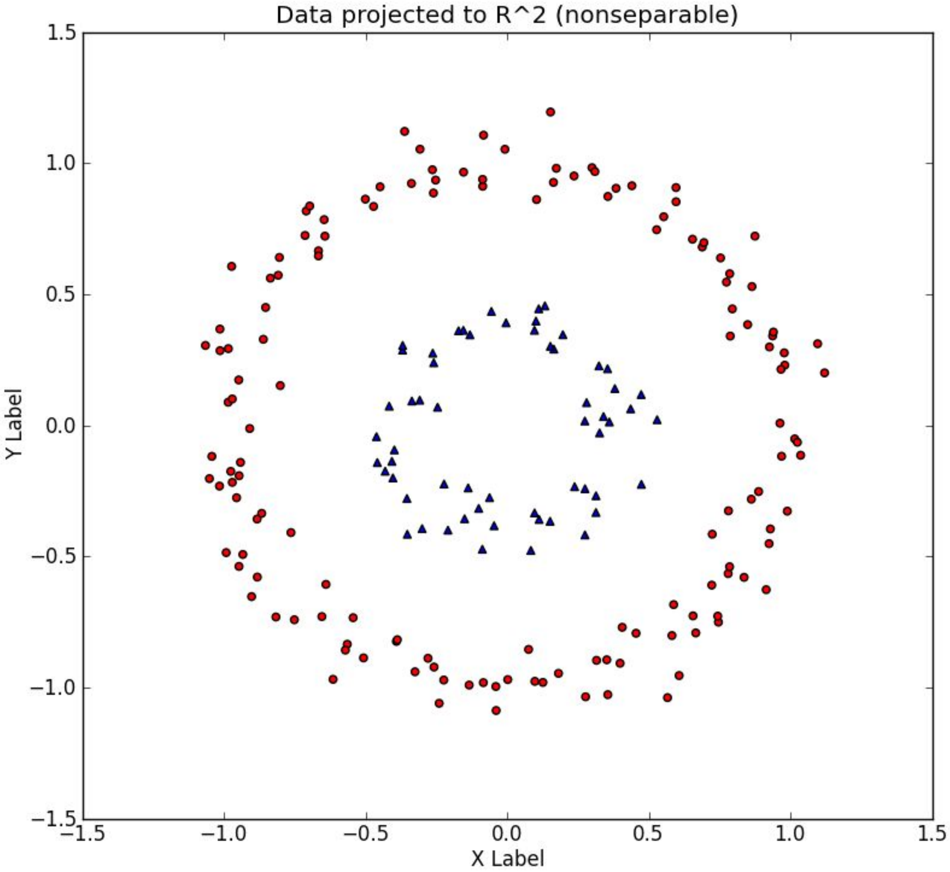
- Linear SVM: $x_i \cdot x_j$
- Nonlinear SVM: $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$, feature mapping ϕ

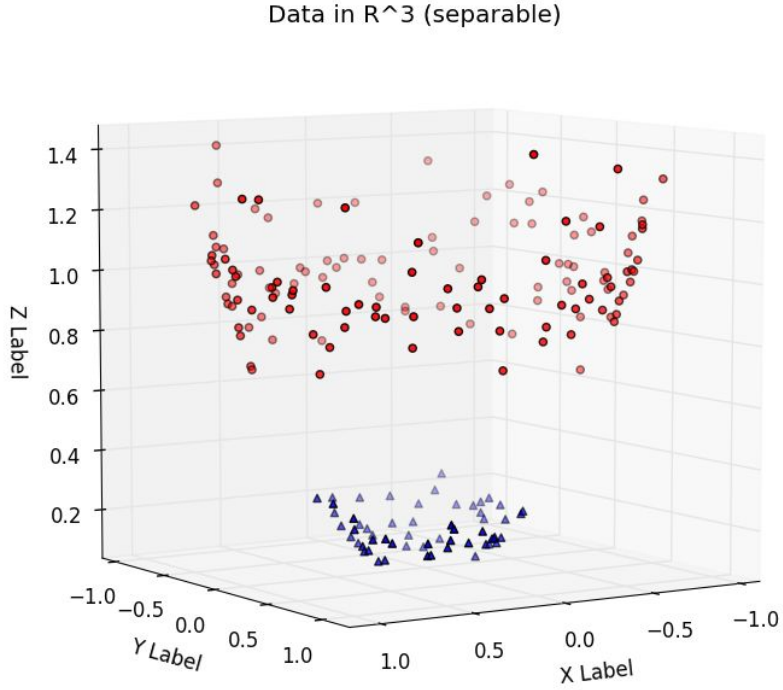
Kernel Trick

- 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)
X Label
Y Label


Data in R^3 (separable)
X Label
Y Label
Z Label

SVM & scikit-learn

SVM is also available in scikit-learn library and follow the same structure: import library, object creation, fitting model and prediction.

SVM & scikit-learn

```
#Import Library
from sklearn import svm
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor)
of test_dataset
# Create SVM classification object
model = svm.svc(kernel='linear', c=1, gamma=1)
# there is various option associated with it, like changing kernel, gamma and C value. Will
discuss more # about it in next section.Train the model using the training sets and check s
core
model.fit(X, y)
model.score(X, y)
#Predict Output
predicted= model.predict(x_test)
```

SVM & scikit-learn

```
#Import Library
from sklearn import svm
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor)
of test_dataset
# Create SVM classification object
model = svm.svc(kernel='linear', c=1, gamma=1)
# there is various option associated with it, like changing kernel, gamma and C value. Will
discuss more # about it in next section.Train the model using the training sets and check s
core
model.fit(X, y)
model.score(X, y)
#Predict Output
predicted= model.predict(x_test)
```

object creation

SVM & scikit-learn

```
#Import Library
from sklearn import svm
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor)
of test_dataset
# Create SVM classification object
model = svm.svc(kernel='linear', c=1, gamma=1)
# there is various option associated with it, like changing kernel, gamma and C value. Will
discuss more # about it in next section.Train the model using the training sets and check s
core
model.fit(X, y)
model.score(X, y)
#Predict Output
predicted= model.predict(x_test)
```

fitting model

SVM & scikit-learn

```
#Import Library
from sklearn import svm
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor)
of test_dataset
# Create SVM classification object
model = svm.svc(kernel='linear', c=1, gamma=1)
# there is various option associated with it, like changing kernel, gamma and C value. Will
discuss more # about it in next section.Train the model using the training sets and check s
core
model.fit(X, y)
model.score(X, y)
#Predict Output
predicted= model.predict(x_test)
```

prediction

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.

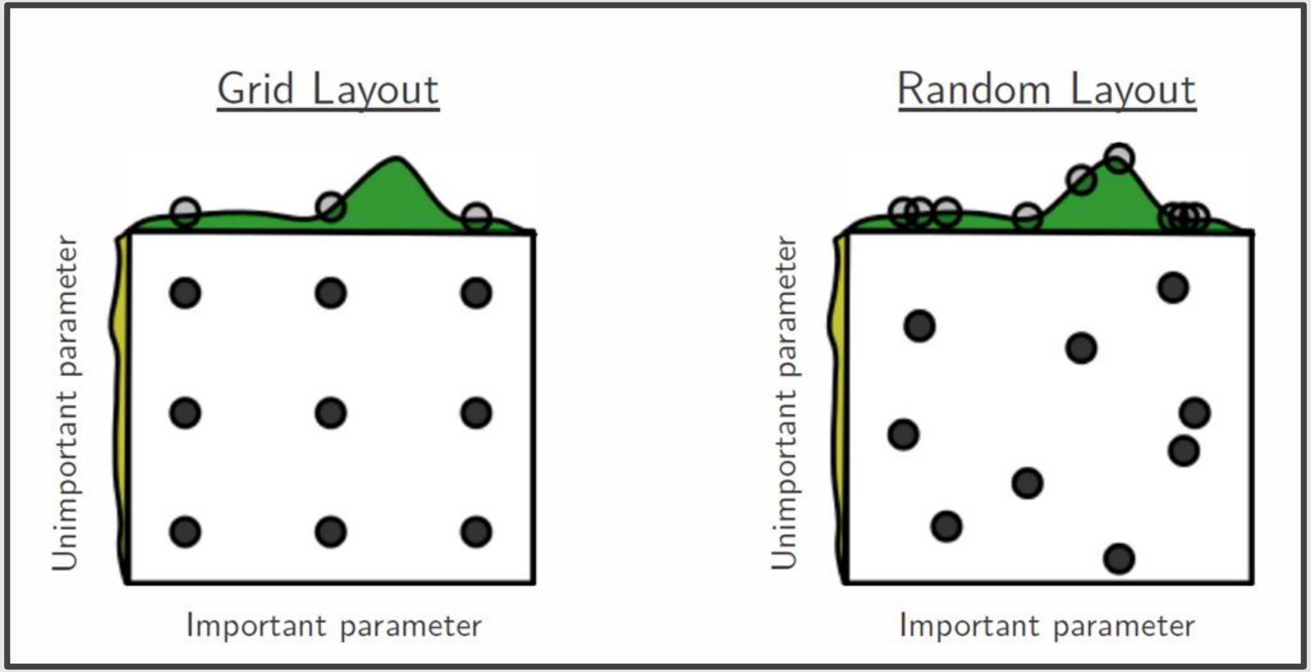
Important Parameters

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