Advanced Machine Learning for KCS
Lecture 3.1: Kernel Methods + Sparse Kernel Machines

11.09.2023

Stella

Department of Computer Science

IT UNIVERSITY OF COPENHAGEN

Join at menti.com use code 7807 1887

Mentimeter

I read the material for today...

no

yes

I read the material for today

I understood the material

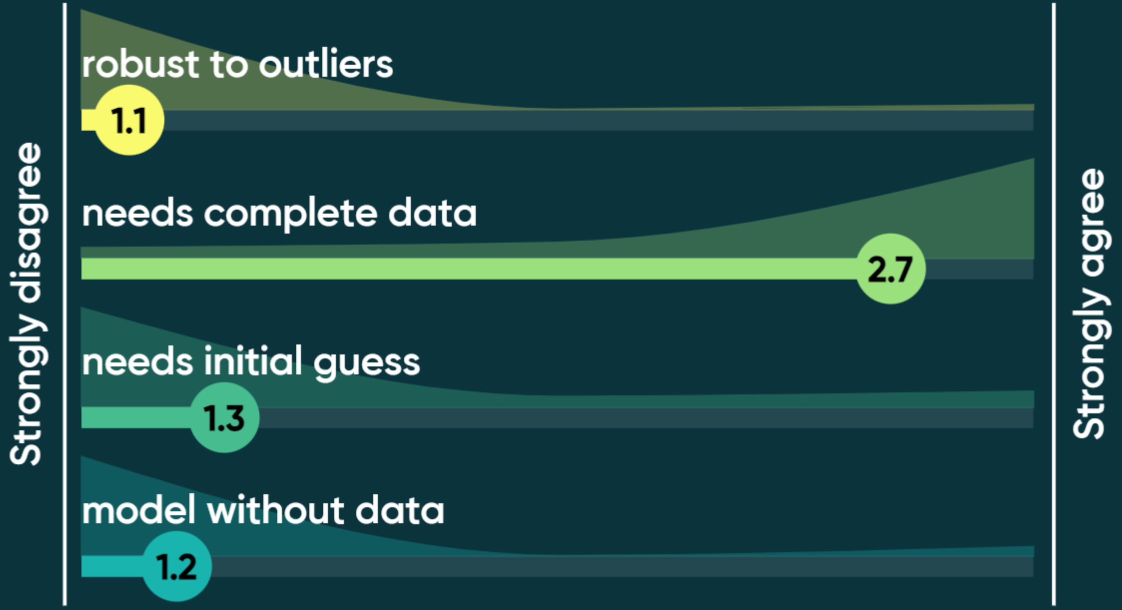
Stella uses too many slides

-

Join at menti.com use code 6869 0571

Mentimeter

Direct Least Squares Estimate properties

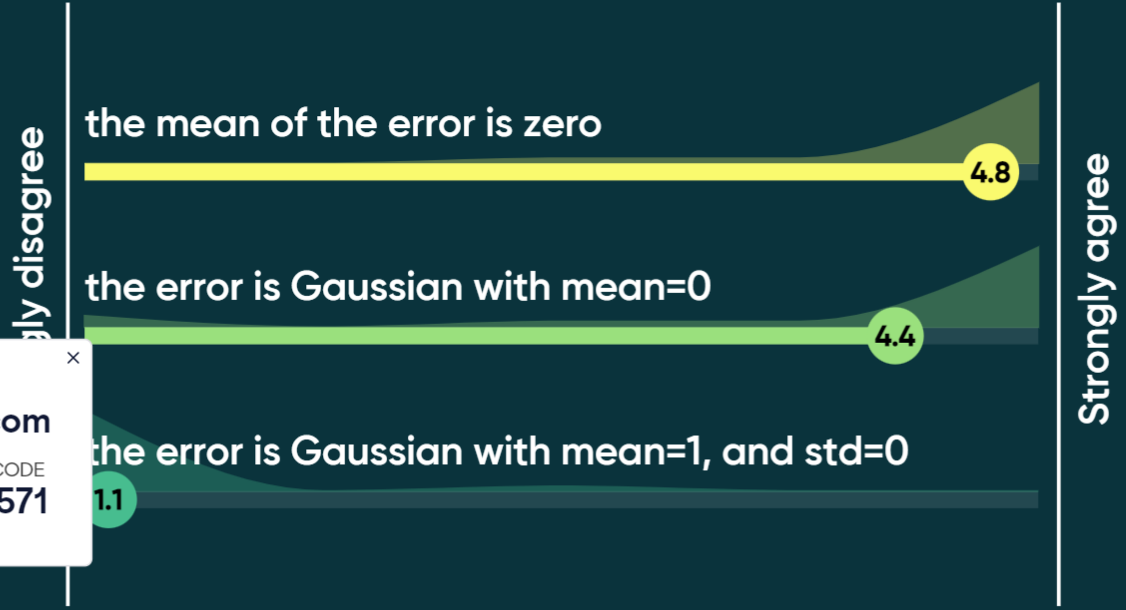


24

Join at menti.com use code 6869 0571

Mentimeter

In the standard linear regression model we assume



GO TO
menti.com
ENTER THE CODE
6869 0571

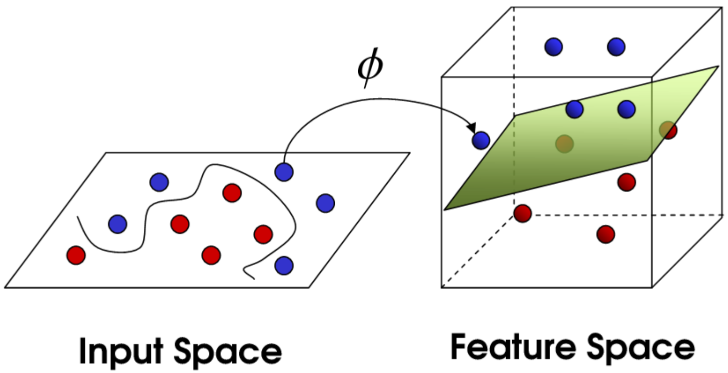
18

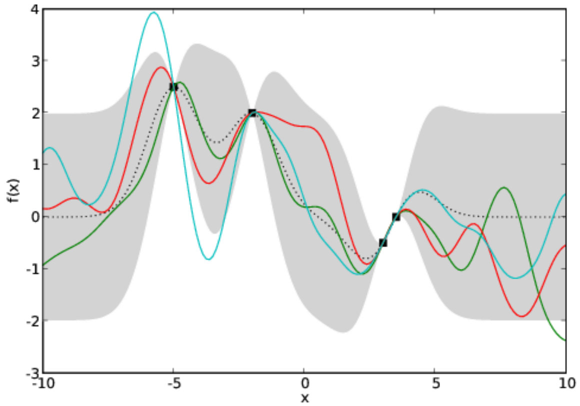
ILOs

- Define what kernel methods are
- Apply kernel methods to new problems
- Define what a Gaussian Process is
- Define what a SVM is

Outline

- Kernel Methods
  - Kernel functions
  - dual form
  - Gaussian Processes for Regression
- Sparse Kernel Machines
  - Support Vector Machine (SVM)





Linear Models

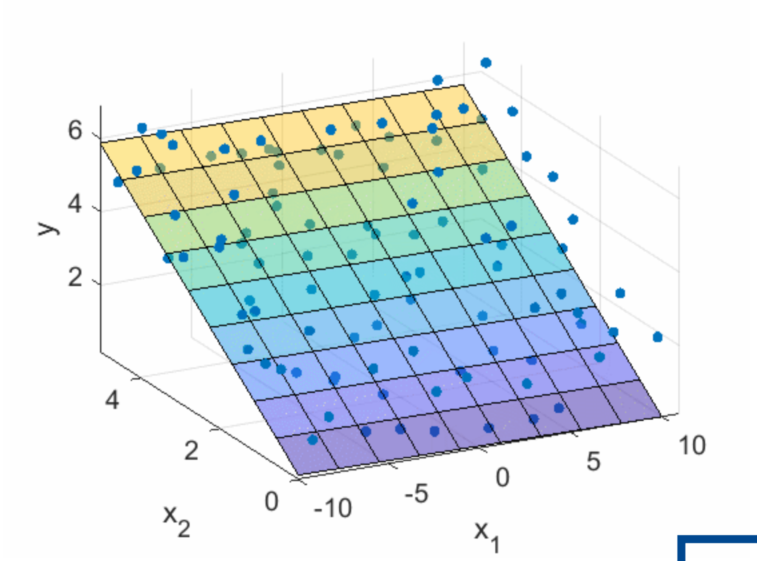
What is a „linear“ model?

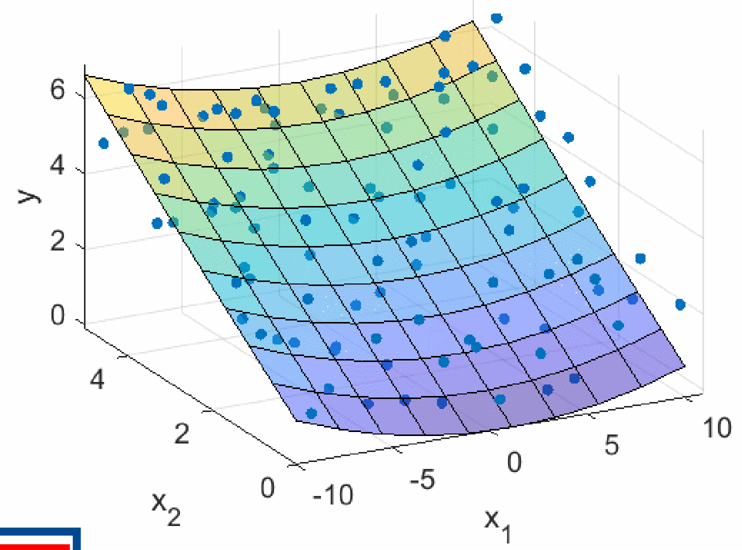
$$(1) \quad y_n = w_0 + w_1 x_n + \epsilon_n$$
$$(2) \quad y_n = w_0 + w_1 x_{1n} + w_2 x_{2n} + \epsilon_n$$
$$(3) \quad y_n = w_0 + w_1 x_n + w_2 x_n^2 + \epsilon_n$$
$$(4) \quad y_n = w_0 + w_1 x_n + w_2^2 x_n + \epsilon_n$$
$$(5) \quad y_n = w_0 + w_1 \log(x_n) + w_2 x_n + \epsilon_n$$
$$(6) \quad y_n = w_0 + w_1 \log(x_n) + w_2^3 x_n + \epsilon_n$$

$$\leadsto \boldsymbol{y} = \boldsymbol{Z}\boldsymbol{w} + \boldsymbol{\epsilon}$$

1, 2, 3, 5 are linear with respect to the parameters $\boldsymbol{w}$

Linear Models vs. Nonlinear decision boundaries



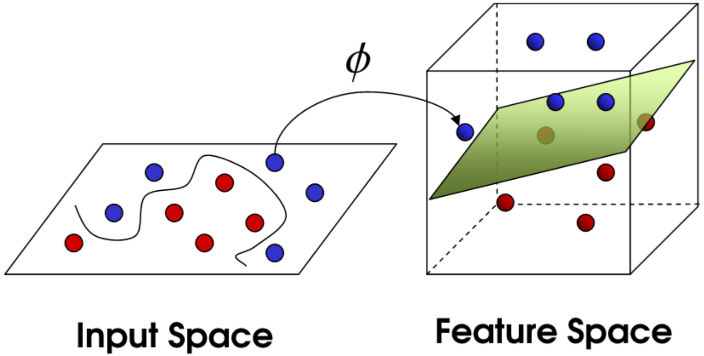


$$\widehat{y}_n = \widehat{w}_0 + \sum_{d=1}^{2} \widehat{w}_d \boxed{x_{nd}}$$

$$\boxed{\begin{aligned} z_{n1} &= x_{n1} \\ z_{n2} &= x_{n2} \\ z_{n3} &= x_{n1}^2 \\ z_{n4} &= x_{n2}^2 \end{aligned}}$$

$$\widehat{y}_n = \widehat{w}_0 + \sum_{m=1}^{4} \widehat{w}_m \boxed{z_{nm}}$$

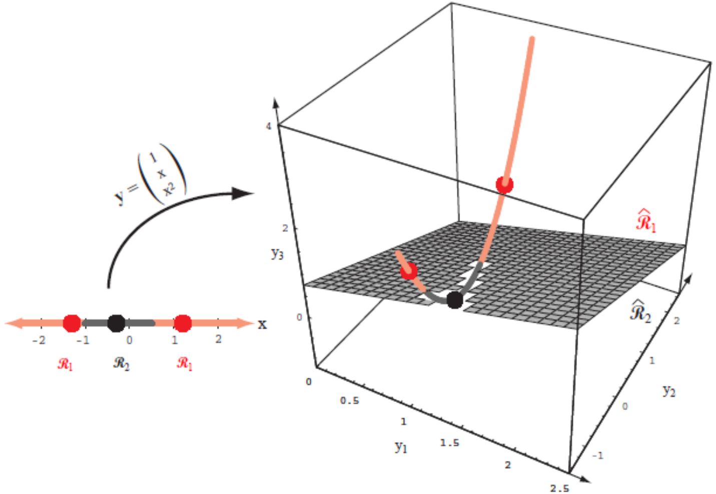
Non-linear transformation

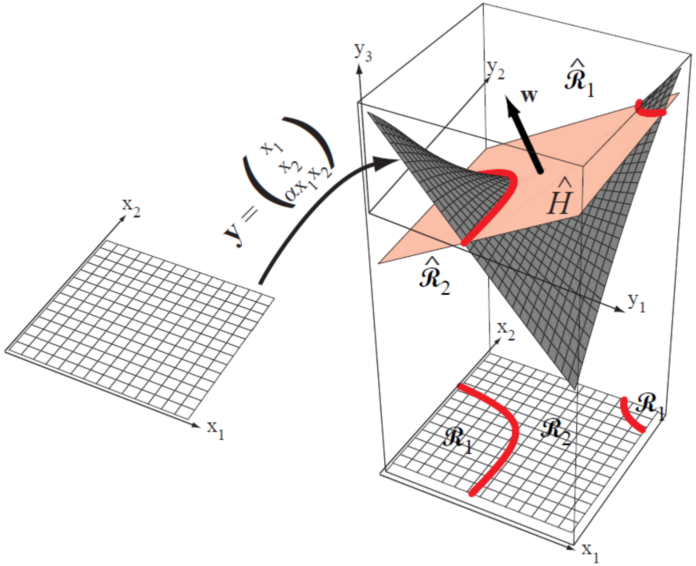


$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + w_0$$

If input space not linearly separable:

- Choose feature function
- Increase dimension





“Pattern Classification“, Duda et al., 2001, Fig. 5.1

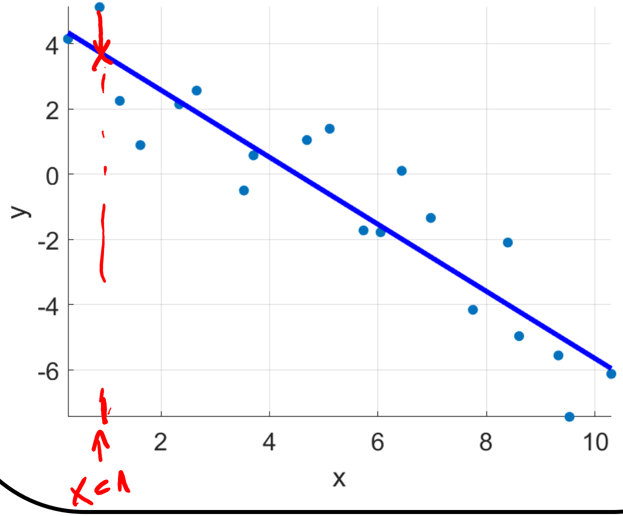
Why kernels? - Motivation

represent data points

$$(x_n, y_n)_{n=1}^{N}$$

by standard regression line

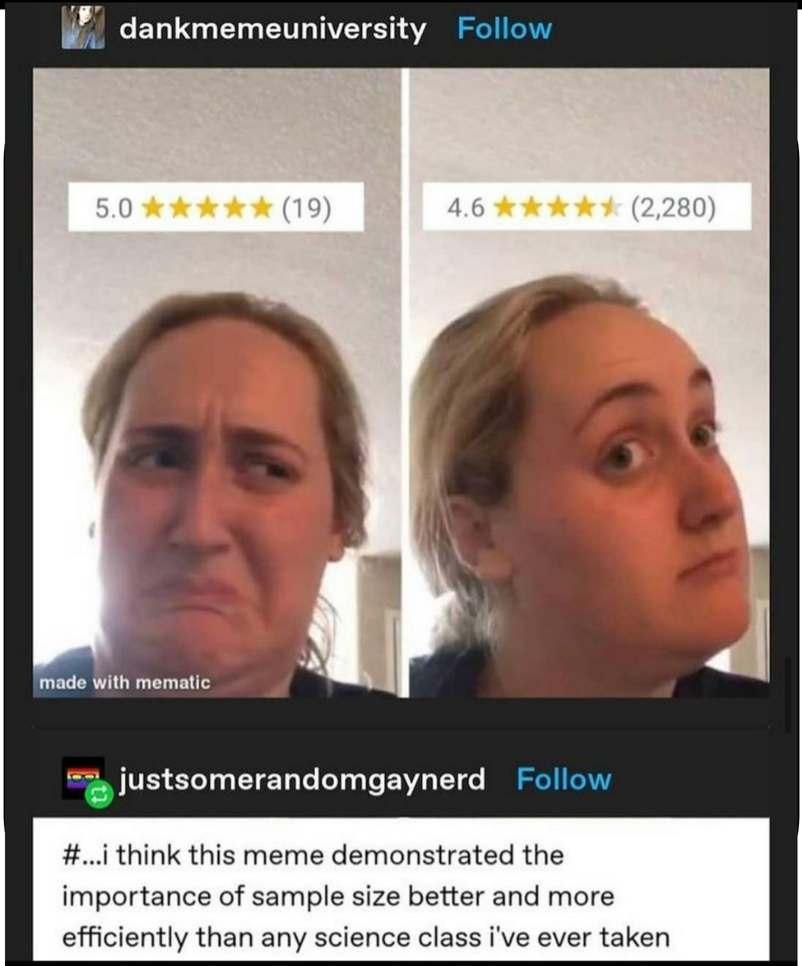
$$f(x) = w_0 + w_1 x$$



- Previously:
  1. Use (training) data to estimate parameters
  2. Discard data
  3. Use model parameters for future predictions

- Now:
  - Keep (subset of) data
  - Use it for future predictions
  - "memory-based"

Why kernels? - Motivation



- Previously:
  1. Use (training) data to estimate parameters
  2. Discard data
  3. Use model parameters for future predictions

- Now:
  - Keep (subset of) data
  - Use it for future predictions
  - “memory-based” approach

Dual Representation

Linear regression model $\boxed{y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x})}$
with least square error:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\boxed{\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)} - t_n\right)^2 + \frac{\lambda}{2}\underbrace{\mathbf{w}^{\mathrm{T}}\mathbf{w}}_{\text{regularizer}}$$

$\|w\|_2^2 = w^T w$

Rewriting

$$\mathbf{w} = \mathbf{\Phi}^{\mathrm{T}}\mathbf{a}$$
$$K = \mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}$$

$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{\Phi}\mathbf{w} - \mathbf{t}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

$$J(\mathbf{a}) = \frac{1}{2}\|\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\mathbf{a} - \mathbf{t}\|_2^2 + \frac{\lambda}{2}\|\mathbf{\Phi}^{\mathrm{T}}\mathbf{a}\|_2^2$$

Dual Representation

Linear regression model $\boxed{y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x})}$
with least square error:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\boxed{\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)} - t_n\right)^2 + \frac{\lambda}{2}\underbrace{\mathbf{w}^{\mathrm{T}}\mathbf{w}}_{\text{regularizer}}$$

Rewriting

$$\mathbf{w} = \mathbf{\Phi}^{\mathrm{T}}\mathbf{a}$$
$$\boldsymbol{K} = \mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}$$

$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{\Phi}\mathbf{w} - \mathbf{t}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

$$J(\mathbf{a}) = \frac{1}{2}\|\boldsymbol{K}\mathbf{a} - \mathbf{t}\|_2^2 + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\boldsymbol{K}\mathbf{a}$$

Why dual form?

Disadvantage?

Previously

$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}), \ \ \mathbf{w} \in \mathbb{R}^M$

invert MxM matrix

$$\mathbf{w} = \left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} + \lambda \boldsymbol{I}_M\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}$$

Now $k_n(\mathbf{x}) = \mathbf{k}(\mathbf{x}_n, \mathbf{x})$

$y(\mathbf{x}) = \mathbf{k}(\mathbf{x})^{\mathrm{T}}\mathbf{a}$

N = # data points

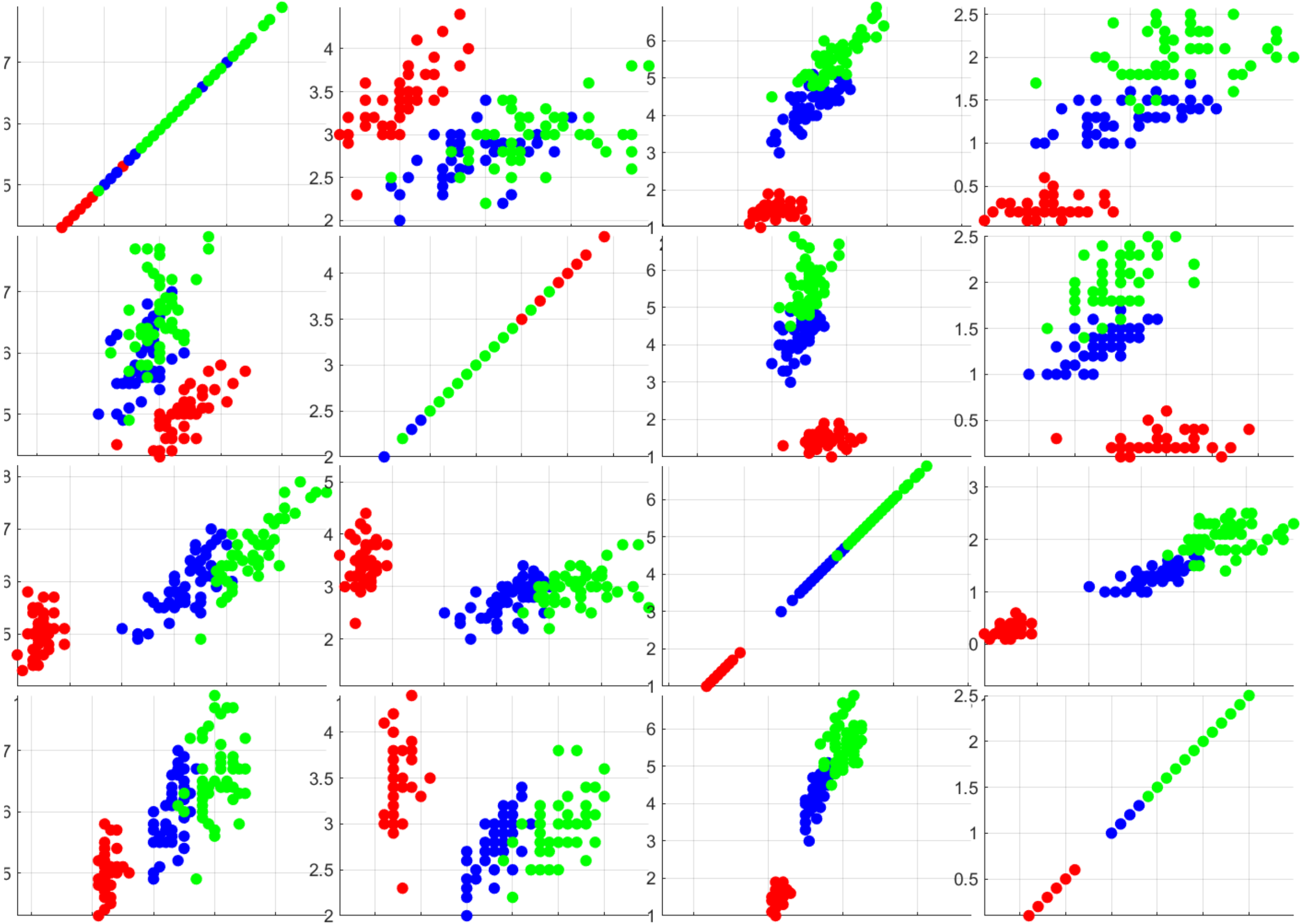
invert NxN matrix

$$\mathbf{a} = \left(\underbrace{\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}}_{\boldsymbol{K}} + \lambda \boldsymbol{I}_N\right)^{-1}\mathbf{t}$$

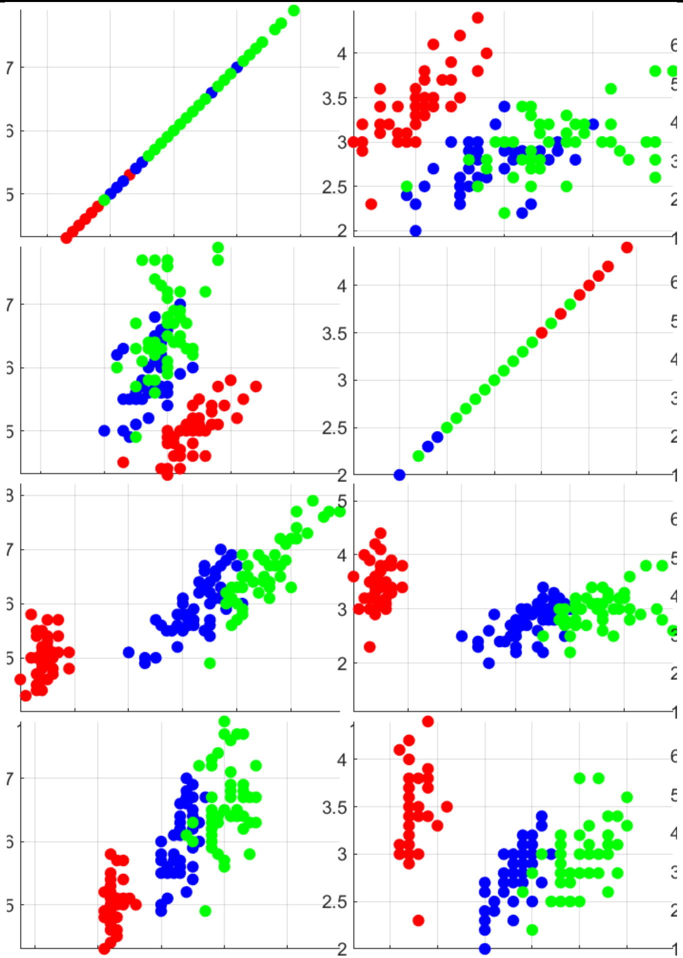
Advantage:

- Only use the kernel representation $K_{nm} = \phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}_m)$
- No need for feature vector
- $\phi(\mathbf{x})$ could even be infinite

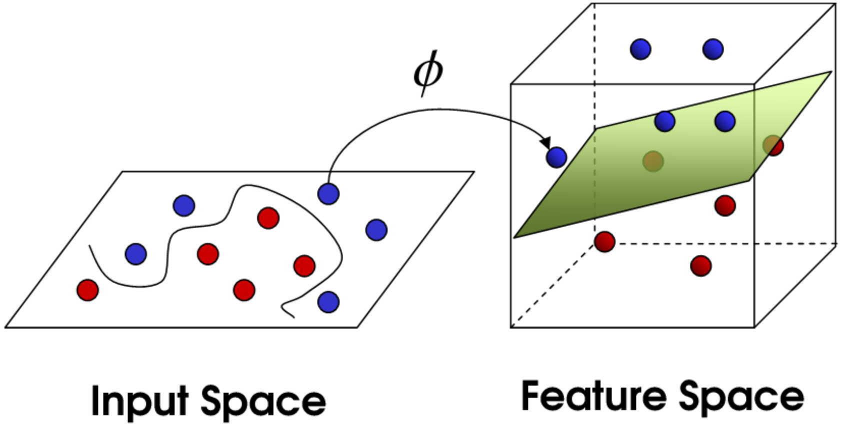
Why kernels? - Iris dataset: 4D features, 3 classes



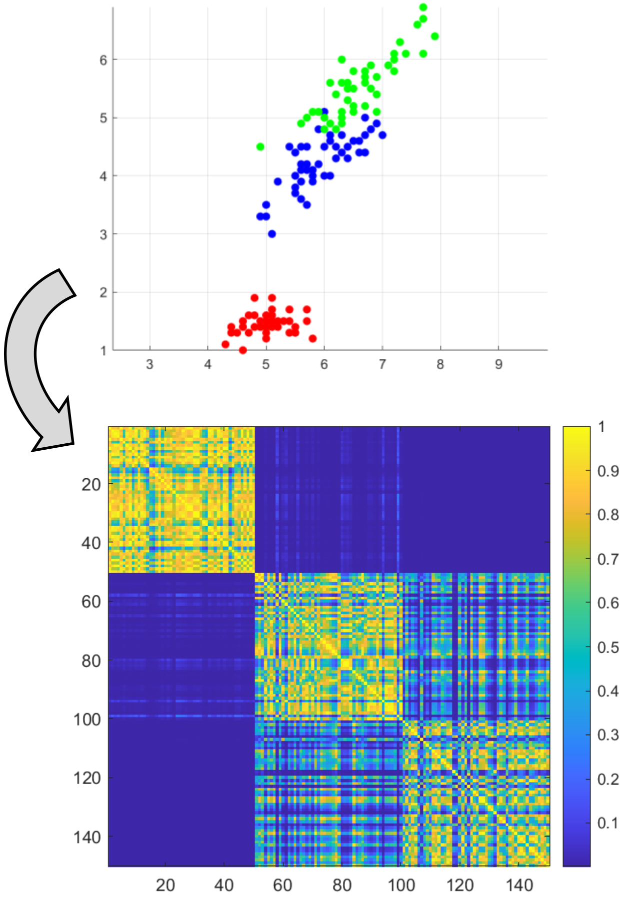
Why kernels? - Iris dataset: 4D features, 3 classes



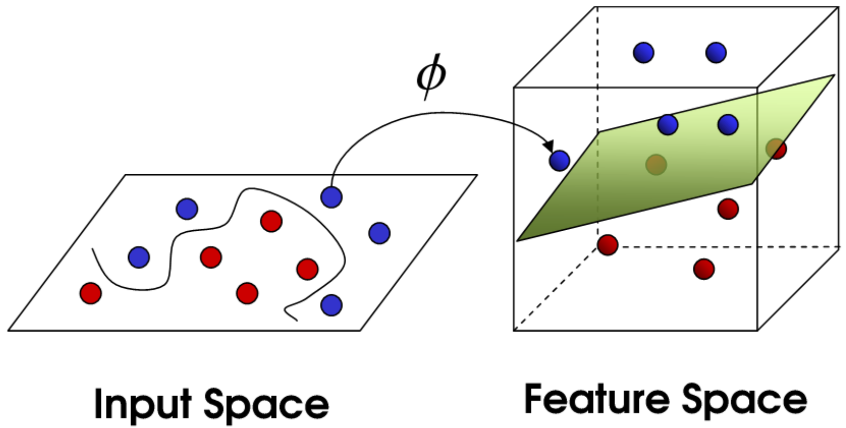
How do we select the best feature space for classification?



Why kernels? - Iris dataset: 4D features, 3 classes

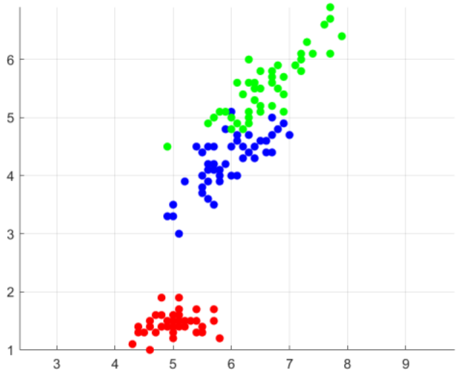


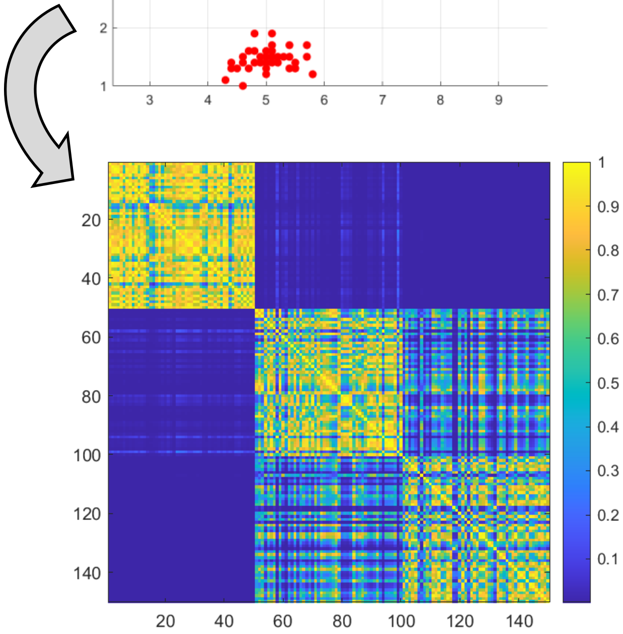
How do we select the best feature space for classification?



Idea: create a similarity measure between data points

How to construct Kernels





What is a valid kernel aka
Gram matrix $\boldsymbol{K}$ with elements $k(\mathbf{x}_n, \mathbf{x}_m)$?

- Symmetric
- Positive semidefinite $x^T K x \geq 0$
- Low for small distance = high similarity
- High for high distance = low similarity

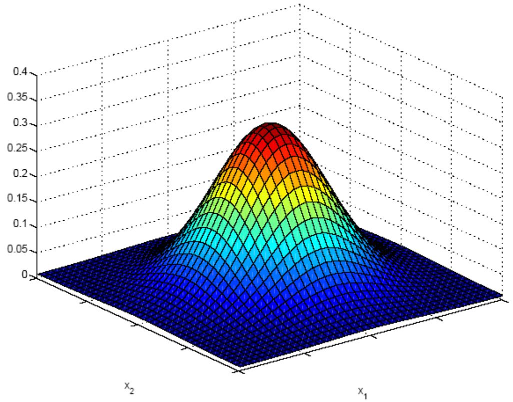
How to build?

- From feature space $\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}$
  $$K_{nm} = \phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}_m)$$
- Build them from simpler kernels …

How to construct Kernels

How to build kernels from simpler kernels

$$k\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \exp\left(\frac{-||\boldsymbol{x} - \boldsymbol{x}'||^2}{2\sigma^2}\right)$$



Techniques for Constructing New Kernels.

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}') \tag{6.13}$$
$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') \tag{6.14}$$
$$k(\mathbf{x}, \mathbf{x}') = q\left(k_1(\mathbf{x}, \mathbf{x}')\right) \tag{6.15}$$
$$k(\mathbf{x}, \mathbf{x}') = \exp\left(k_1(\mathbf{x}, \mathbf{x}')\right) \tag{6.16}$$
$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \tag{6.17}$$
$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}') \tag{6.18}$$
$$k(\mathbf{x}, \mathbf{x}') = k_3\left(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')\right) \tag{6.19}$$
$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}' \tag{6.20}$$
$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) \tag{6.21}$$
$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b) \tag{6.22}$$

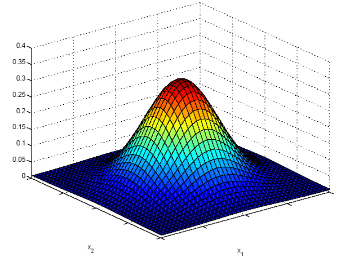
where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\boldsymbol{\phi}(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

Gaussian Processes

- Gaussian processes:
  extend the role of kernels to probabilistic discriminative models
- Recap Bayesian Regression ….

Recap: Bayesian Regression

$$y_n = w_0 + w_1 x_n$$

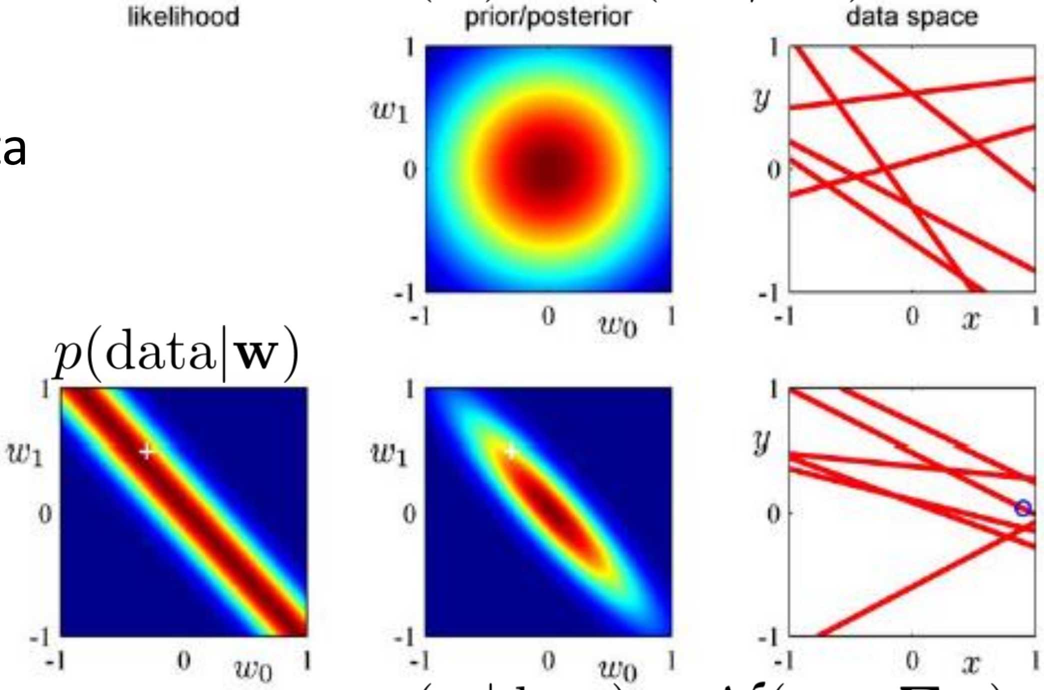


$$p(\mathbf{w}) \sim \mathcal{N}(\mathbf{0}, 1/\alpha \boldsymbol{I})$$

No data
N=0

N=1

$$p(\mathrm{data}|\mathbf{w})$$



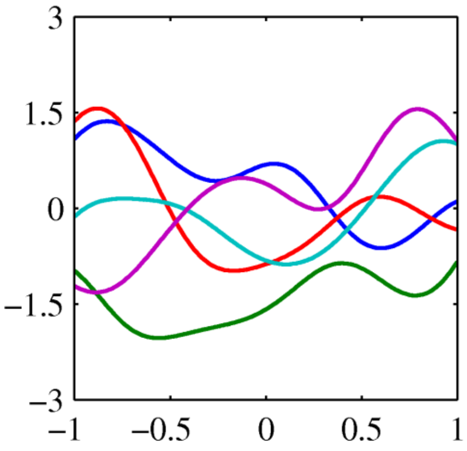
$$p(\mathbf{w}|\mathrm{data}) \sim \mathcal{N}(\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N)$$
$$p(\mathbf{w}|\boldsymbol{X}, \mathbf{t}) \propto p(\boldsymbol{X}, \mathbf{t}|\mathbf{w})p(\mathbf{w})$$

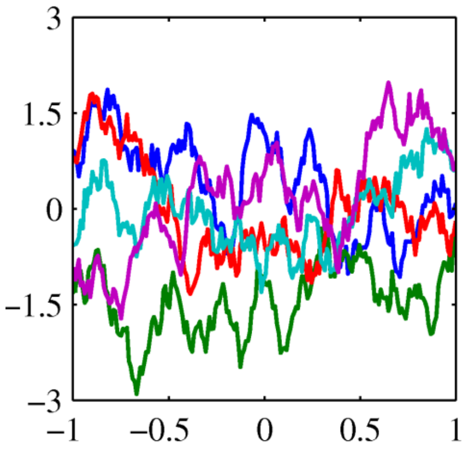
Bishop, “Pattern Recognition“

Posterior =likelihood x prior

Gaussian Process

"a probability distribution over functions $y(\mathbf{x})$ such that the set of values of $y(\mathbf{x})$ evaluated at an arbitrary set of points $\mathbf{x}1, \ldots, \mathbf{x}N$ jointly have a Gaussian distribution."





https://pythonhosted.org/infpy/gps.html

Previously

$$y(\mathbf{x}) = \phi(\mathbf{x})^{\mathrm{T}}\mathbf{w}$$

$$p(\mathbf{w}) \sim \mathcal{N}(\mathbf{0}, 1/\alpha \boldsymbol{I})$$

Now

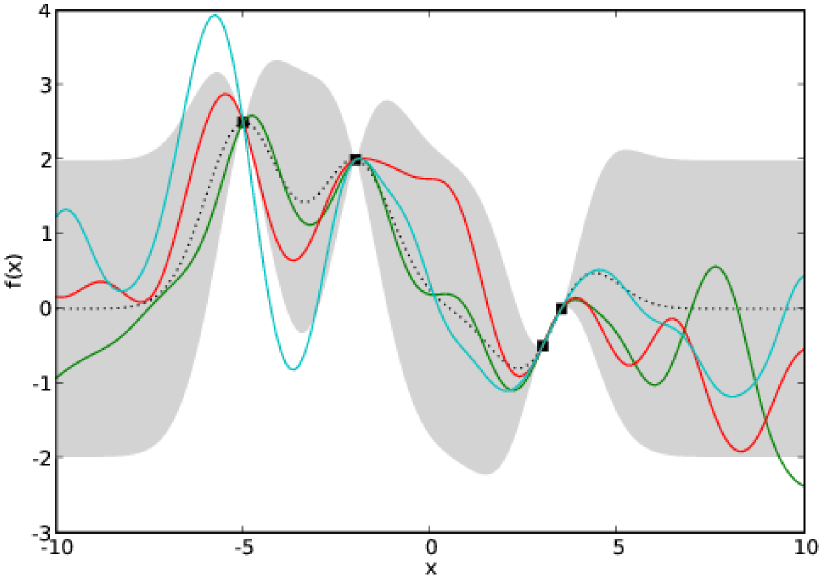
$$\mathbf{y} = \mathbf{\Phi}\mathbf{w}$$

$$\mathbb{E}[\mathbf{y}] = \mathbf{\Phi}\mathbb{E}[\mathbf{w}] = \mathbf{0}$$

$$\mathrm{cov}[\mathbf{y}] = \frac{1}{\alpha}\mathbf{\Phi}\mathbf{\Phi}^T = \boldsymbol{K}$$

Gaussian Process

"a probability distribution over functions $y(\mathbf{x})$ such that the set of values of $y(\mathbf{x})$ evaluated at an arbitrary set of points $\mathbf{x}1, \ldots, \mathbf{x}N$ jointly have a Gaussian distribution."



https://pythonhosted.org/infpy/gps.html

Previously

$$y(\mathbf{x}) = \phi(\mathbf{x})^{\mathrm{T}}\mathbf{w}$$

$$p(\mathbf{w}) \sim \mathcal{N}(\mathbf{0}, 1/\alpha \boldsymbol{I})$$

Now

$$\mathbf{y} = \boldsymbol{\Phi}\mathbf{w}$$

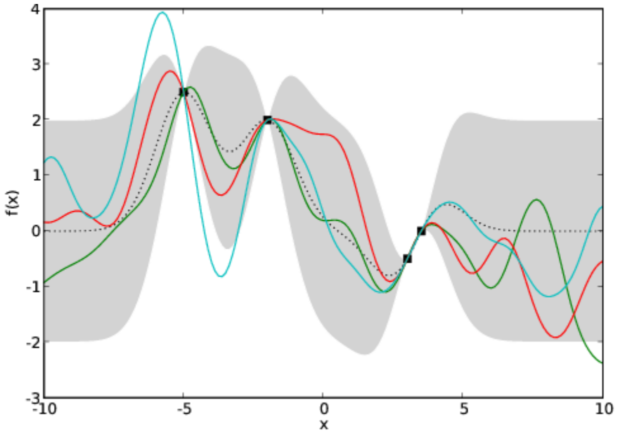
$$\mathbb{E}[\mathbf{y}] = \boldsymbol{\Phi}\mathbb{E}[\mathbf{w}] = \mathbf{0}$$

$$\operatorname{cov}[\mathbf{y}] = \frac{1}{\alpha}\boldsymbol{\Phi}\boldsymbol{\Phi}^T = \boldsymbol{K}$$

Gaussian Process – for Regression

$$\mathbb{E}[\mathbf{y}] = \mathbf{0}$$

$$\operatorname{cov}[\mathbf{y}] = \boldsymbol{K}$$



- Bayesian approach
- Fully defined by expectation value and covariance matrix
- Kernel function can be defined directly
- Model the target variable directly

$$y_n = w_0 + w_1 x_n$$

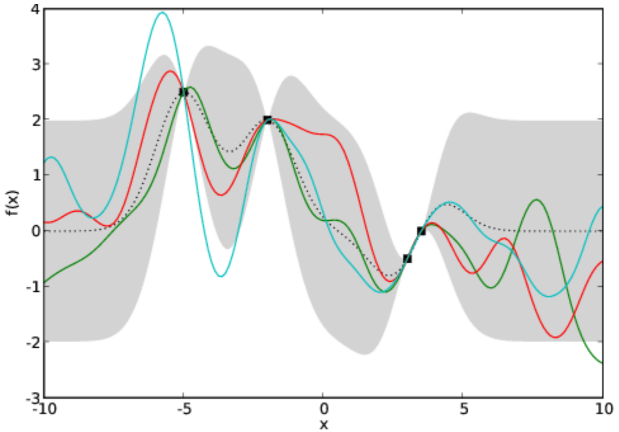
$$t_n = y_n + \epsilon_n$$

$$p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1})$$

Gaussian Process – for Regression

$$\mathbb{E}[\mathbf{y}] = \mathbf{0}$$

$$\mathrm{cov}[\mathbf{y}] = \boldsymbol{K}$$



- Bayesian approach
- Fully defined by expectation value and covariance matrix
- Kernel function can be defined directly
- Model the target variable directly

$$y_n = w_0 + w_1 x_n$$

$$t_n = y_n + \epsilon_n$$

$$p(\boldsymbol{t}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{t}|\boldsymbol{y}, \beta^{-1}\boldsymbol{I}_N)$$

$$p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \boldsymbol{K})$$

$$p(\boldsymbol{t}) = \mathcal{N}(\boldsymbol{t}|\mathbf{0}, \boldsymbol{C})$$

Gaussian Process – for Regression

Goal:
predict targets for new samples

$$\mathbf{t}_{N+1} = (t_1, \ldots, t_{N+1})^{\mathrm{T}}$$

$$\begin{bmatrix} C & v \\ v^T & k(x_{N+1}, x_{N+1}) \end{bmatrix}$$

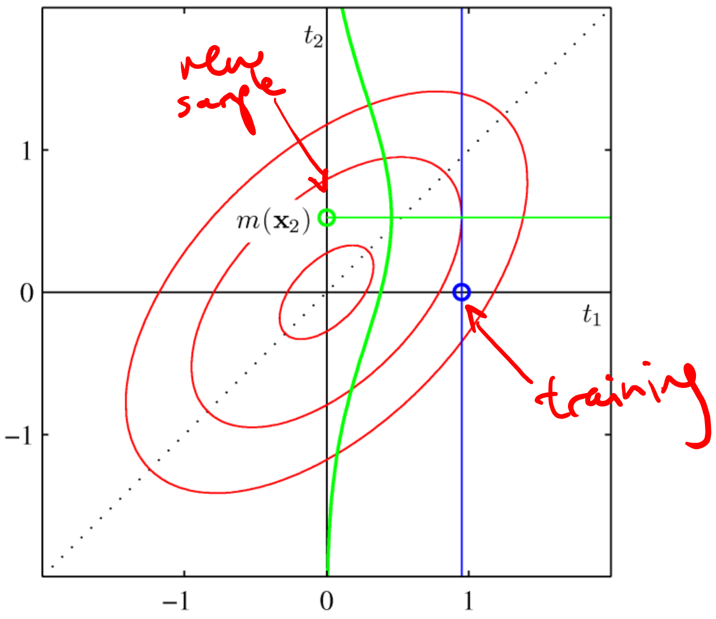
$$p(\mathbf{t}) = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C})$$

$$p(\mathbf{t}_{N+1}) = \mathcal{N}(\mathbf{t}_{N+1}|\mathbf{0}, \mathbf{C}_{N+1})$$

$$p(t_{N+1}|\mathbf{t}) = \mathcal{N}(m, \sigma^2)$$

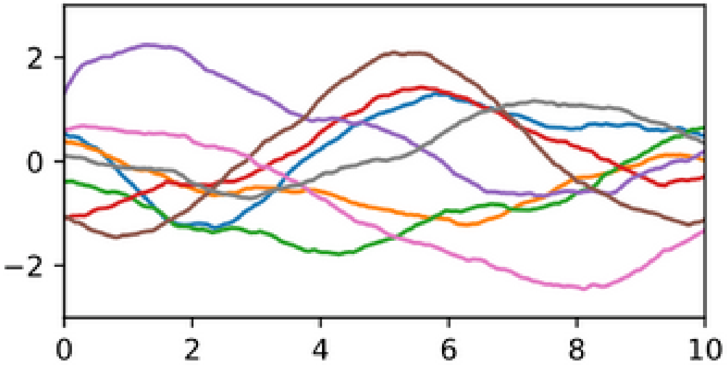
$v_i = k(x_i, x_{N+1})$

x_{N+1}

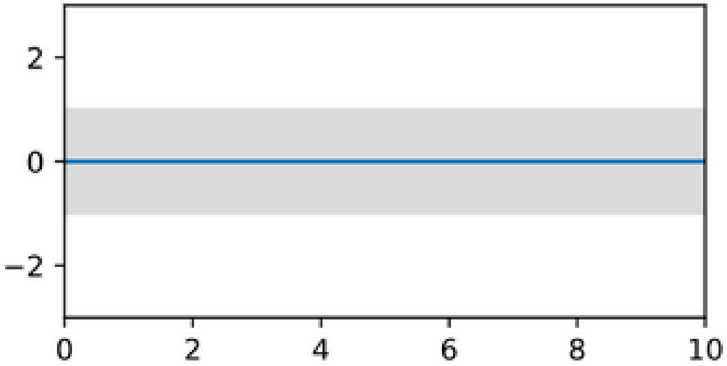


Gaussian Process – for Regression

$$p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, \boldsymbol{K})$$



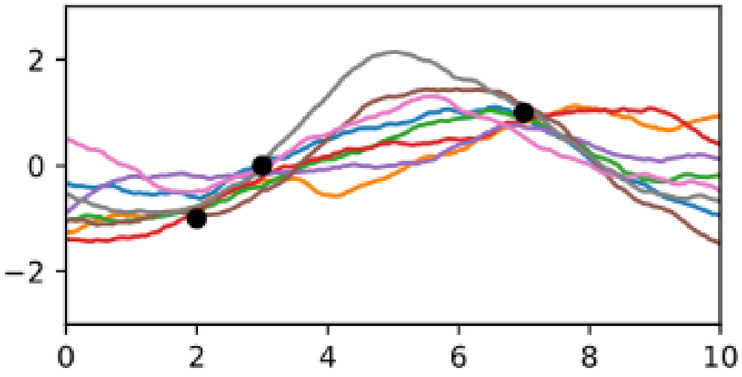
Random samples from the prior

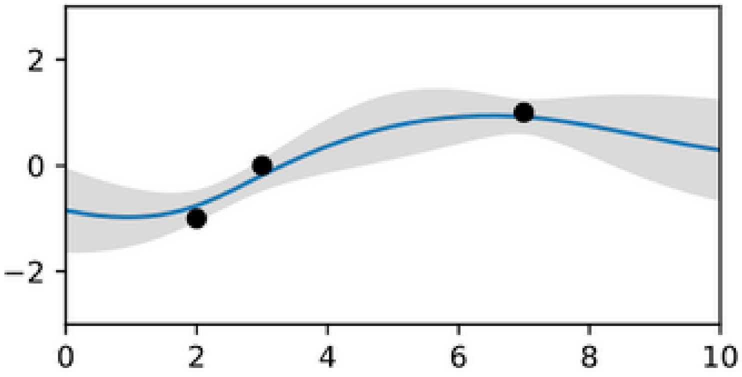


Expectation
= mean of samples

wikipedia

Gaussian Process – for Regression





A-posteriori:

Prior and evidence

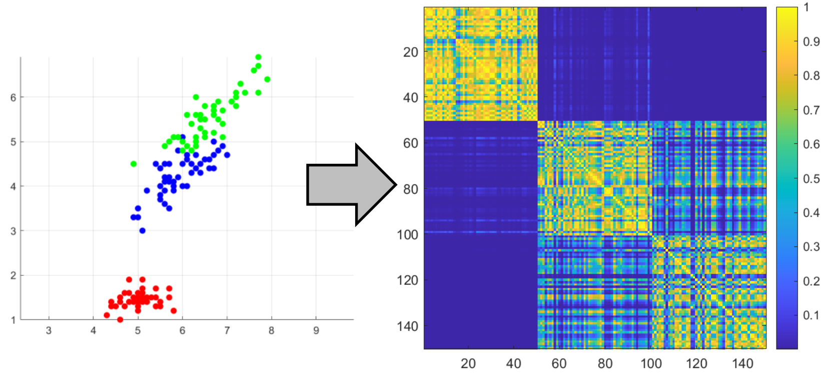
data

Expectation

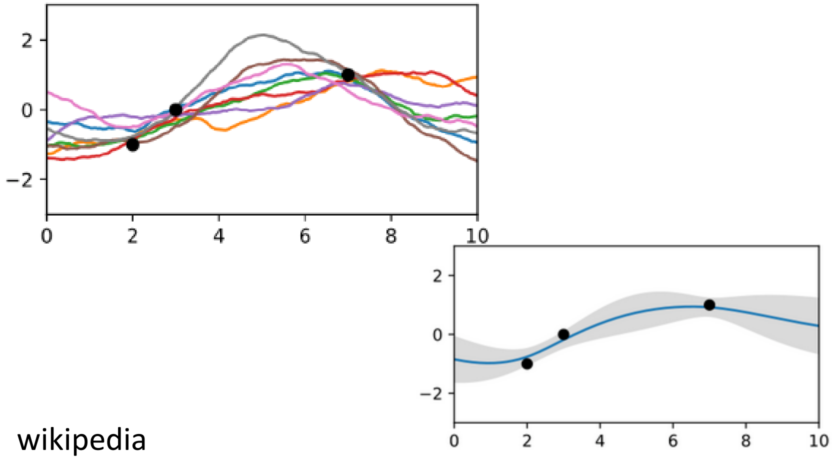
= mean of samples

wikipedia

Summary - Kernel Methods



- **kernel** $k(\mathbf{x}_n, \mathbf{x}_m)$ represents similarity between samples
- no need for function $\phi(\mathbf{x})$
- different modalities, data representation possible



wikipedia

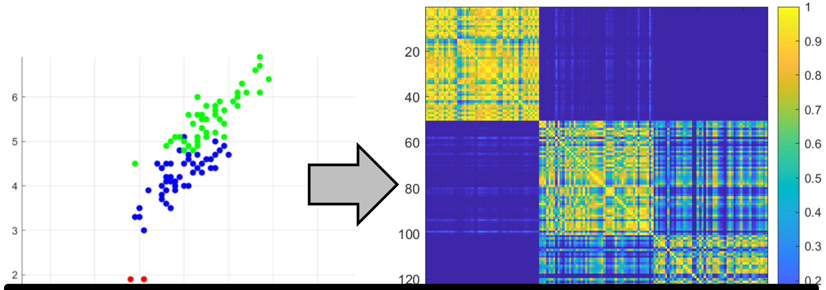
Gaussian Processes:

- model distribution of target variable directly
- uncertainty

$$p(\boldsymbol{t}) = \mathcal{N}(\boldsymbol{t}|\mathbf{0}, \boldsymbol{C})$$

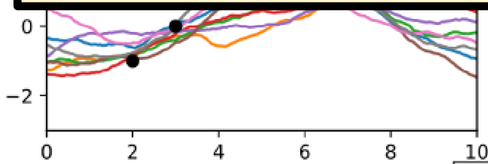
$$p(t_{N+1}|\boldsymbol{t}) = \mathcal{N}(m, \sigma^2)$$

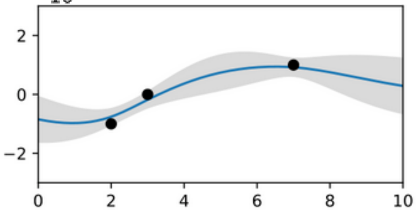
Summary - Kernel Methods



- **kernel** $k(\mathbf{x}_n, \mathbf{x}_m)$ represents similarity between samples
- no need for function $\phi(\mathbf{x})$
- different modalities, data representation possible

Drawback:
Kernel must be evaluated on all combinations of datapoints





wikipedia

Gaussian Processes:

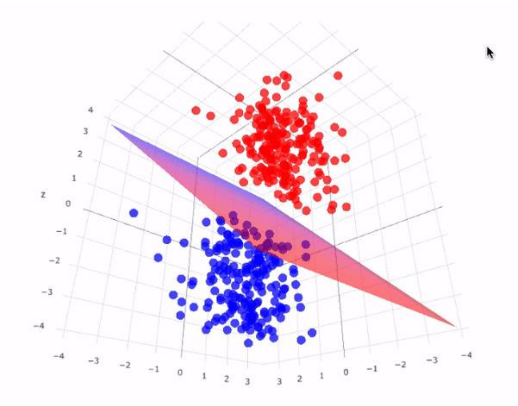
- model distribution of target variable directly
- uncertainty

$$p(\boldsymbol{t}) = \mathcal{N}(\boldsymbol{t}|\mathbf{0}, \boldsymbol{C})$$

$$p(t_{N+1}|\boldsymbol{t}) = \mathcal{N}(m, \sigma^2)$$

Recap: Linear Classifier

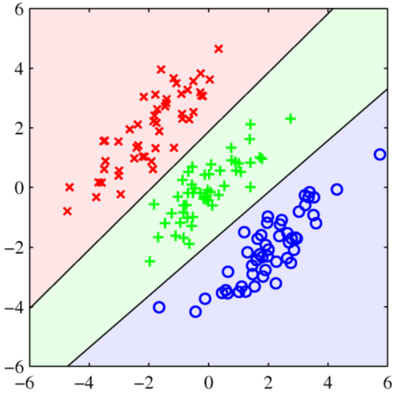
K=2 classes



$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + w_0$$

$$y(\mathbf{x}) \begin{cases} \geq 0 & , \mathbf{x} \in \mathcal{C}_1 \\ < 0 & , \mathbf{x} \in \mathcal{C}_2 \end{cases}$$

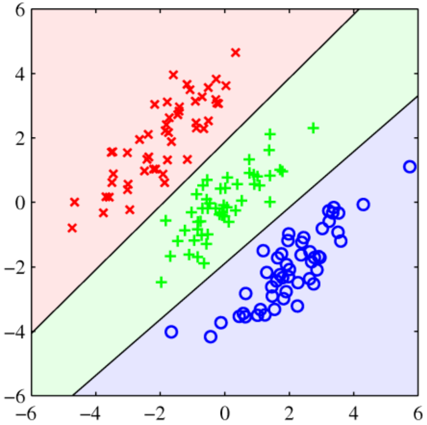
K>2 classes

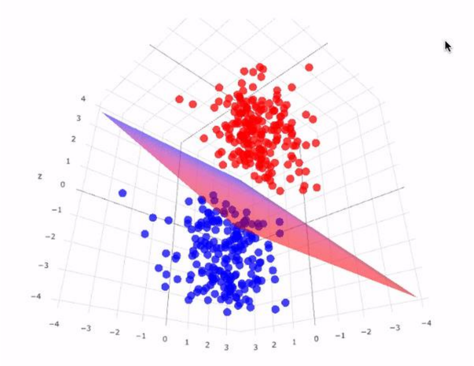


$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\phi(\mathbf{x}) + w_{k0}$$

$$m = \arg\max_k \; y_k(\mathbf{x})$$

Recap: Linear Classifier





Linear Discriminant Functions

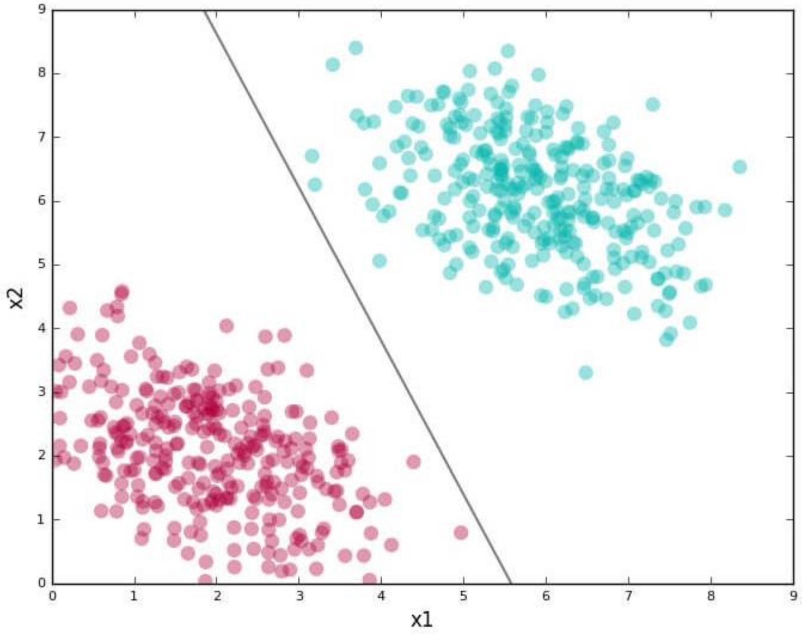
$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0} \qquad m = \arg\max_k \, y_k(\mathbf{x})$$

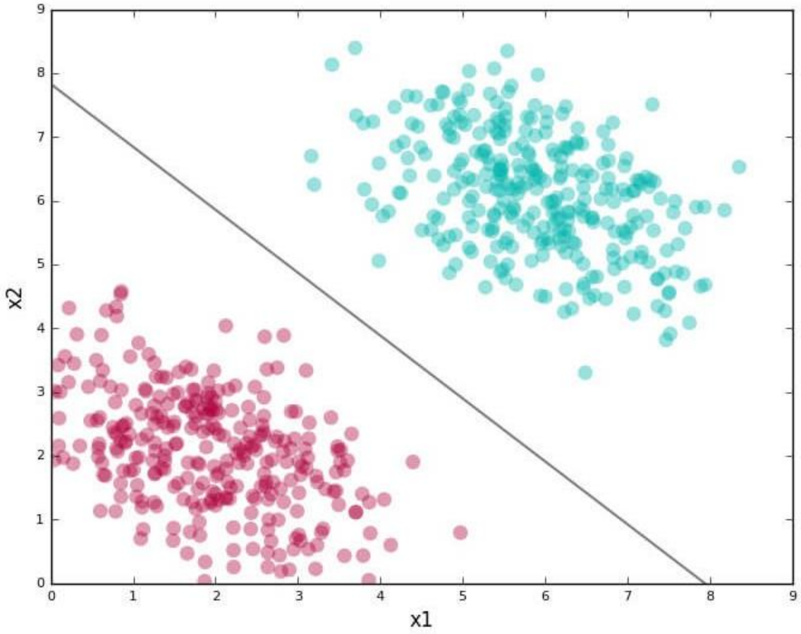
1. Least Squares
   - minimize distance between estimate and true
   - Direct unique solution
2. Fisher’s Discriminant
   - Maximize distance between classes
   - Minimize distance within class
3. Perceptron: K=2 only
   - Minimize misclassified samples
   - One sample at a time, iterative
   - No unique solution

https://medium.com/nyu-a3sr-data-science-team/support-vector-machines-and-wine-cef59ad38b41

Separating classes by one line

Which is better?

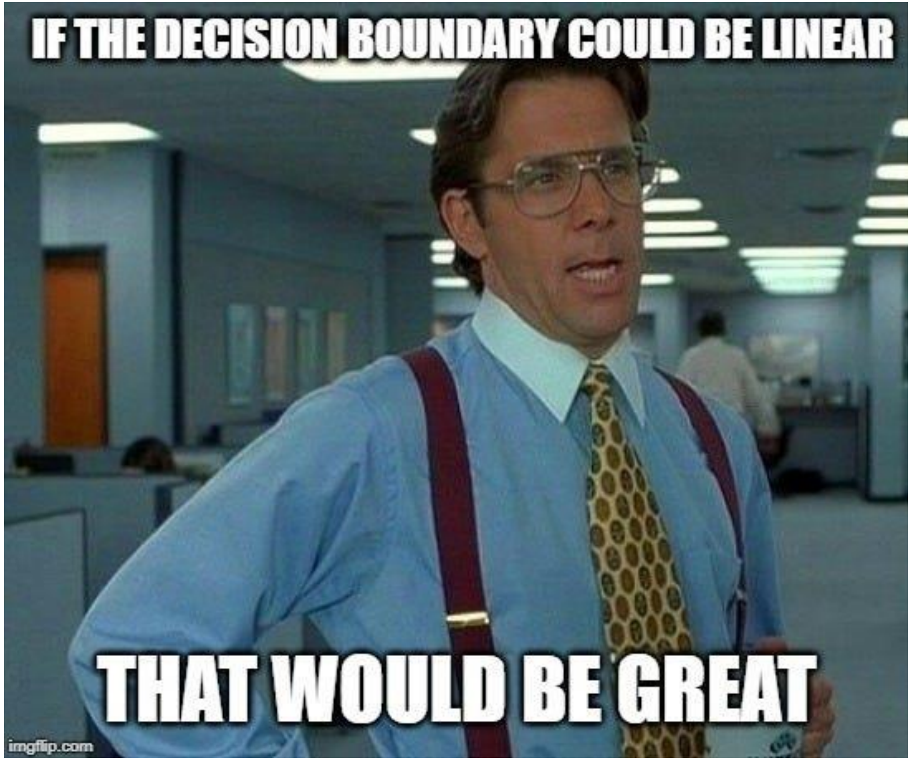


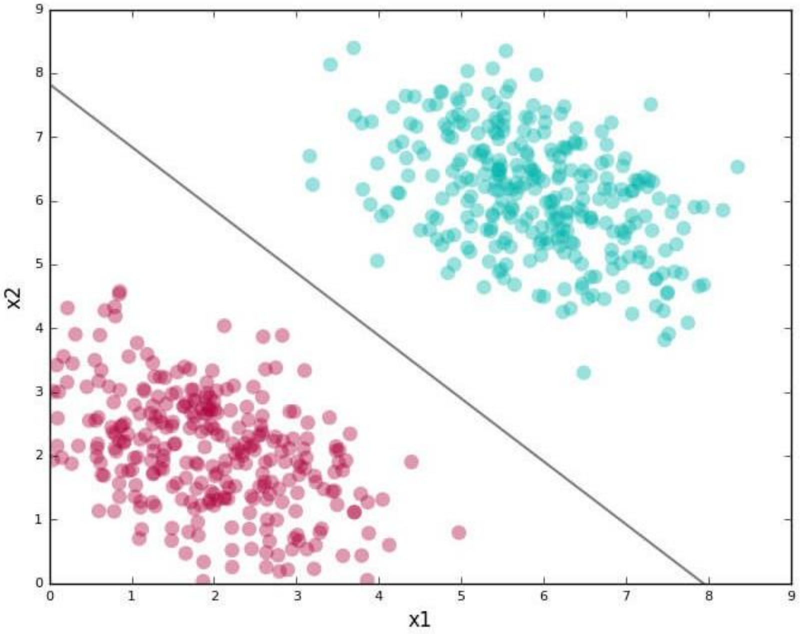


https://blog.statsbot.co/support-vector-machines-tutorial-c1618e635e93

Separating classes by one line

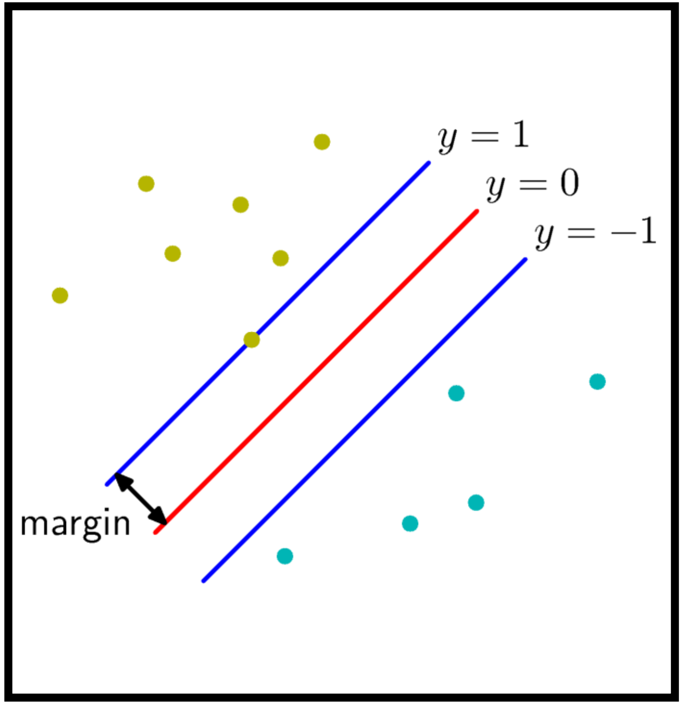
Which is better?

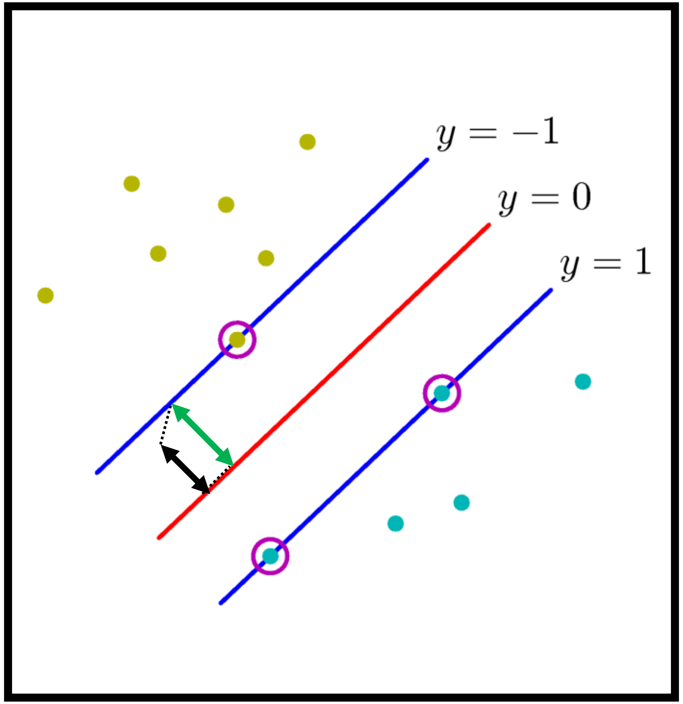




https://blog.statsbot.co/support-vector-machines-tutorial-c1618e635e93

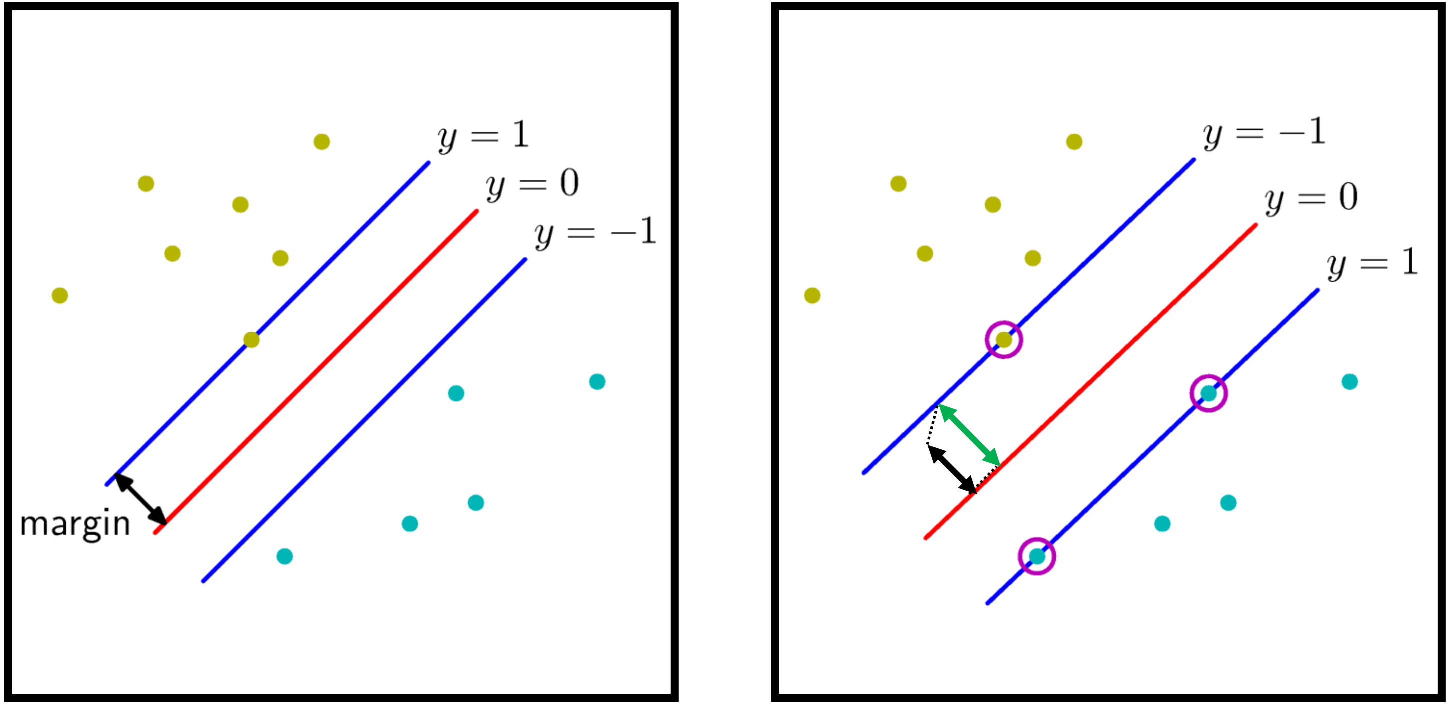
Support Vector Machines (SVMs)





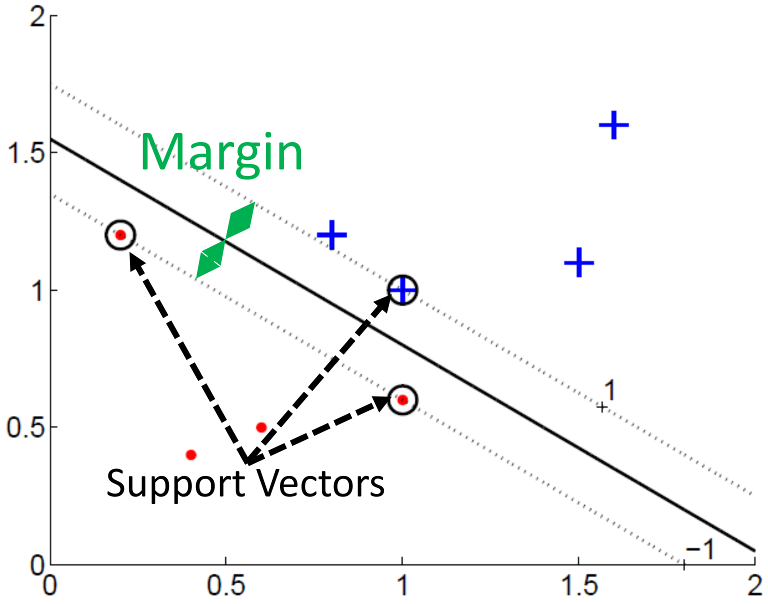
Goal: Maximize margin

Support Vector Machines (SVMs)



Goal: Maximize margin

Support Vector Machines (SVMs)



$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + w_0$$

Support Vector Machines (SVMs)
optimal separating hyperplane:

- defined by few training samples: **support vectors**
- separates two classes
- maximizes $\boxed{\text{margin}}$

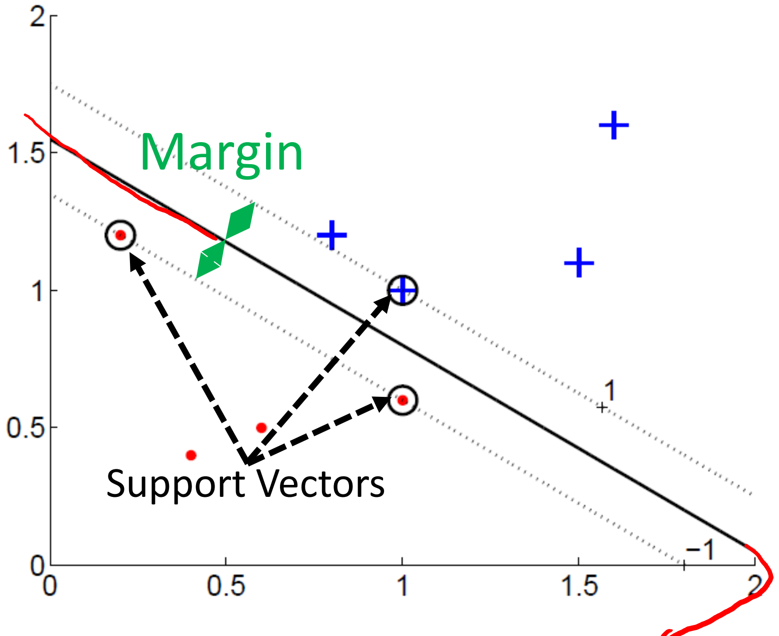
Step 1:
Assume classes are perfectly linearly separable

Alpaydin, “Introduction to Machine Learning”

Support Vector Machines (SVMs)

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + w_0 \begin{cases} > 0 & , \ t = +1 \\ \leq 0 & , \ t = -1 \end{cases}$$

$$\Rightarrow y(\mathbf{x}_n) = \underbrace{t_n}_{-1} \underbrace{\left(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + w_0\right)}_{\leq 0 \ \Rightarrow \text{pos.}} > 0$$



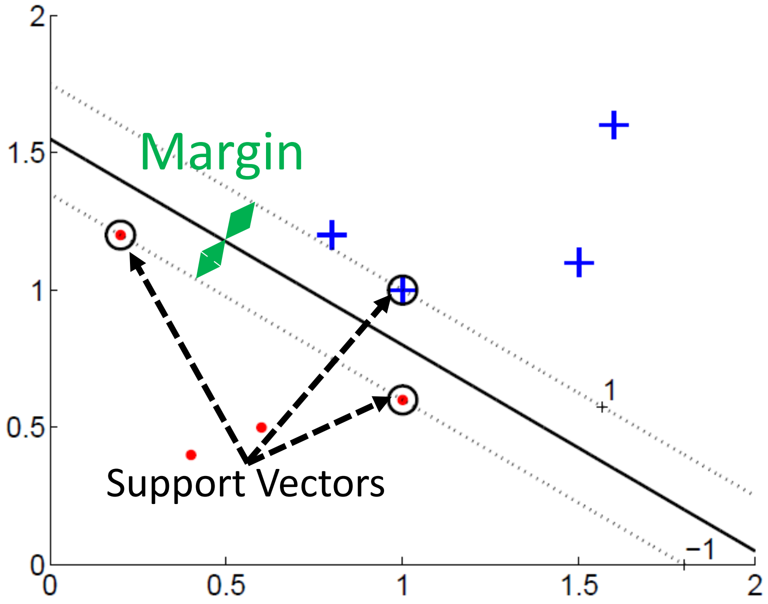
$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + w_0 = 0$$

Distance point $\mathbf{x}^*$ to hyperplane

$$\frac{|y(\mathbf{x}^*)|}{\|\mathbf{w}\|} = \frac{\left|\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}^*) + w_0\right|}{\|\mathbf{w}\|}$$

Alpaydin, “Introduction to Machine Learning”

Support Vector Machines (SVMs)



$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + w_0$$

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + w_0 \begin{cases} > 0 & , \; t = +1 \\ \leq 0 & , \; t = -1 \end{cases}$$

$$\Rightarrow y(\mathbf{x}_n) = t_n \left(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + w_0\right) > 0$$

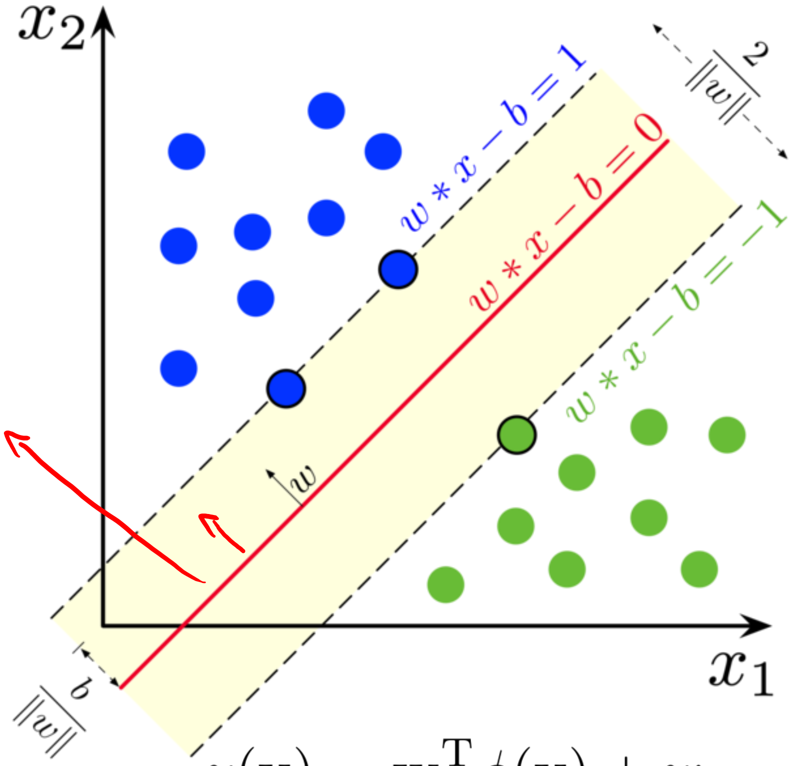
Distance point $\mathbf{x}^*$ to hyperplane

$$\boxed{\frac{|y(\mathbf{x}_n)|}{\|\mathbf{w}\|}} = \frac{\left|\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + w_0\right|}{\|\mathbf{w}\|}$$

$$= \frac{t_n\left(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + w_0\right)}{\|\mathbf{w}\|}$$

Alpaydin, “Introduction to Machine Learning”

Support Vector Machines (SVMs)



$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + w_0$$

Maximize the margin:

maximize

$$\max \frac{t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + w_0)}{\|\mathbf{w}\|}$$

minimize

$$\Rightarrow \min \|\mathbf{w}\|_2^2$$

For the support vectors:

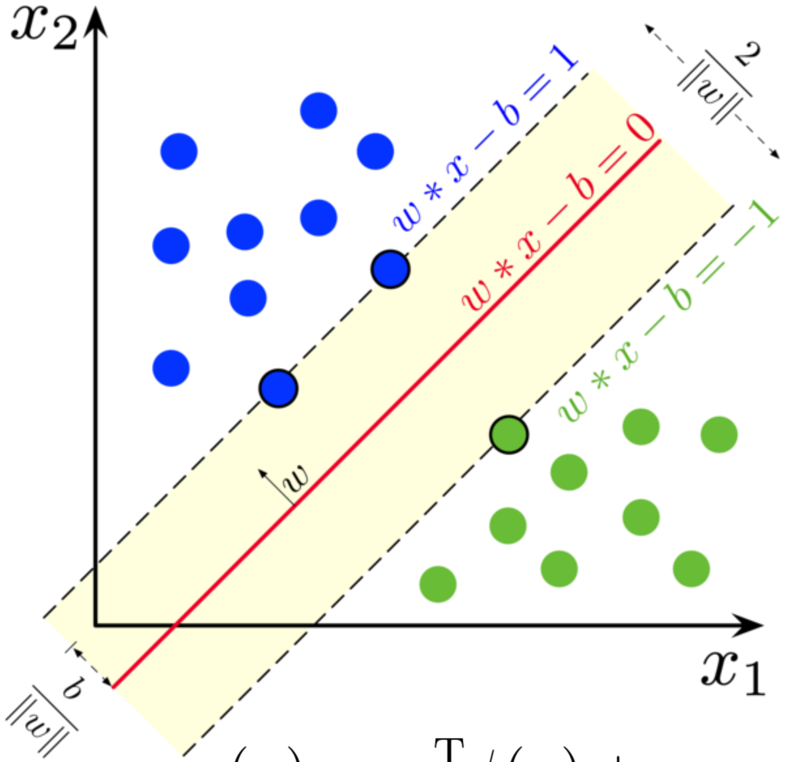
$$t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n^*) + w_0) = 1$$

For all samples:

$$t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + w_0) \geq 1$$

wikipedia

Support Vector Machines (SVMs)



$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \phi(\mathbf{x}) + w_0$$

Maximize the margin:

- Min norm $\min \|\mathbf{w}\|_2^2$
- Such that

$$t_n(\mathbf{w}^{\mathrm{T}} \phi(\mathbf{x}_n) + w_0) \geq 1$$

$$\underset{\mathbf{w}, w_0}{\arg\min} \frac{1}{2} \|\mathbf{w}\|_2^2$$

$$\text{subject to } t_n(\mathbf{w}^{\mathrm{T}} \phi(\mathbf{x}_n) + w_0) \geq 1, \ \forall n$$

Constrained optimization problem
=> Solve by Lagrange

wikipedia

Max Margin Problem

$$\arg\min_{\mathbf{w},w_0} \frac{1}{2}\|\mathbf{w}\|_2^2 \quad \text{subject to } t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)+w_0) \geq 1,\ \forall n$$

Lagrange function with Lagrange multipliers $a_n \geq 0$

$$\min L(\mathbf{w}, w_0, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|_2^2 - \sum_{n=1}^{N} a_n \left(t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)+w_0)-1\right)$$

Max Margin Problem

$$\arg\min_{\mathbf{w},w_0} \frac{1}{2}\|\mathbf{w}\|_2^2 \quad \text{subject to } t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + w_0) \geq 1, \ \forall n$$

Lagrange function with Lagrange multipliers $a_n \geq 0$

$$\min L(\mathbf{w}, w_0, \mathbf{a}) = \frac{1}{2}\underbrace{\|\mathbf{w}\|_2^2}_{w^T w} - \sum_{n=1}^{N} a_n t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + w_0) + \sum_{n=1}^{N} a_n$$

$$\nabla_{\mathbf{w}} L(\mathbf{w}, w_0, \mathbf{a}) = \mathbf{w} - \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n) \stackrel{!}{=} 0 \quad \Rightarrow \mathbf{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n)$$

$$\frac{\partial}{\partial w_0} L(\mathbf{w}, w_0, \mathbf{a}) = -\sum_{n=1}^{N} a_n t_n \stackrel{!}{=} 0 \quad \Rightarrow \sum_{n=1}^{N} a_n t_n = 0$$

Max Margin Problem

$$\arg\min_{\mathbf{w},w_0} \frac{1}{2}\|\mathbf{w}\|_2^2 \quad \text{subject to } t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)+w_0) \geq 1,\ \forall n$$

Lagrange function with Lagrange multipliers $a_n \geq 0$

$$\min L(\mathbf{w},w_0,\mathbf{a}) = \frac{1}{2}\|\boxed{\mathbf{w}}\|_2^2 - \sum_{n=1}^{N} a_n t_n \boxed{\mathbf{w}}^{\mathrm{T}}\phi(\mathbf{x}_n) - w_0 \boxed{\sum_{n=1}^{N} a_n t_n} + \sum_{n=1}^{N} a_n$$

$$\nabla_{\mathbf{w}} L(\mathbf{w},w_0,\mathbf{a}) = \mathbf{w} - \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n) \overset{!}{=} 0 \quad \Rightarrow \boxed{\mathbf{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n)}$$

$$\frac{\partial}{\partial w_0} L(\mathbf{w},w_0,\mathbf{a}) = -\sum_{n=1}^{N} a_n t_n \overset{!}{=} 0 \quad \Rightarrow \boxed{\sum_{n=1}^{N} a_n t_n = 0}$$

Max Margin Problem

$$\arg\min_{\mathbf{w},w_0} \frac{1}{2}\|\mathbf{w}\|_2^2 \quad \text{subject to } t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)+w_0) \geq 1,\ \forall n$$

Lagrange function with Lagrange multipliers $a_n \geq 0$

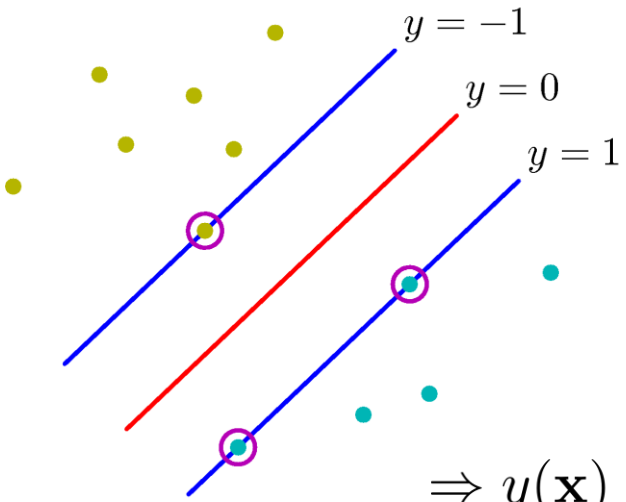
$$\min L(\mathbf{w}, w_0, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|_2^2 - \sum_{n=1}^{N} a_n t_n \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) - w_0 \sum_{n=1}^{N} a_n t_n + \sum_{n=1}^{N} a_n$$

Dual representation

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m \overbrace{k(\mathbf{x}_n, \mathbf{x}_m)}^{\phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}_m)}$$

With constraints: $a_n \geq 0 \qquad \sum_{n=1}^{N} a_n t_n = 0$

Kernel SVM: How to classify new datapoints



$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + w_0 \begin{cases} > 0 & , \; t = +1 \\ \leq 0 & , \; t = -1 \end{cases}$$

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n)$$

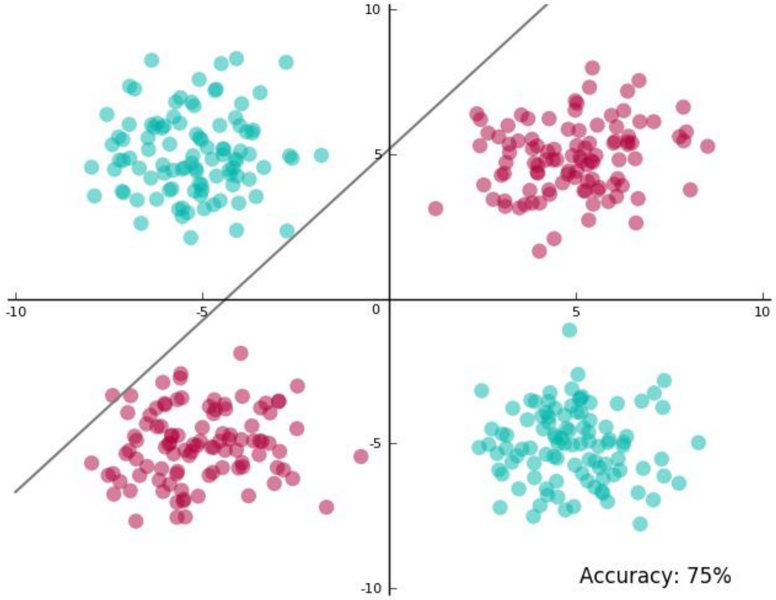
$$\Rightarrow y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}_n, \mathbf{x}) + w_0 \begin{cases} > 0 & , \; t = +1 \\ \leq 0 & , \; t = -1 \end{cases}$$

- only for the small set of support vectors $a_n > 0$
- yet unknown: a_n, w_0

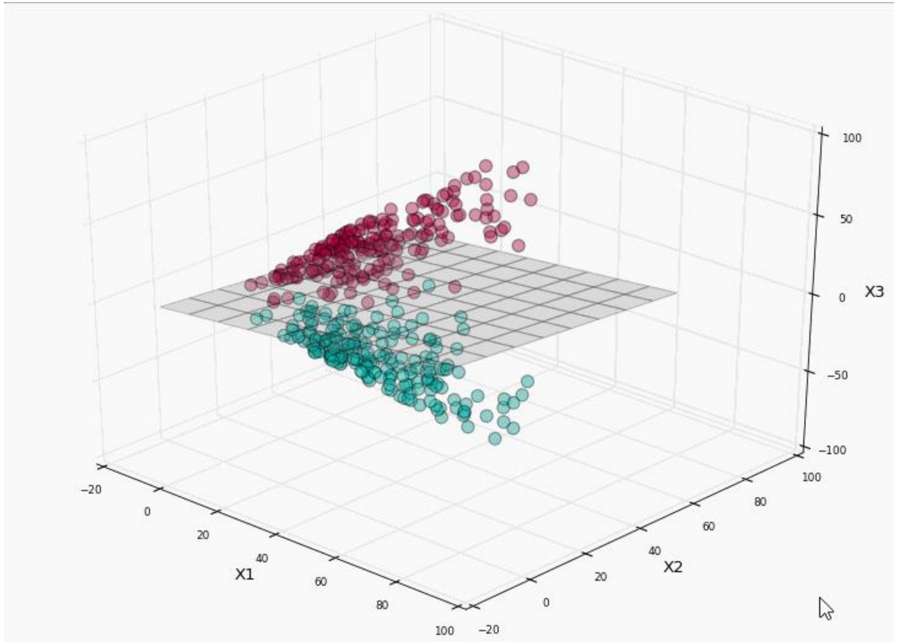
Support Vector Machines (SVMs)

Transform data space with an appropriate basis function $\phi(\mathbf{x})$

before



after



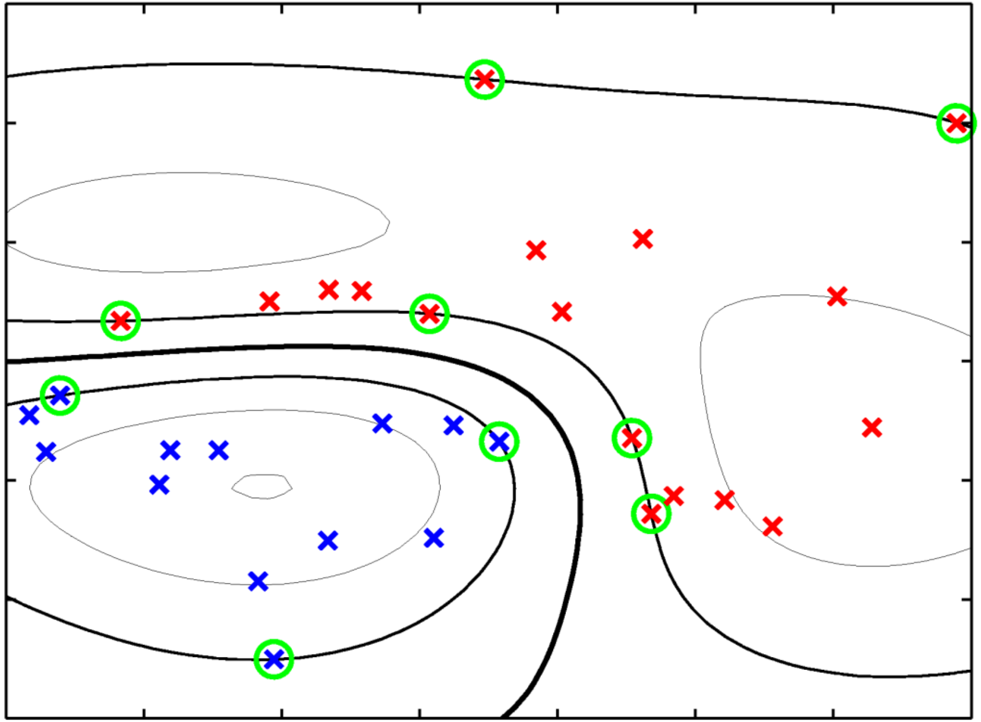
https://blog.statsbot.co/support-vector-machines-tutorial-c1618e635e93

kSVM

Radial Basis Kernel (RBF)

Gaussian Kernel

$$k(\mathbf{x}_m, \mathbf{x}_n) = \exp\left(-\frac{\|\mathbf{x}_m - \mathbf{x}_n\|_2^2}{2s^2}\right)$$

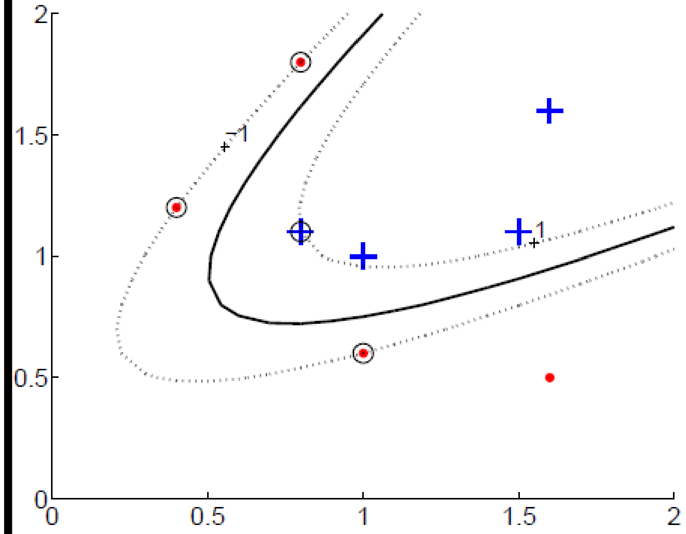


Support vectors

kSVM

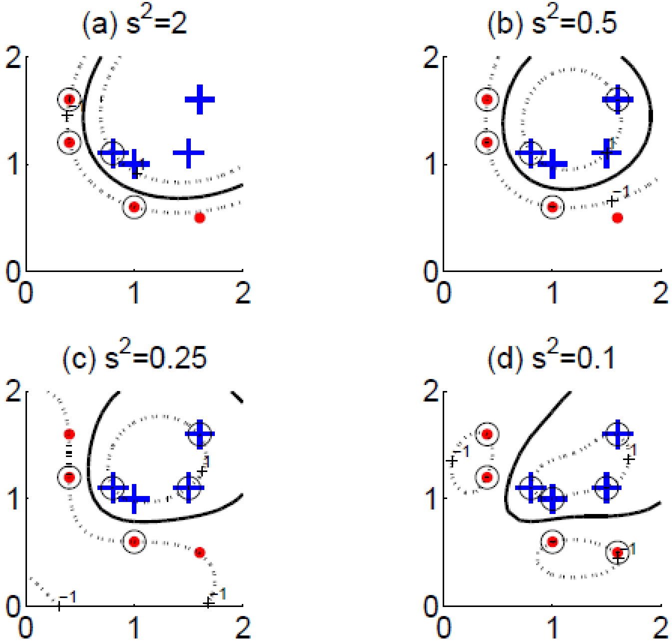
Polynomial

$$k(\mathbf{x}_m, \mathbf{x}_n) = \left(\mathbf{x}_m^{\mathrm{T}}\mathbf{x}_n + c\right)^q$$



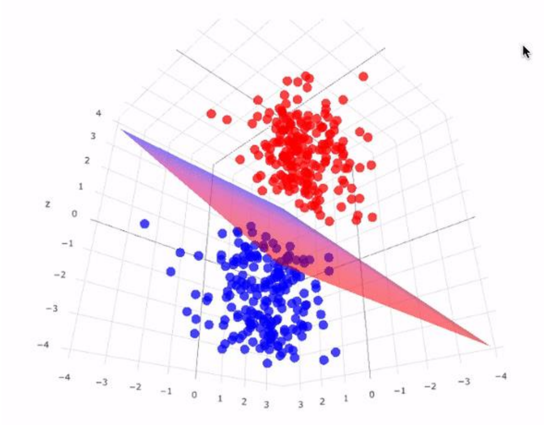
Gaussian

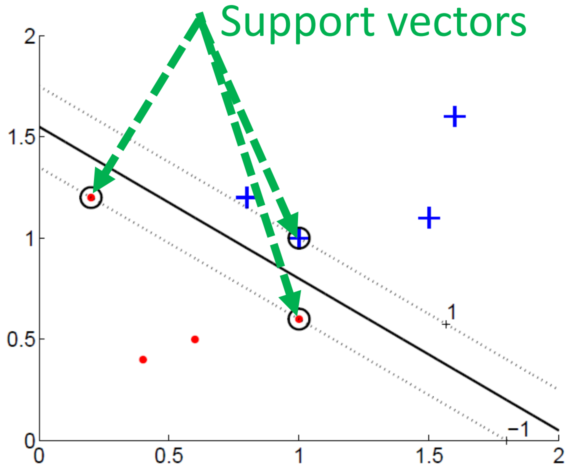
$$k(\mathbf{x}_m, \mathbf{x}_n) = \exp\left(-\frac{\|\mathbf{x}_m - \mathbf{x}_n\|_2^2}{2s^2}\right)$$



Alpaydin, “Introduction to Machine Learning”

SVM so far ...





- Data points $\mathbf{x}_n \in \mathbb{R}^D$ with labels $t_n \in \{-1, +1\}$
- SVM:
  (1) find support vectors $\alpha_n > 0$
  they define ....
  (2) the parameters $\mathbf{w}, \ w_0$

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + w_0$$

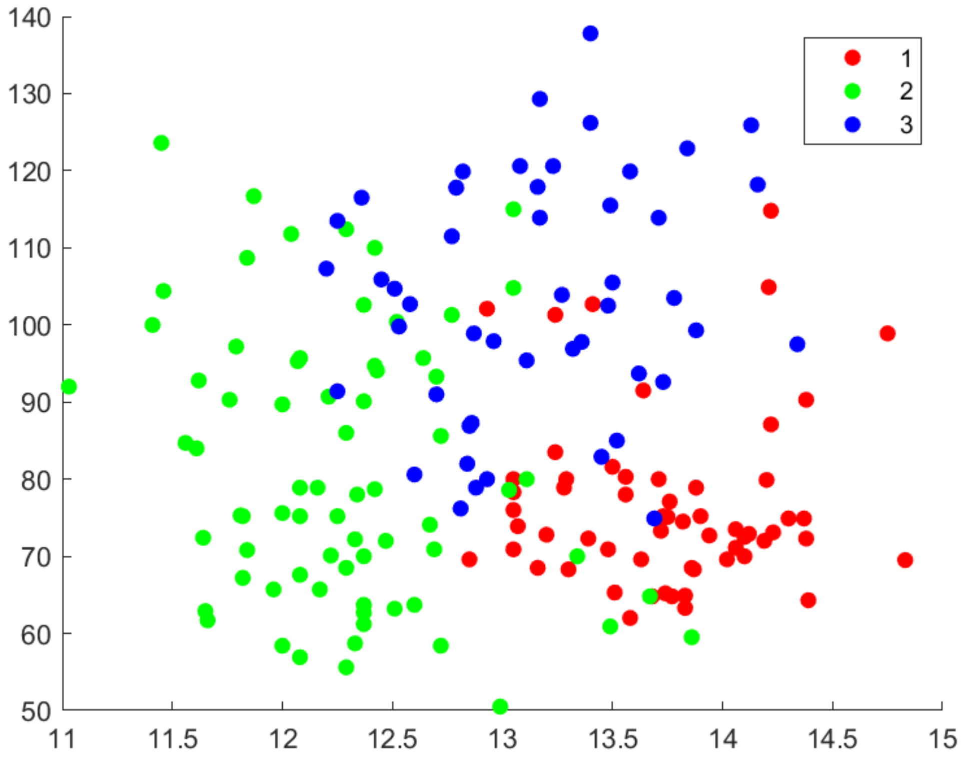
$$= \sum_{n=1}^{N} a_n t_n k(\mathbf{x}_n, \mathbf{x}) + w_0 \begin{cases} \geq 1, & t = +1 \\ \leq -1, & t = -1 \end{cases}$$

Problem

- Assumption linearly separable
- 2 classes only

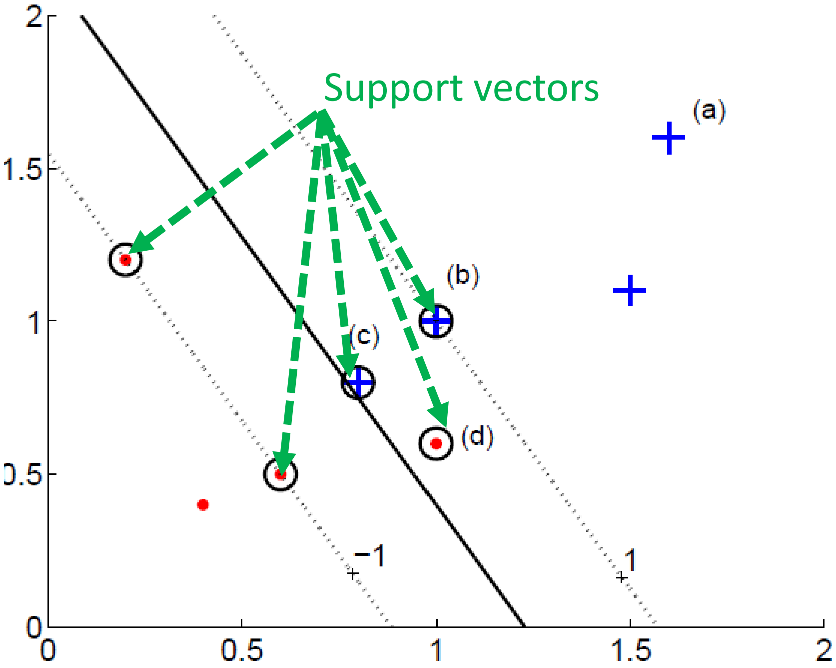
Non-linearly separable Classes

What if data is not linearly separable with zero error?



Non-linearly separable Classes

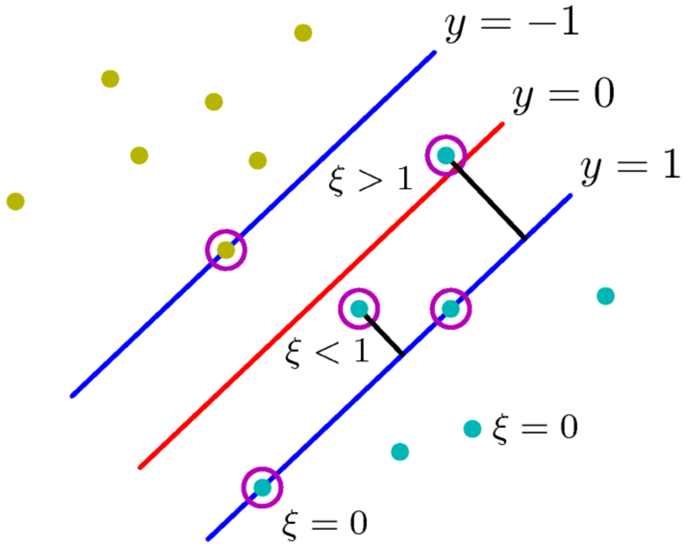
No Separating Hyperplane with zero error?



Alpaydin, “Introduction to Machine Learning”

Non-linearly separable Classes

No Separating Hyperplane with zero error!



$$t_n \in \{-1, +1\}$$

$$t_n \underbrace{\left(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + w_0\right)}_{y(\mathbf{x}_n)} \geq 1$$

$$t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + w_0) \geq 1 - \xi_n$$

$$\xi_n \begin{cases} = 0 & , \ \mathbf{x}_n \text{ correctly classified} \\ \in (0, 1] & , \ \mathbf{x}_n \text{ in margin} \\ > 1 & , \ \mathbf{x}_n \text{ falsely classified} \end{cases}$$

Non-linearly separable Classes

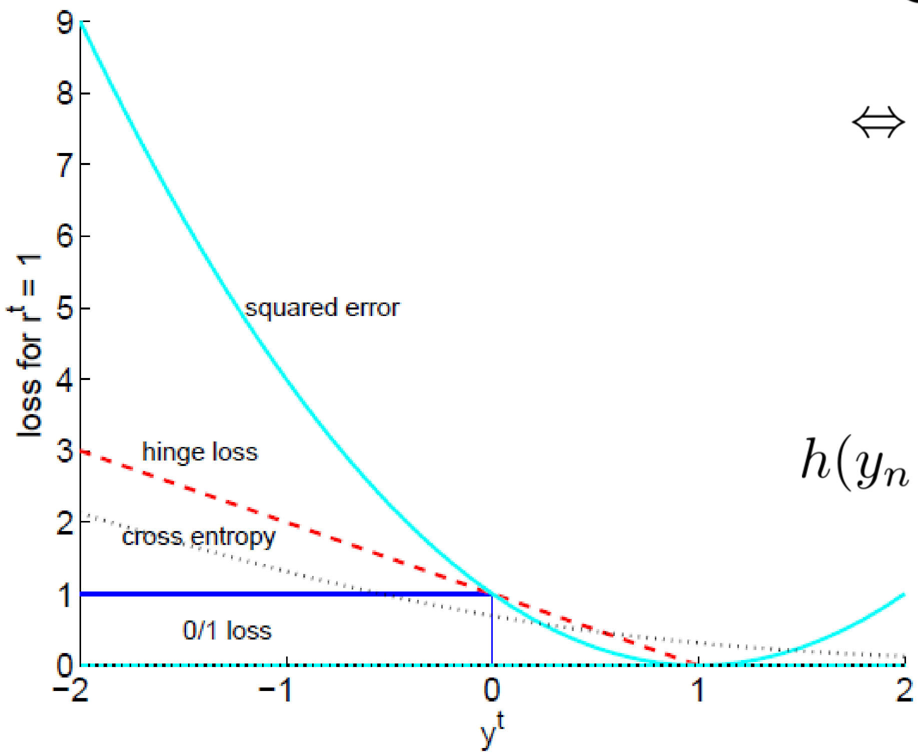
- Linearly separable $t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + w_0) \geq 1$
- Not linearly separable $t_n \underbrace{(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + w_0)}_{y_n} \geq 1 - \xi_n$

$$\Leftrightarrow \quad 1 - t_n y_n \leq \xi_n$$

$t_n y_n > 0$ correctly classified

$t_n y_n < 0$ falsely classified

$$h(y_n, t_n) = \begin{cases} 0, & \text{if } y_n t_n \geq 1 \\ 1 - y_n t_n, & \text{otherwise} \end{cases}$$



Hinge loss is more robust than squared error

Non-linearly separable Classes

- Linearly separable $t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + w_0) \geq 1$
- Not linearly separable $t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + w_0) \geq 1 - \xi_n$

$$\xi_n \begin{cases} = 0 & , \ \mathbf{x}_n \text{ correctly classified} \\ \in (0,1] & , \ \mathbf{x}_n \text{ in margin} \\ > 1 & , \ \mathbf{x}_n \text{ falsely classified} \end{cases}$$

- Soft error

$$\underset{\mathbf{w}, w_0}{\arg\min} \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_n \xi_n$$

$$L(\mathbf{w}, w_0, \boldsymbol{\xi}, \mathbf{a}, \boldsymbol{\mu}) = \frac{1}{2} \|\mathbf{w}\|_2^2 - \sum_{n=1}^{N} a_n t_n \left((\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + w_0) - 1 + \xi_n\right)$$

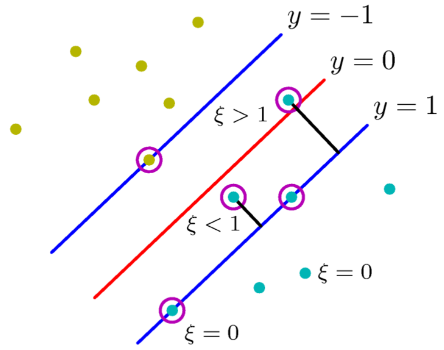
$$+ C \sum_n \xi_n - \sum_{n=1}^{N} \mu_n \xi_n$$

How to train an SVM?

max $\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$

With constraints: $0 \leq a_n \leq C \quad \sum_{n=1}^{N} a_n t_n = 0$

$$\Rightarrow y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}_n, \mathbf{x}) + w_0 \begin{cases} > 0 & , t = +1 \\ \leq 0 & , t = -1 \end{cases}$$

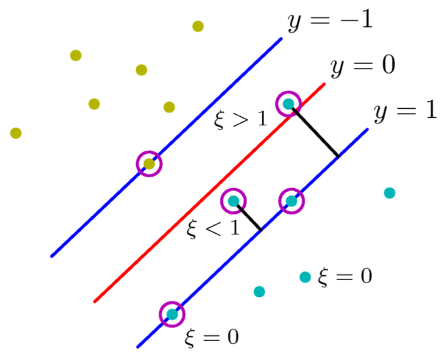


Step 1: Assume support vectors are known: estimate bias

$$w_0 = \frac{1}{M}\sum_{n \in \mathcal{M}} t_n - \sum_{n=1}^{N} a_n t_n k(\mathbf{x}_n, \mathbf{x})$$

How to train an SVM?

max $\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$



With constraints: $0 \leq a_n \leq C \quad \sum_{n=1}^{N} a_n t_n = 0$

$$\Rightarrow y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}_n, \mathbf{x}) + w_0 \begin{cases} > 0 & , \ t = +1 \\ \leq 0 & , \ t = -1 \end{cases}$$

$$w_0 = \frac{1}{M}\sum_{n \in \mathcal{M}} t_n - \sum_{n=1}^{N} a_n t_n k(\mathbf{x}_n, \mathbf{x})$$

Step 2: How to get the support vectors defined by a_n?

- No direct solution (or unfeasible)

Sequential Minimal Optimization (SMO)

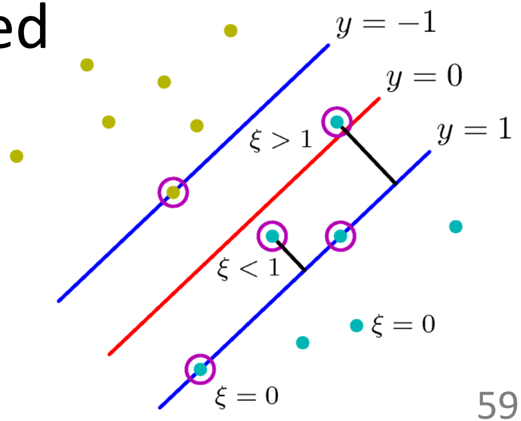
$$\max \tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

With constraints: $0 \leq a_n \leq C \qquad \sum_{n=1}^{N} a_n t_n = 0$

How to estimate the Lagrange Multipliers a_n?

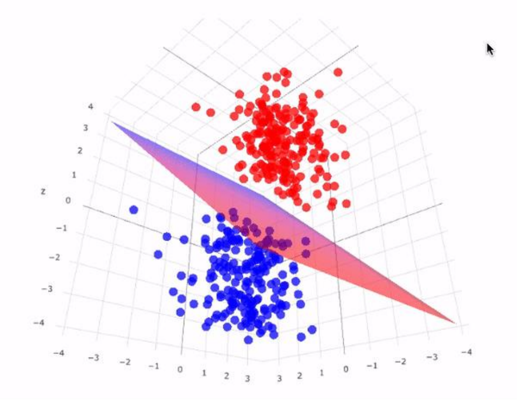
Idea: estimate two: a_n a_m, keep the rest fixed

1) Find a_n that violates conditions
2) Pick a second multiplier a_m
3) optimize the pair
4) Repeat 1) - 3)



Recap: Linear Classifier

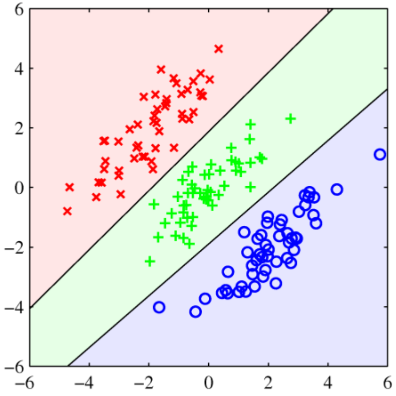
K=2 classes



$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + w_0$$

$$y(\mathbf{x}) \begin{cases} \geq 0 & , \mathbf{x} \in \mathcal{C}_1 \\ < 0 & , \mathbf{x} \in \mathcal{C}_2 \end{cases}$$

K>2 classes

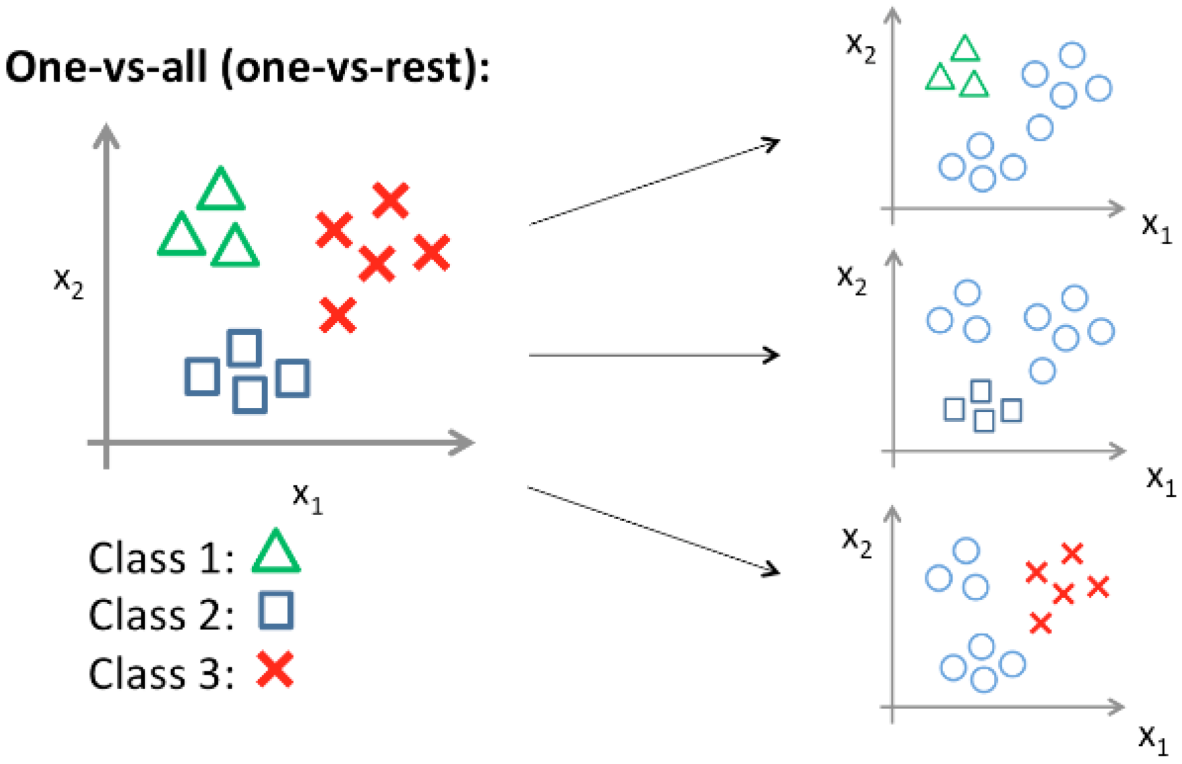


$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\phi(\mathbf{x}) + w_{k0}$$

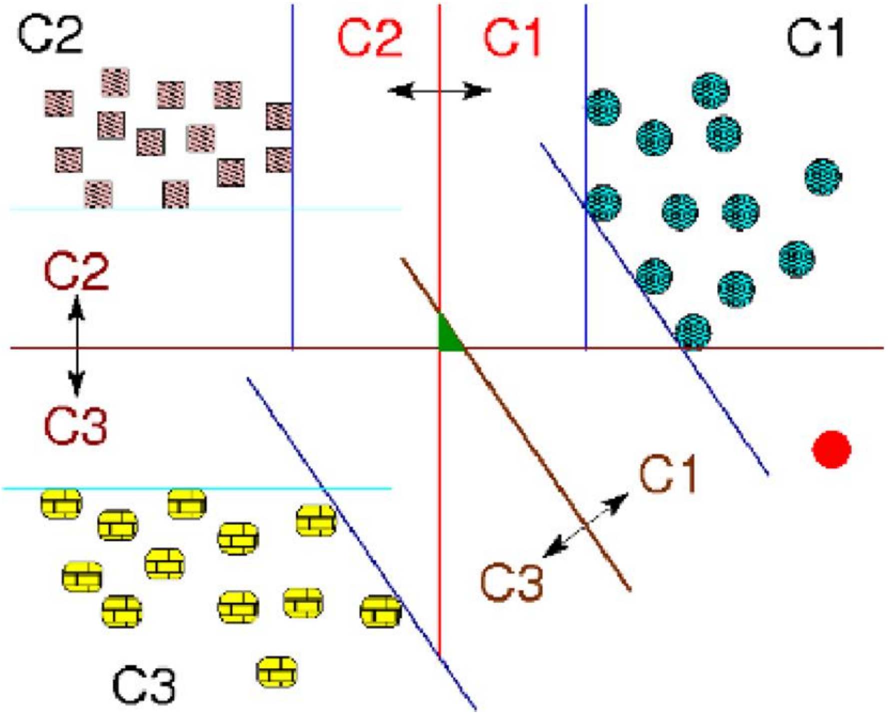
$$m = \arg\max_k \; y_k(\mathbf{x})$$

Multiclass SVM

One-vs-all: K subproblems $y(\mathbf{x}) = \max_k \; y_k(\mathbf{x})$



Multiclass SVM

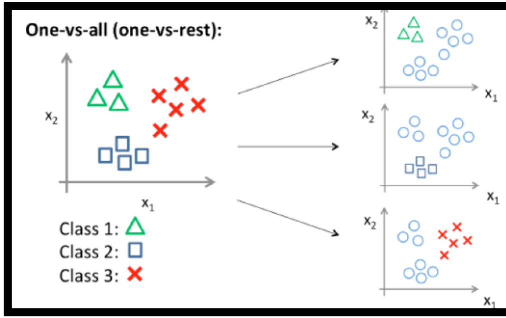


One-vs-one aka
Pairwise separation

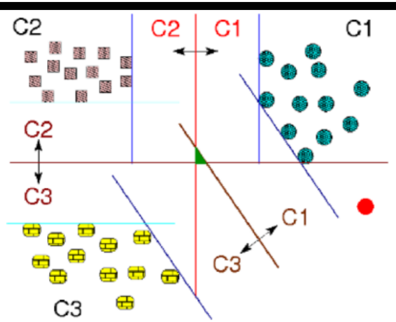
- focus on two classes at a time
- K(K-1)/2

https://www.researchgate.net/figure/Illustration-of-the-SVM-principle-and-of-the-one-versus-one-multiclass-classification_fig2_220098164

Multiclass SVM



1. 1-vs-all: K problems

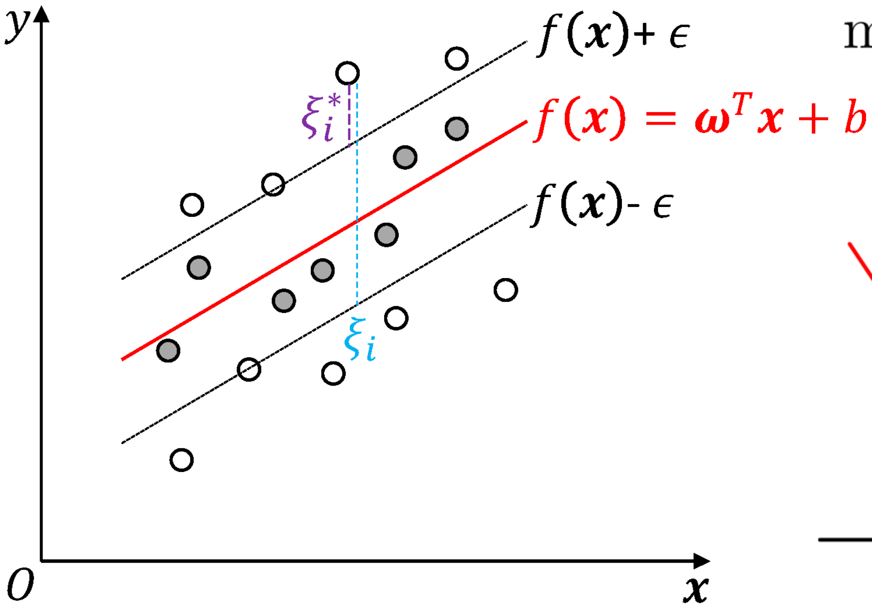


2. Pairwise separation:
   2 classes at a time:
   K(K-1)/2 problems

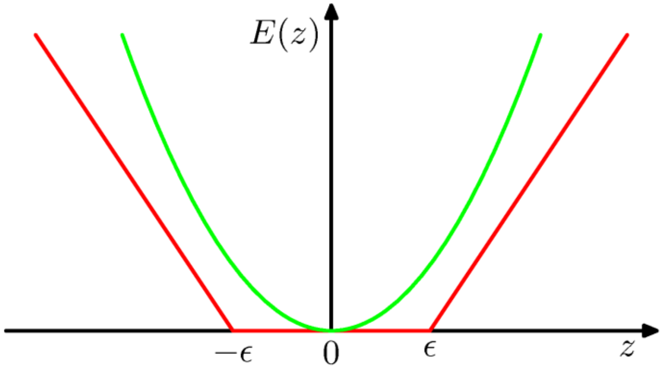
3. Single multiclass optimization
   involving all classes

Support Vector Regression

- Goal: not all points contribute
- allow some error by slack variables (soft margin)



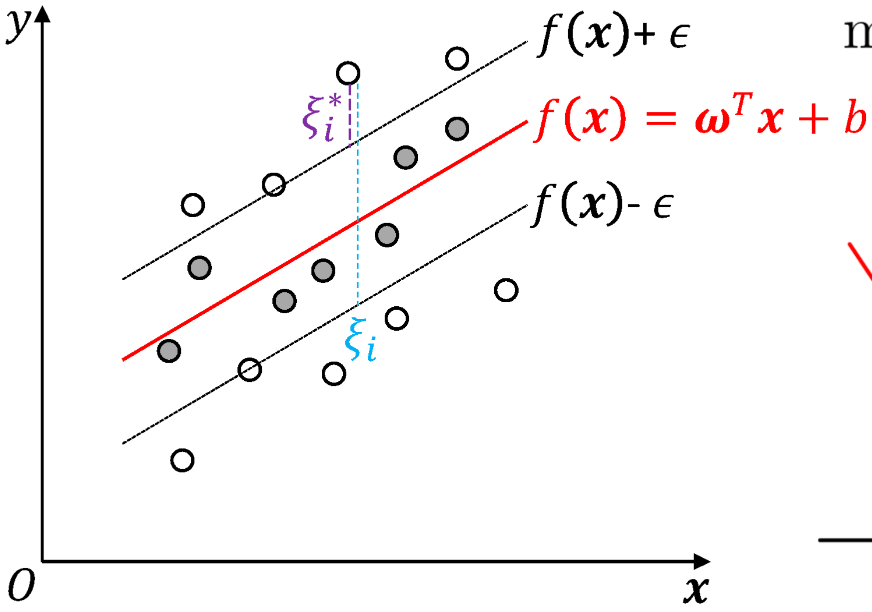
$$\min \frac{1}{2}\sum_{n=1}^{N}(y_n - t_n)^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$



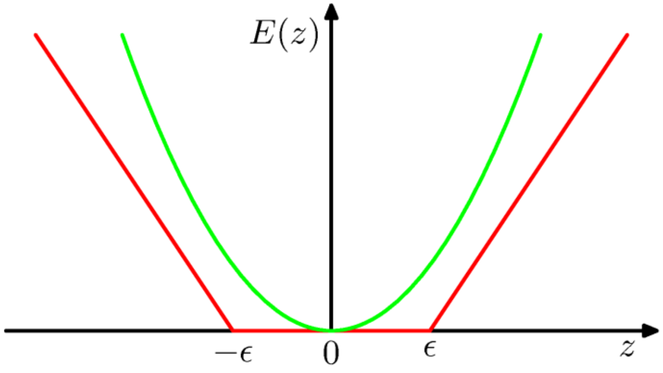
https://www.mdpi.com/2076-3417/10/9/3086/htm

Support Vector Regression

- Goal: not all points contribute
- allow some error by slack variables (soft margin)



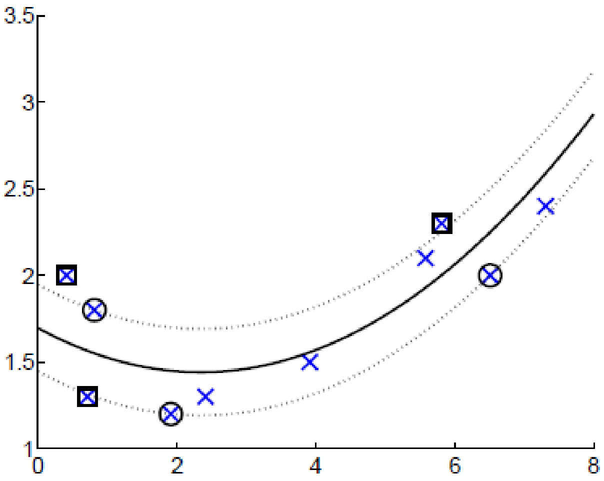
$$\min C \sum_{n=1}^{N} E(y_n - t_n) + \frac{1}{2}\|\mathbf{w}\|_2^2$$



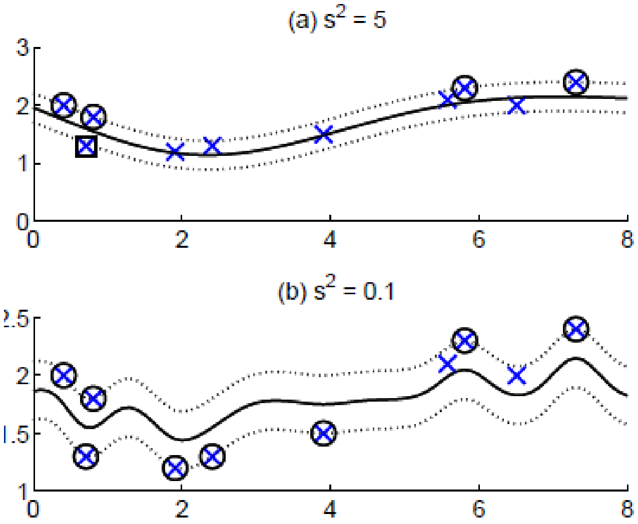
https://www.mdpi.com/2076-3417/10/9/3086/htm

Kernel Regression

Polynomial kernel



Gaussian kernel



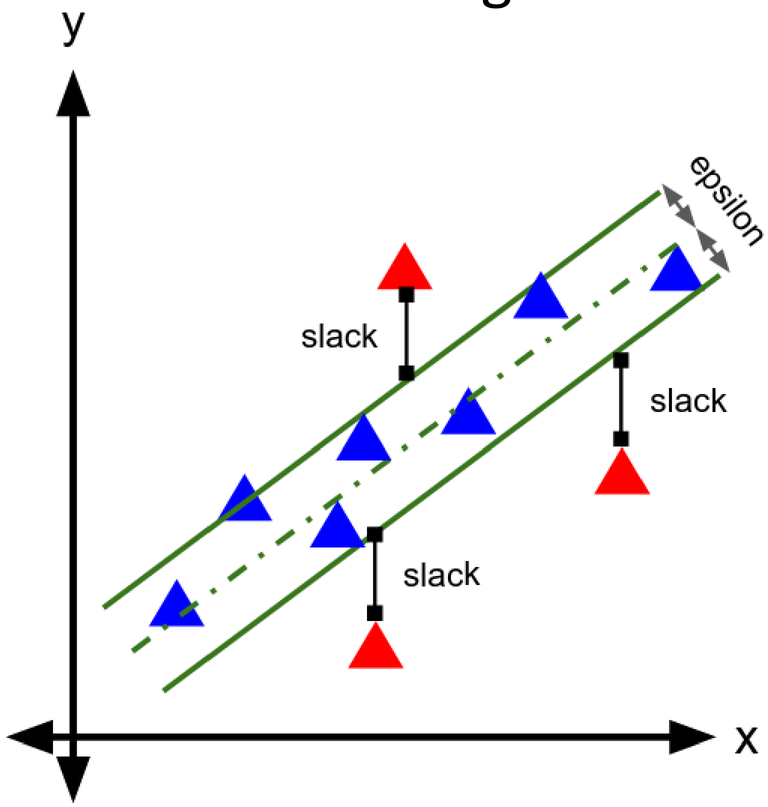
support vectors

○ inside of tube

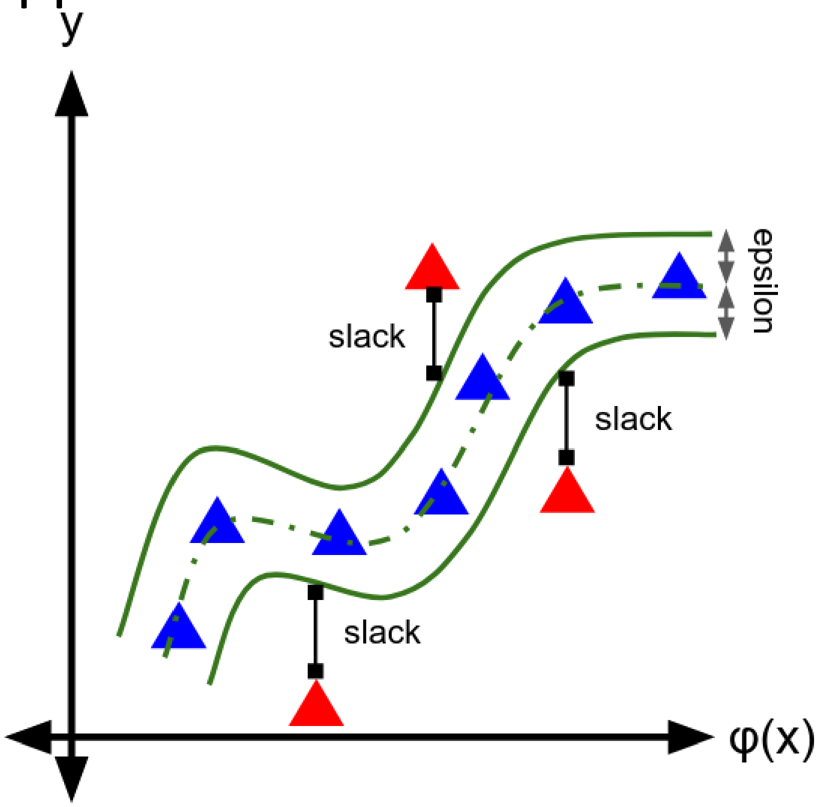
□ outside of tube

Support Vector Regression

Fitted line is weighted sum of support vectors

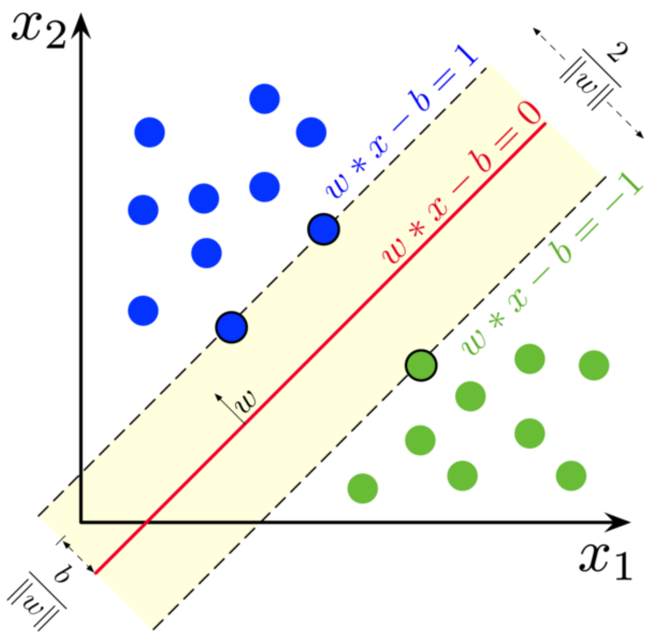


Linear



Non-linear

Summary SVMs



$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + w_0$$

Pros:

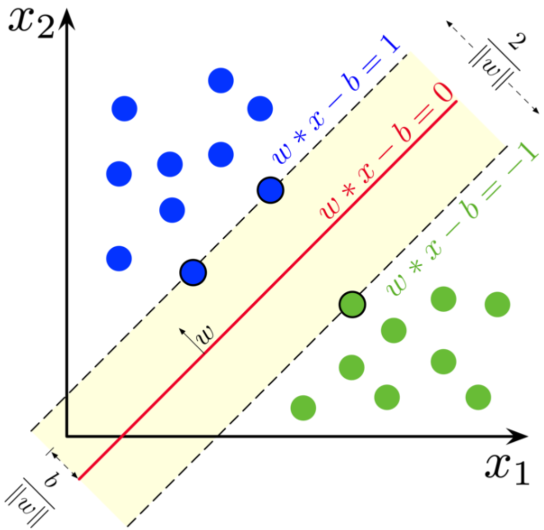
- light-weight model
- less likely to overfit
- Kernel-trick
- works well in higher dimensions

Cons:

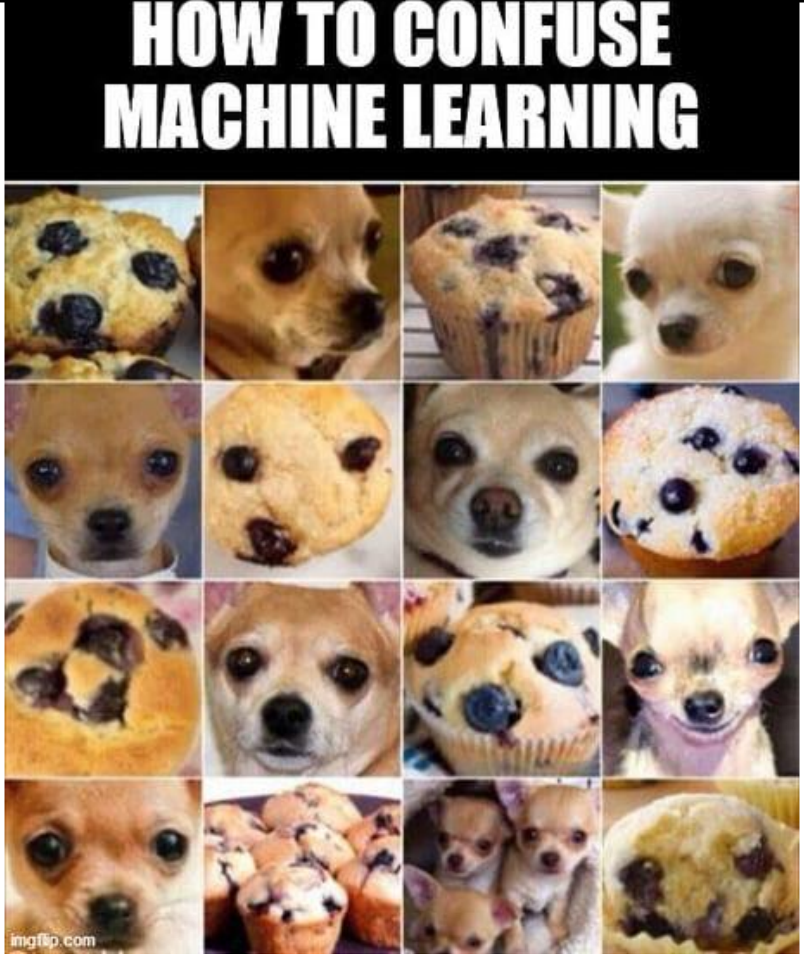
- Not suitable on large datasets
- Kernel is difficult to choose
- Only class decision, i.e. discriminative
  - No class probabilities
  - Not generative

Lessons learnt

- Kernel methods are good if:
  - number of samples is "small"
  - (only) pairwise similarity or distance is known
- Sparse Kernel Machines
  - Only few support vectors define the separating hyperplane or regression line
- Next: words of warning:
  Let the data speak for itself…



“Let the Data speak for itself”



“Let the Data speak for itself” !

Amazon’s Face Recognition Falsely Matched 28 Members of Congress With Mugshots

By Jacob Snow, Technology & Civil Liberties Attorney, ACLU of Northern California
JULY 26, 2018 | 8:00 AM

TAGS: Face Recognition Technology, Surveillance Technologies, Privacy & Technology

Amazon’s face surveillance technology is the target of growing opposition nationwide, and today, there are 28 more causes for concern. In a test the ACLU recently conducted of the facial recognition tool, called “Rekognition,” the software incorrectly matched 28 members of Congress, identifying them as other people who have been arrested for a crime.

The members of Congress who were falsely matched with the mugshot database we used in the test include Republicans and Democrats, men and women, and legislators of all ages, from all across the country.

“The false matches were disproportionately of people of color”

https://www.aclu.org/blog/privacy-technology/surveillance-technologies/amazons-face-recognition-falsely-matched-28

“Let the Data speak for itself”?

CBS NEWS

“An MIT study of three commercial gender-recognition systems found they had errors rates of up to 34% for dark-skinned women — a rate nearly 49 times that for white men.”

Why face-recognition technology has a bias problem

BY IRINA IVANOVA
JUNE 12, 2020 / 7:57 AM / MONEYWATCH

https://www.cbsnews.com/news/facial-recognition-systems-racism-protests-police-bias/

IBM will no longer offer, develop, or research facial recognition technology

IBM’s CEO says we should reevaluate selling the technology to law enforcement

By Jay Peters | @jaypeters | Jun 8, 2020, 8:49pm EDT

https://www.theverge.com/2020/6/8/21284683/ibm-no-longer-general-purpose-facial-recognition-analysis-software

Quality Metrics

- Confusion matrix
- Accuracy $ACC = \frac{TP+TN}{TP+FN+TN+FP}$

		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

Type I error
(false positive)

Type II error
(false negative)

Quality Metrics

- Confusion matrix
- Accuracy $ACC = \frac{TP+TN}{TP+FN+TN+FP}$
- FP-rate $\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{N}} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}$
- Precision $\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$
- Recall $\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$
- F1-Score $\mathrm{F}_1 = 2 \cdot \frac{\mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV}+\mathrm{TPR}} = \frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$
- ROC (Receiver operating characteristic)
- AUC (Area under the curve)
- …