COGS 118A, Spring 2019

Supervised Machine Learning Algorithms

Lecture 13: Kernel

Zhuowen Tu

Midterm II

Midterm II, 02/27/2020 (Thursday)

Time: 12:30-13:50PM

Location: Ledden Auditorium

You can bring one page “cheat sheet”. No use of computers/smart-phones during the exam.

Bring your pen.

Bring your calculator.

A study guide and practice questions have been provided.

Intuition about classification power

$$e^{(f)}_{testing} = e^{(f)}_{training} + e_{gen}(f)$$

- Typically, more powerful a classifier f is, the smaller the training error it can achieve.

$$e^{(f)}_{training} \to 0$$

- However, more powerful a classifier f is, the larger the generalization error it incurs.

$$e_{gen}(f) \to 0.5$$

- The power of a classifier is dependent on the type of classifier (e.g. perceptron, decision tree, nearest neighborhood, etc.) and how many parameters are being learned.

- The power of a classifier doesn't depend on the exact optimal parameters learned after training on a specific task.

Intuition about shattering

- We want to come up a way to characterize the classification power of a given type of classifier that should be agnostic across ALL types of classifiers (disqualifying counting the number of parameters since they have different interpretations for different classifier types).
- Using the concept of shattering allows us to find out the capability of a classifier, given a number of non-overlapping points, by successfully classifying them under all possible labeling configurations.
- If you are checking on n points, then there are 2^n possibilities to verify. Failing on any one of the situations will deem the classifier incapable of shattering n points.
- This is like a bank stress test.

Cross-validation
(works for both regression and classification)

Pros:

Easy to implement.

Works well on both small training data and large training data.

Widely used in data analysis.

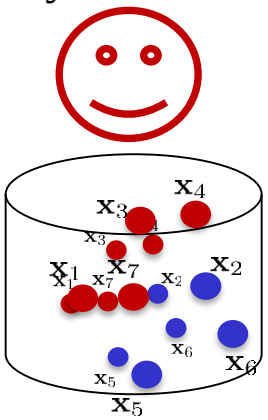
Cons:

It is time-consuming to compute.

Not needed when your data is truly large: keep a hold-out dataset is sufficient.

How would you use cross-validation: example 1

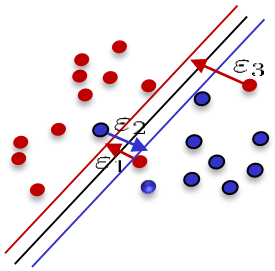
your boss



Labeled Training Data

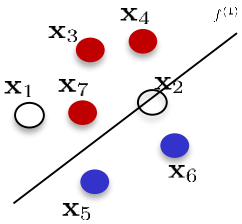
Your task:

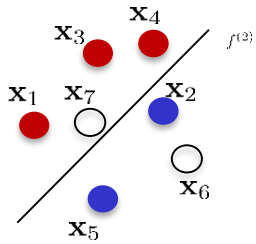
To obtain the "optimal" classifier using the given training data. Find the best hyper-parameter value for C.

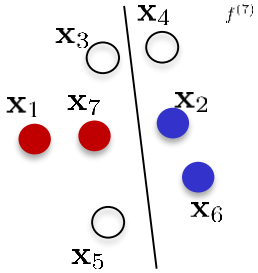


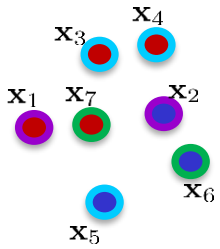
Minimize:

$\mathcal{L}(\mathbf{w}, b) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))_+$









$e(f^{(1)})$ $e(f^{(2)})$ $e(f^{(3)})$

$$\bar{e} = \frac{1}{3}[e(f^{(1)}) + e(f^{(2)}) + e(f^{(3)})]$$

A. $\bar{e}_{C=0} = 0.38$

B. $\bar{e}_{C=0.1} = 0.30$

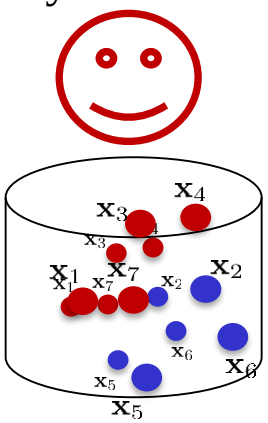
C. $\bar{e}_{C=1.0} = 0.15$

★ D. $\bar{e}_{C=10.0} = 0.10$

E. $\bar{e}_{C=100.0} = 0.25$

How would you use cross-validation: example 2

your boss

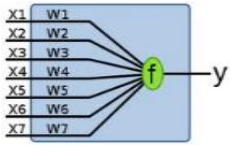


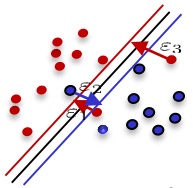
Labeled Training Data

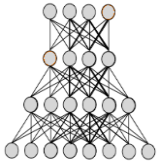
Your task:

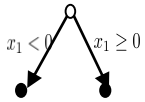
To obtain the "optimal" classifier using the given training data.

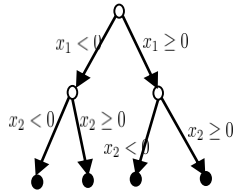
But there are so many design choices for what types of classifiers and configurations (often decided by the hyper-parameters) to use.

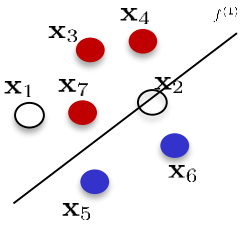


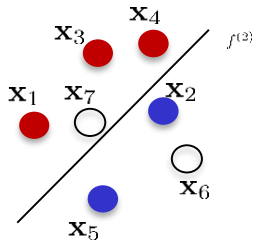


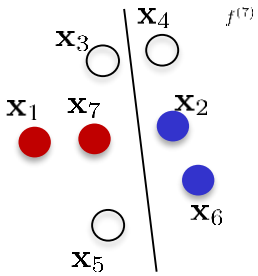


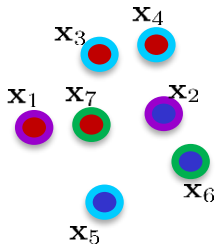






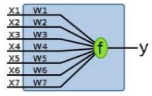




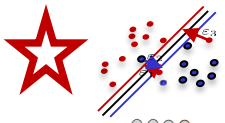


$e(f^{(1)})$ $e(f^{(2)})$ $e(f^{(3)})$

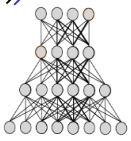
$$\bar{e} = \frac{1}{3}[e(f^{(1)}) + e(f^{(2)}) + e(f^{(3)})]$$



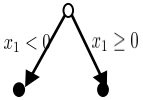
A. $\bar{e}_{perceptron} = 0.39$



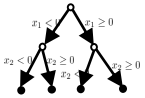
B. $\bar{e}_{SVM} = 0.19$



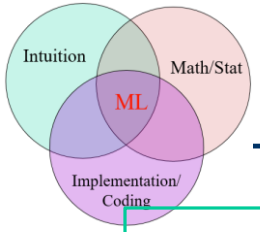
C. $\bar{e}_{NN} = 0.27$



D. $\bar{e}_{Tree1} = 0.38$



E. $\bar{e}_{Tree2} = 0.35$



Recap: Structural risk minimization and cross-validation

1. Given a set of training data: $S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$

$$e_{testing} \leq e_{training} + \sqrt{\frac{h(\log(2n/h+1)) - \log(\eta/4)}{n}}$$

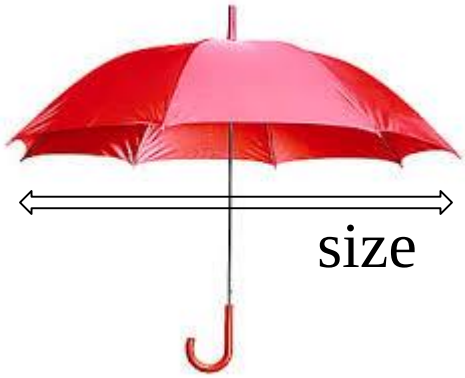
where $e_{testing}$ is unobserved but can be estimated.

2. To achieve the minimal testing error for a given classifier $f(x;\theta)$, we want to: **(a)** attain a small training error $e_{training}$, and **(b)** adopt a large training set of large n **(c)** while making $f(\mathbf{x};\mathbf{w})$ as simple as possible (characterized by the power/VC dimension of $f(\mathbf{x};\mathbf{w}) - - h$).

3. The optimal choise for $f(\mathbf{x};\mathbf{w})$ can be guided by the structural risk minimization principle (in theory). Typically, there will additional hyper-parameters γ.

4. In practice, we use e.g. cross-validation to do hyper-parameter tuning on $\gamma$to choose $f(\mathbf{x};\mathbf{w})$. γ can be e.g. the parameter C in SVM, the choice between L2 vs. L1, the type of classifier, etc.

Understanding the Kernel

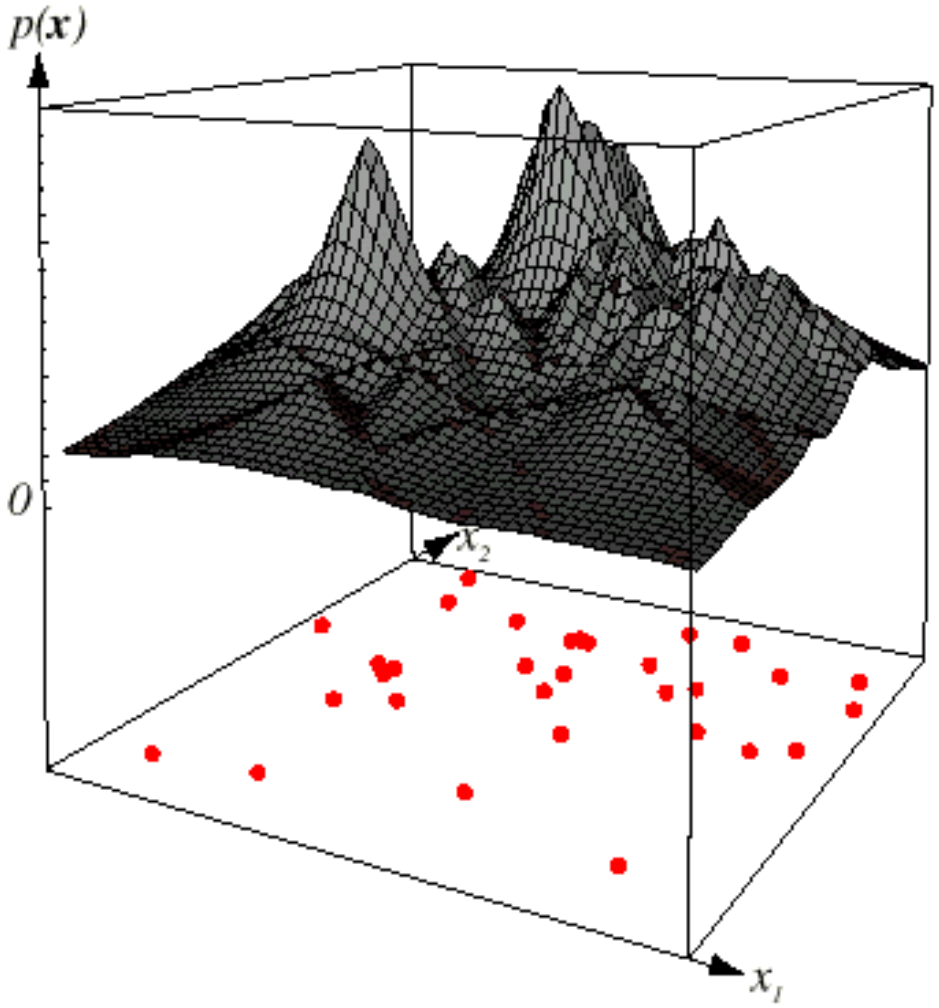


Kernel function $f(x)$.

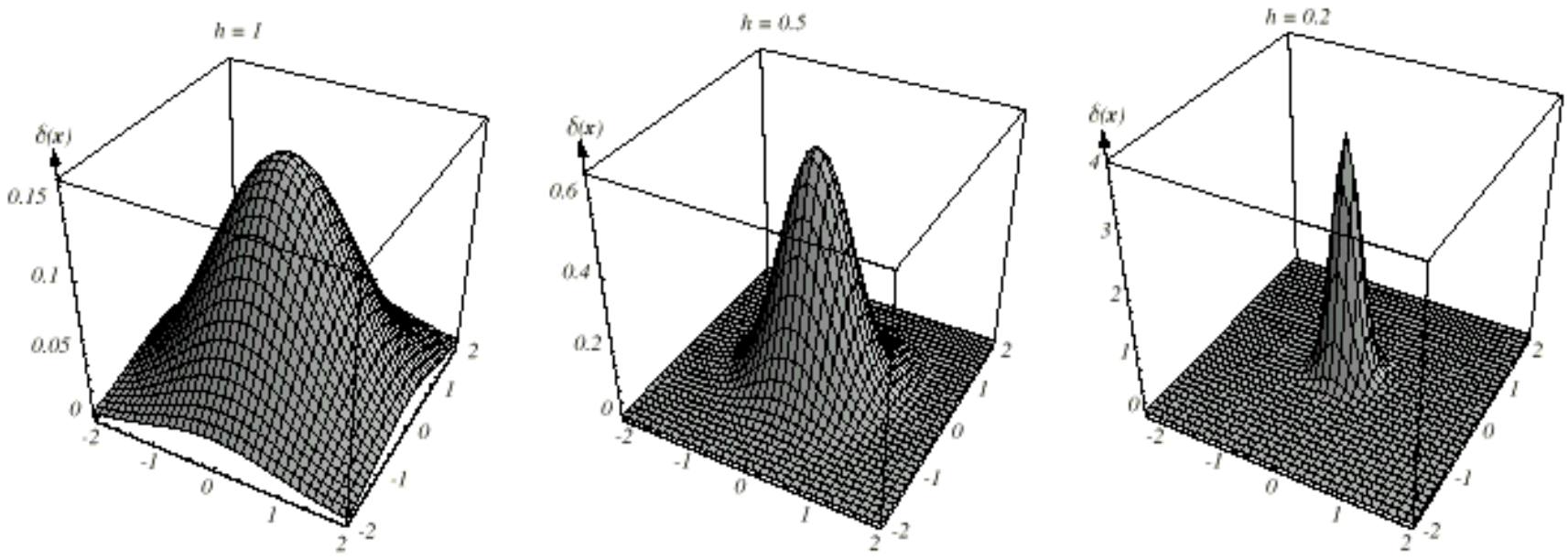
Adding up all the K points attached with a kernel for each point:

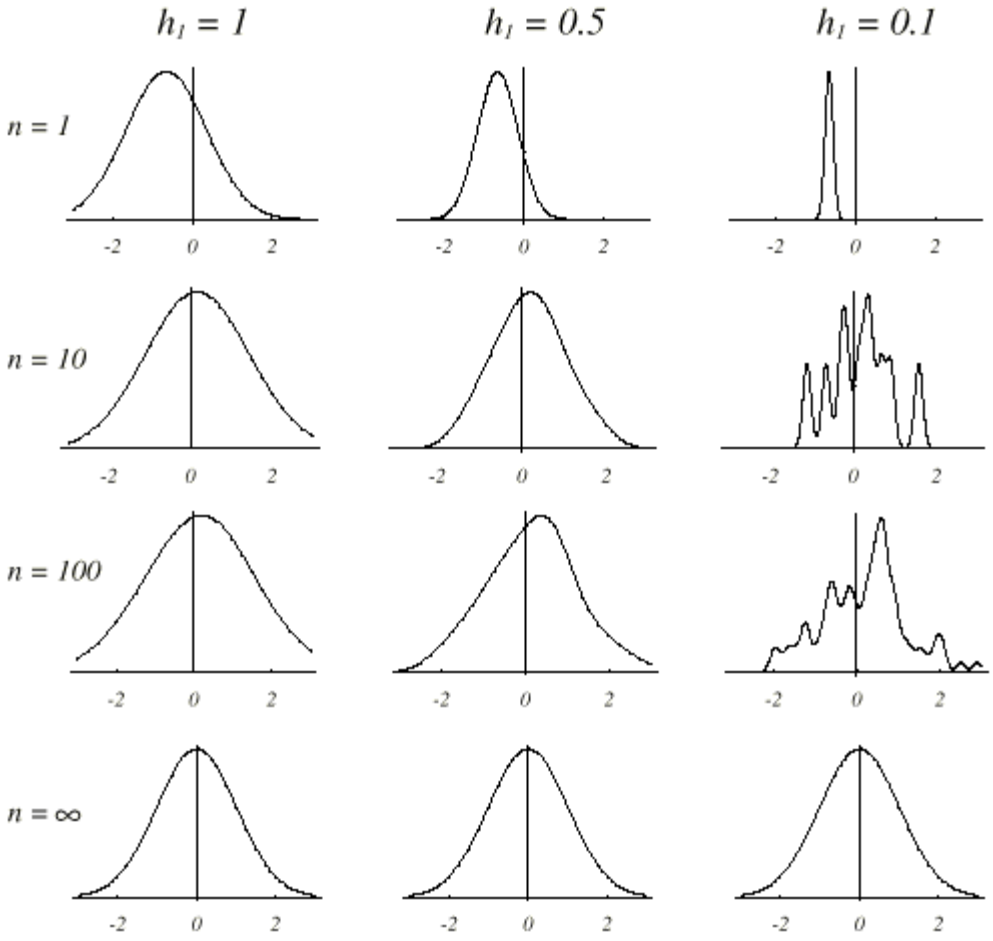
$$\sum_{k=1}^{K} f_k(x)$$

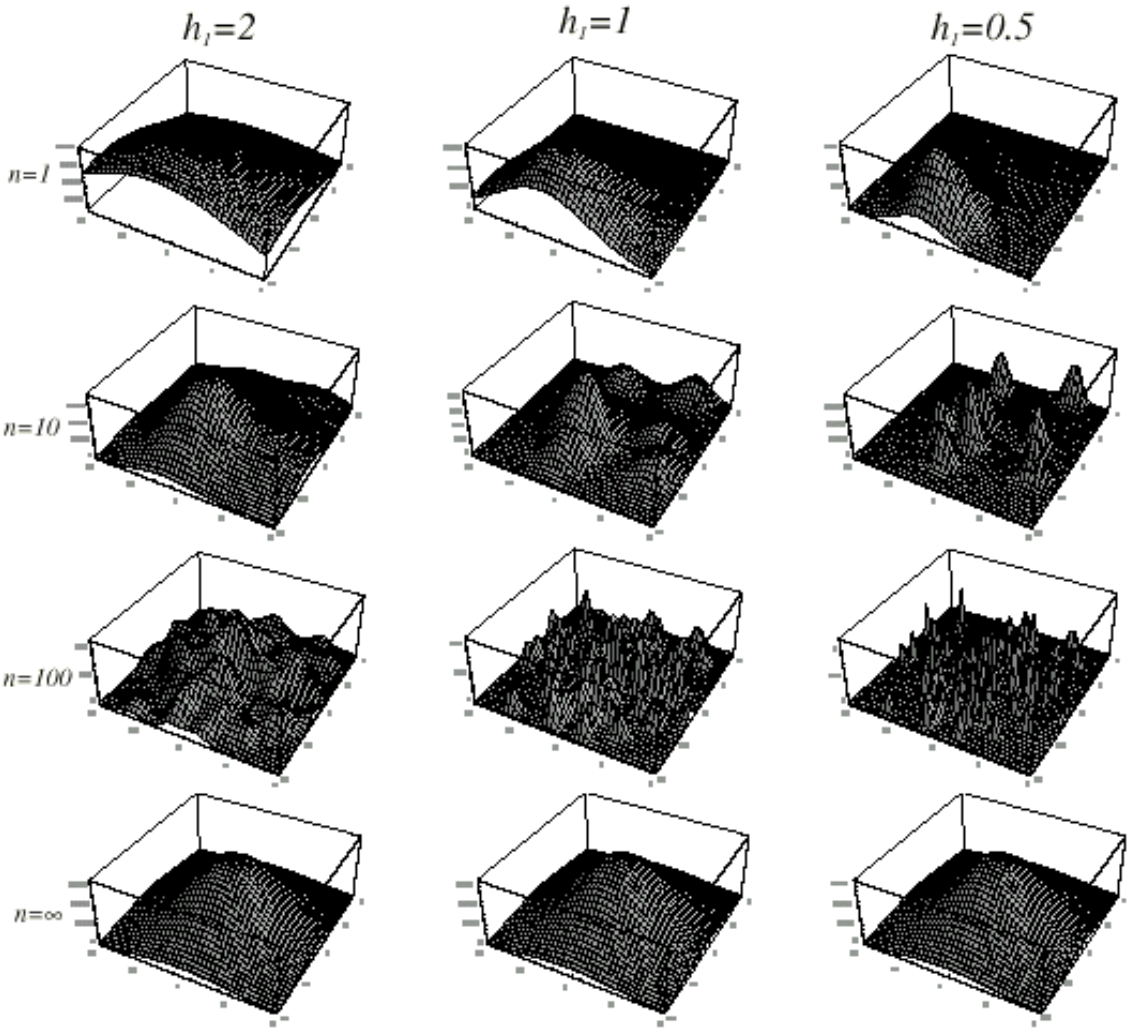
K_n-Nearest Neighbor Estimation Examples – cont.



Choice of kernel sizes: $\varphi(\frac{\mathbf{x}-\mathbf{x}_i}{h_n})$




$h_1 = 1$
$h_1 = 0.5$
$h_1 = 0.1$
$n = 1$
$n = 10$
$n = 100$
$n = \infty$
-2
0
2


h_1=2
h_1=1
h_1=0.5
n=1
n=10
n=100
n=∞

How to compute?

To remove the potential confusion, lest use l instead.

$$p_l(\mathbf{x}) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})}$$

$\mathbf{x}$: a test/query data sample

l: a hyper-parameter

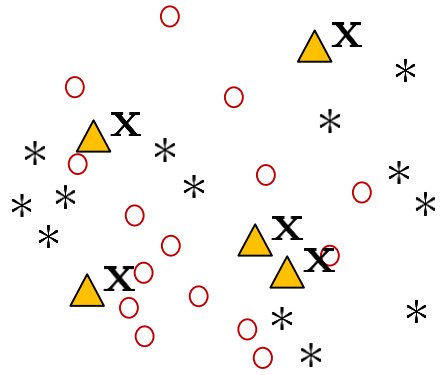
$k_l(\mathbf{x})$: number of samples within the Region.

$V_l(\mathbf{x})$: the volume of the Region.

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

○ $y = +1$

∗ $y = -1$



How to compute?

Strategy I

$$p_l(\mathbf{x}) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})}$$

$\mathbf{x}$: a test/query data sample

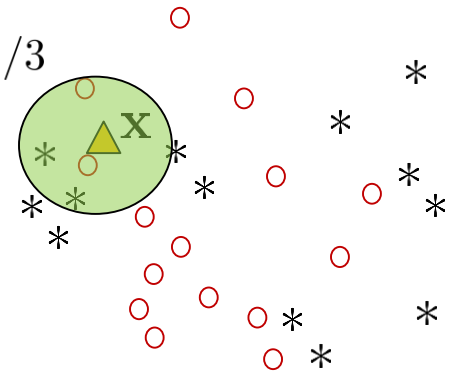
l: a hyper-parameter

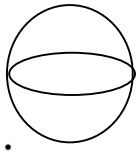
$k_l(\mathbf{x})$: number of samples within the Region.

$V_l(\mathbf{x})$: the volume of the Region.

$\circ$ $y = +1$ $\quad *$ $y = -1$

$V_9(\mathbf{x}) = 1/3$





Strategy 1: $V_l(\mathbf{x}) = 1/\sqrt{l}$: fixed region

Say: $l = 9 \rightarrow$: fixed region/ball size of $1/3$.

$$p_l(\mathbf{x}|y=+1) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})} \propto \frac{2}{9 \times (1/3)} \propto \frac{2}{3}$$

$$p_l(\mathbf{x}|y=-1) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})} \propto \frac{2}{9 \times (1/3)} \propto \frac{2}{3}$$

$$p(y=+1|\mathbf{x}) = \frac{p_l(\mathbf{x}|y=+1)}{p_l(\mathbf{x}|y=-1) + p_l(\mathbf{x}|y=+1)} = 0.5$$

How to compute?

Strategy I

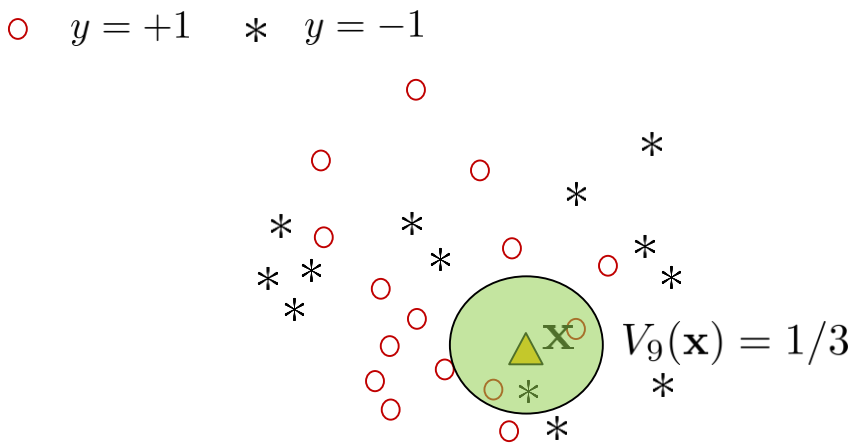
$$p_l(\mathbf{x}) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})}$$

$\mathbf{x}$: a test/query data sample

l: a hyper-parameter

$k_l(\mathbf{x})$: number of samples within the Region.

$V_l(\mathbf{x})$: the volume of the Region.



Strategy 1: $V_l(\mathbf{x}) = 1/\sqrt{l}$: fixed region

Say: $l = 9 \rightarrow$: fixed region/ball size of $1/3$.

$$p_l(\mathbf{x}|y=+1) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})} \propto \frac{2}{9 \times (1/3)} \propto \frac{2}{3}$$

$$p_l(\mathbf{x}|y=-1) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})} \propto \frac{1}{9 \times (1/3)} \propto \frac{1}{3}$$

$$p(y=+1|\mathbf{x}) = \frac{p_l(\mathbf{x}|y=+1)}{p_l(\mathbf{x}|y=-1) + p_l(\mathbf{x}|y=+1)} = 0.66$$

How to compute?

Strategy I

$$p_l(\mathbf{x}) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})}$$

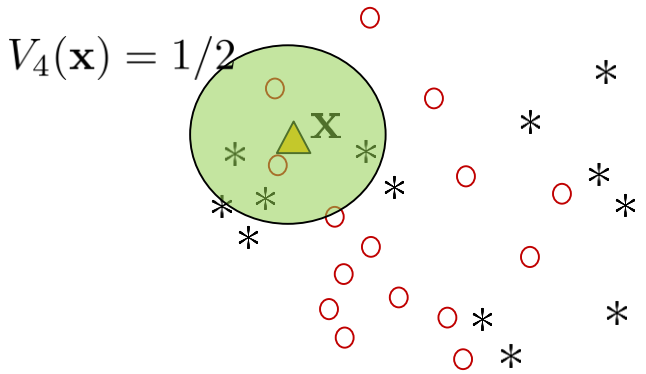
$\mathbf{x}$: a test/query data sample

l: a hyper-parameter

$k_l(\mathbf{x})$: number of samples within the Region.

$V_l(\mathbf{x})$: the volume of the Region.

$\circ$ $y = +1$ $\quad *$ $y = -1$



Strategy 1: $V_l(\mathbf{x}) = 1/\sqrt{l}$: fixed region

Say: $l = 4 \rightarrow$: fixed region/ball size of $1/2$.

$$p_l(\mathbf{x}|y=+1) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})} \propto \frac{2}{4 \times (1/2)} \propto 1$$

$$p_l(\mathbf{x}|y=-1) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})} = \frac{3}{4 \times (1/2)} \propto \frac{3}{2}$$

$$p(y=+1|\mathbf{x}) = \frac{p_l(\mathbf{x}|y=+1)}{p_l(\mathbf{x}|y=-1) + p_l(\mathbf{x}|y=+1)} = 0.4$$

How to compute?

Strategy I

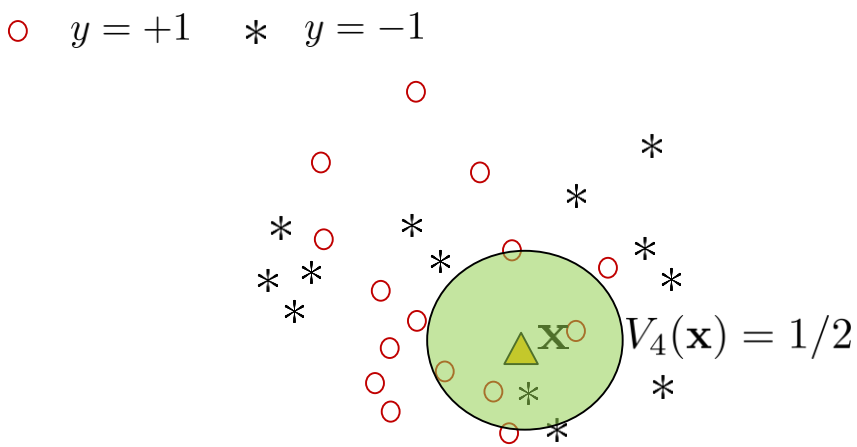
$$p_l(\mathbf{x}) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})}$$

$\mathbf{x}$: a test/query data sample

l: a hyper-parameter

$k_l(\mathbf{x})$: number of samples within the Region.

$V_l(\mathbf{x})$: the volume of the Region.



Strategy 1: $V_l(\mathbf{x}) = 1/\sqrt{l}$: fixed region

Say: $l = 4 \rightarrow$: fixed region/ball size of $1/2$.

$$p_l(\mathbf{x}|y=+1) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})} \propto \frac{3}{4 \times (1/2)} \propto \frac{3}{2}$$

$$p_l(\mathbf{x}|y=-1) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})} = \frac{1}{4 \times (1/2)} \propto \frac{1}{2}$$

$$p(y=+1|\mathbf{x}) = \frac{p_l(\mathbf{x}|y=+1)}{p_l(\mathbf{x}|y=-1) + p_l(\mathbf{x}|y=+1)} = 0.75$$

How to compute?

Strategy II

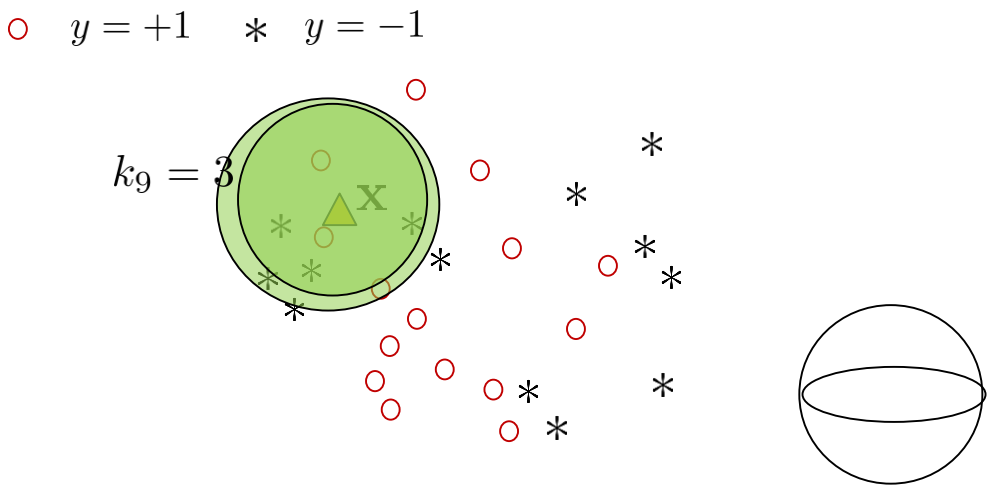
$$p_l(\mathbf{x}) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})}$$

$\mathbf{x}$: a test/query data sample

l: a hyper-parameter

$k_l(\mathbf{x})$: number of samples within the Region.

$V_l(\mathbf{x})$: the volume of the Region.



Strategy 1I: $k_l = \sqrt{l}$: grow region

Say: $l = 9 \rightarrow$: grow the ball to include $\sqrt{9} = 3$ points.

$$p_l(\mathbf{x}|y=+1) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})} \propto \frac{3}{9 \times 0.6} \propto \frac{3}{5.4}$$

$$p_l(\mathbf{x}|y=-1) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})} \propto \frac{3}{9 \times 0.55} \propto \frac{3}{4.95}$$

$$p(y=+1|\mathbf{x}) = \frac{p_l(\mathbf{x}|y=+1)}{p_l(\mathbf{x}|y=-1)+p_l(\mathbf{x}|y=+1)} = 0.48$$

How to compute?

Strategy II

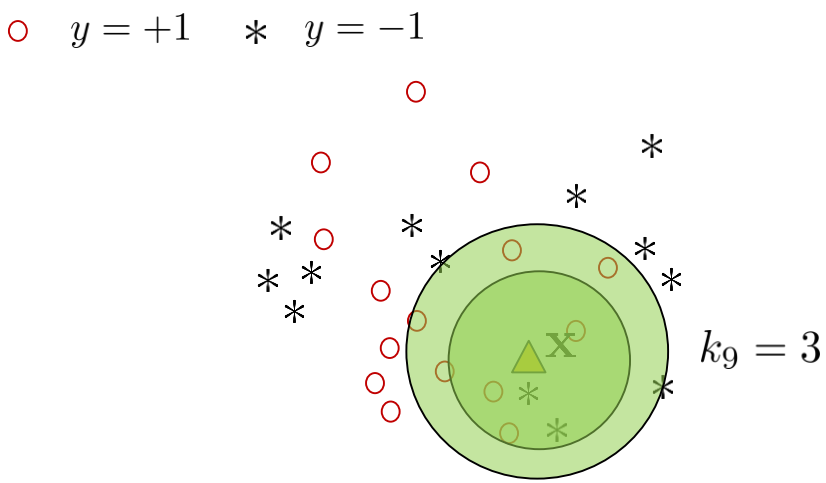
$$p_l(\mathbf{x}) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})}$$

$\mathbf{x}$: a test/query data sample

l: a hyper-parameter

$k_l(\mathbf{x})$: number of samples within the Region.

$V_l(\mathbf{x})$: the volume of the Region.



Strategy 1I: $k_l = \sqrt{l}$: grow region

Say: $l = 9 \rightarrow$: grow the ball to include $\sqrt{9} = 3$ points.

$$p_l(\mathbf{x}|y=+1) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})} \propto \frac{3}{9 \times 0.5} \propto \frac{3}{4.5}$$

$$p_l(\mathbf{x}|y=-1) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})} \propto \frac{3}{9 \times 0.8} \propto \frac{3}{7.2}$$

$$p(y=+1|\mathbf{x}) = \frac{p_l(\mathbf{x}|y=+1)}{p_l(\mathbf{x}|y=-1)+p_l(\mathbf{x}|y=+1)} = 0.62$$

How to compute?

Strategy II

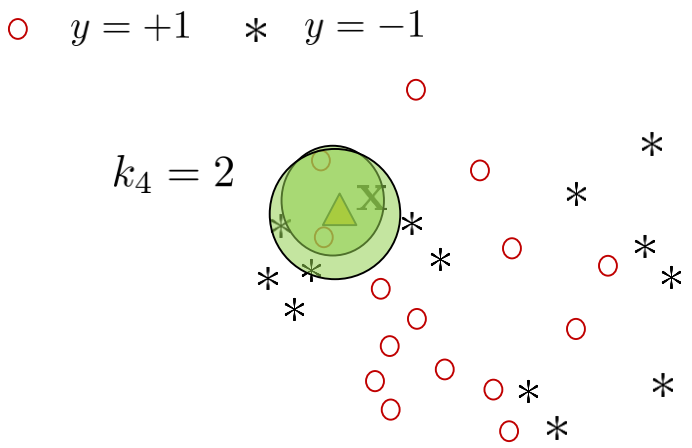
$$p_l(\mathbf{x}) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})}$$

$\mathbf{x}$: a test/query data sample

l: a hyper-parameter

$k_l(\mathbf{x})$: number of samples within the Region.

$V_l(\mathbf{x})$: the volume of the Region.



Strategy 1I: $k_l = \sqrt{l}$: grow region

Say: $l = 4 \rightarrow$: grow the ball to include $\sqrt{4} = 2$ points.

$$p_l(\mathbf{x}|y=+1) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})} \propto \frac{2}{4 \times 0.3} \propto \frac{2}{1.2}$$

$$p_l(\mathbf{x}|y=-1) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})} \propto \frac{2}{4 \times 0.35} \propto \frac{2}{1.4}$$

$$p(y=+1|\mathbf{x}) = \frac{p_l(\mathbf{x}|y=+1)}{p_l(\mathbf{x}|y=-1)+p_l(\mathbf{x}|y=+1)} = 0.53$$

How to compute?

Strategy II

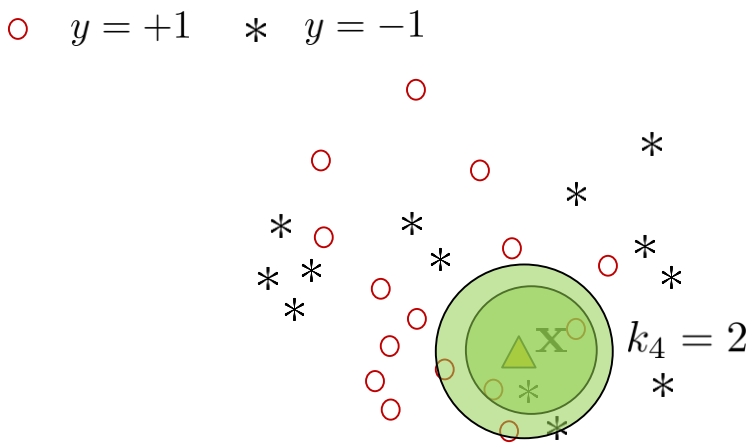
$$p_l(\mathbf{x}) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})}$$

$\mathbf{x}$: a test/query data sample

l: a hyper-parameter

$k_l(\mathbf{x})$: number of samples within the Region.

$V_l(\mathbf{x})$: the volume of the Region.



Strategy 1I: $k_l = \sqrt{l}$: grow region

Say: $l = 4 \rightarrow$: grow the ball to include $\sqrt{4} = 2$ points.

$$p_l(\mathbf{x}|y=+1) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})} \propto \frac{2}{4 \times 0.3} \propto \frac{2}{1.2}$$

$$p_l(\mathbf{x}|y=-1) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})} \propto \frac{2}{4 \times 0.4} \propto \frac{2}{1.6}$$

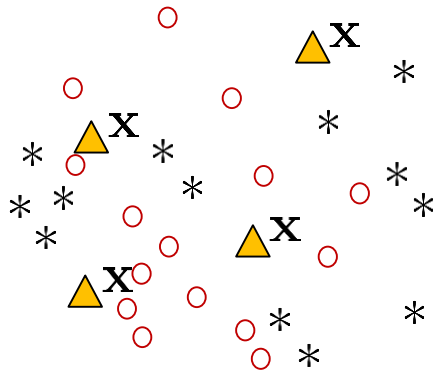
$$p(y=+1|\mathbf{x}) = \frac{p_l(\mathbf{x}|y=+1)}{p_l(\mathbf{x}|y=-1)+p_l(\mathbf{x}|y=+1)} = 0.57$$

The k-Nearest Neighbor Rule

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

An extension of the nearest neighbor rule:
The k-nearest neighbor rule classifies $\mathbf{x}$ by assigning it the label most frequently represented among the k nearest samples In other words, given $\mathbf{x}$, we find the k nearest labeled samples. The label appeared most is assigned to $\mathbf{x}$.

○ $y = +1$
∗ $y = -1$

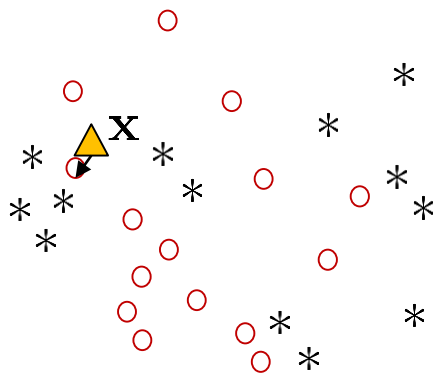


The k-Nearest Neighbor Rule

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

An extension of the nearest neighbor rule:
The k-nearest neighbor rule classifies $\mathbf{x}$ by assigning it the label most frequently represented among the k nearest samples In other words, given $\mathbf{x}$, we find the k nearest labeled samples. The label appeared most is assigned to $\mathbf{x}$.

○ $y = +1$
∗ $y = -1$



$k = 1$

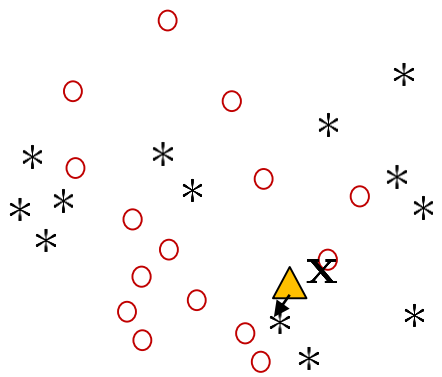
$y = +1 \rightarrow \mathbf{x}$

The k-Nearest Neighbor Rule

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

An extension of the nearest neighbor rule:
The k-nearest neighbor rule classifies $\mathbf{x}$ by assigning it the label most frequently represented among the k nearest samples In other words, given $\mathbf{x}$, we find the k nearest labeled samples. The label appeared most is assigned to $\mathbf{x}$.

○ $y = +1$
∗ $y = -1$



$k = 1$

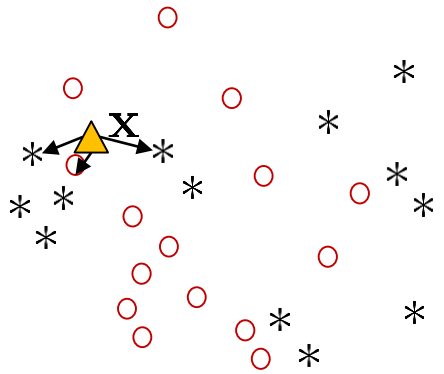
$y = -1 \rightarrow \mathbf{x}$

The k-Nearest Neighbor Rule

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

An extension of the nearest neighbor rule:
The k-nearest neighbor rule classifies $\mathbf{x}$ by assigning it the label most frequently represented among the k nearest samples In other words, given $\mathbf{x}$, we find the k nearest labeled samples. The label appeared most is assigned to $\mathbf{x}$.

○ $y = +1$
∗ $y = -1$



$k = 3$

1 positives
2 negative

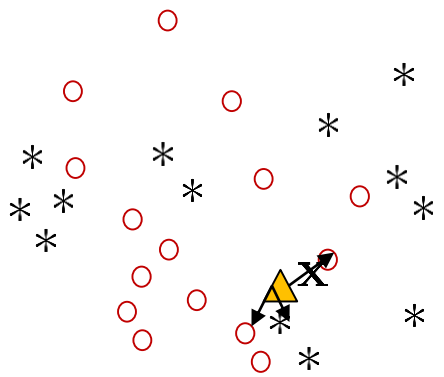
$y = -1 \rightarrow \mathbf{x}$

The k-Nearest Neighbor Rule

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

An extension of the nearest neighbor rule:
The k-nearest neighbor rule classifies $\mathbf{x}$ by assigning it the label most frequently represented among the k nearest samples In other words, given $\mathbf{x}$, we find the k nearest labeled samples. The label appeared most is assigned to $\mathbf{x}$.

○ $y = +1$
∗ $y = -1$



$k = 3$

2 positives
1 negative

$y = +1 \rightarrow \mathbf{x}$

K-Nearest Neighborhood Classifier

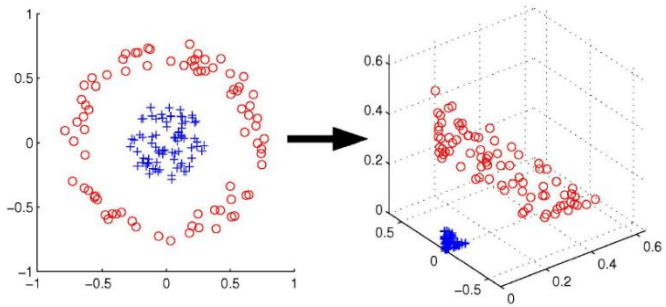
1. Very easy to implement.
2. Works very well in practice.
3. Non-parametric model.
4. Model complexity is too high when the training set is large.
5. Computational complexity is high.

Understanding the kernel

(we only require you to understand the basic concepts here)

Main motivations for applying kernels in machine learning

1. Turn non-separable/difficulty classification problems into separable/easy ones by projecting the original feature space into non-linear (typically higher) dimensions.



original space (non-separable)

kernel space (separable)

2. Turn a parametric (explicit) representation into a non-parametric (implicit) form.

$$sign(\mathbf{w}^T\mathbf{x} + b) \longrightarrow sign(\textstyle\sum_i \alpha_i \times K(\mathbf{x}_i, \mathbf{x}))$$

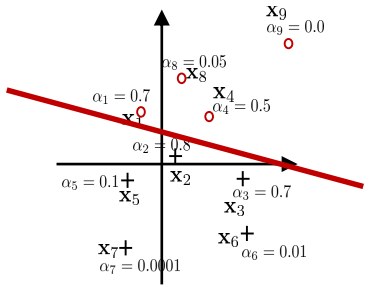
Main motivations for applying kernels in machine learning

Turn a parametric (explicit) representation into a non-parametric (implicit) form.

$$sign(\mathbf{w}^T\mathbf{x} + b) \longrightarrow sign(\sum_i \alpha_i \times K(\mathbf{x}_i, \mathbf{x}))$$

$K(\mathbf{x}_i, \mathbf{x})$ measures the "similarity" between $\mathbf{x}_i$ and $\mathbf{x}$ in the kernel space.

$\alpha_i \in \mathbb{R}$ refers to the learned "weight" for each input sample $\mathbf{x}_i$. Samples with large magnitude of α_i are referred to as the Support Vectors in the SVM classifier.



Main motivations for applying kernels in machine learning

Turn a parametric (explicit) representation into a non-parametric (implicit) form.

$$sign(\mathbf{w}^T\mathbf{x}+b) \longrightarrow sign(\sum_i \alpha_i \times K(\mathbf{x}_i,\mathbf{x}))$$

$K(\mathbf{x}_i,\mathbf{x})$ measures the "similarity" between $\mathbf{x}_i$ and $\mathbf{x}$ in the kernel space.

There are two strategies to compute $K(\mathbf{x}_i,\mathbf{x})$).

1. If we know the projection function: $\phi(\mathbf{x})$.

$$\begin{aligned} K(\mathbf{x}_i,\mathbf{x}) &= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}) \\ &\equiv \phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}) \\ &\equiv <\phi(\mathbf{x}_i),\phi(\mathbf{x})> \end{aligned}$$

2. If we do not know the projection function, then e.g.

$$K(\mathbf{x}_i,\mathbf{x}) = e^{-||\mathbf{x}_i-\mathbf{x}||^2}$$

It's an implicit function to compute the similarity between $\mathbf{x}_i$ and $\mathbf{x}$, without knowing the projection function $\phi(\mathbf{x})$ explicitly.

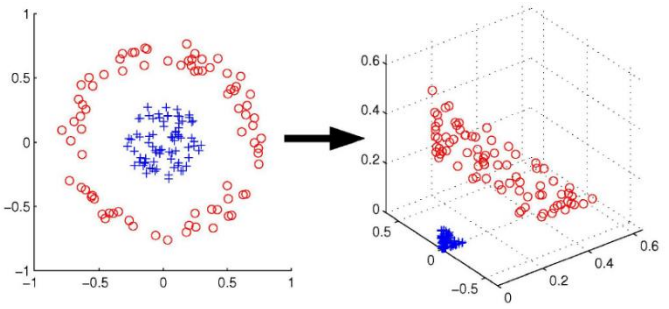
Defining kernels

$$sign(\sum_i \alpha_i \times K(\mathbf{x}_i, \mathbf{x}))$$

There are two strategies to compute $K(\mathbf{x}_i, \mathbf{x})$)

1. If we know the projection function: $\phi(\mathbf{x})$.

$$K(\mathbf{x}_i, \mathbf{x}) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) \equiv \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) \equiv < \phi(\mathbf{x}_i), \phi(\mathbf{x}) >$$



original space kernel space

Example:

$\mathbf{x} = (x_1, x_2)$

non-separable

$\phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1 \times x_2)$

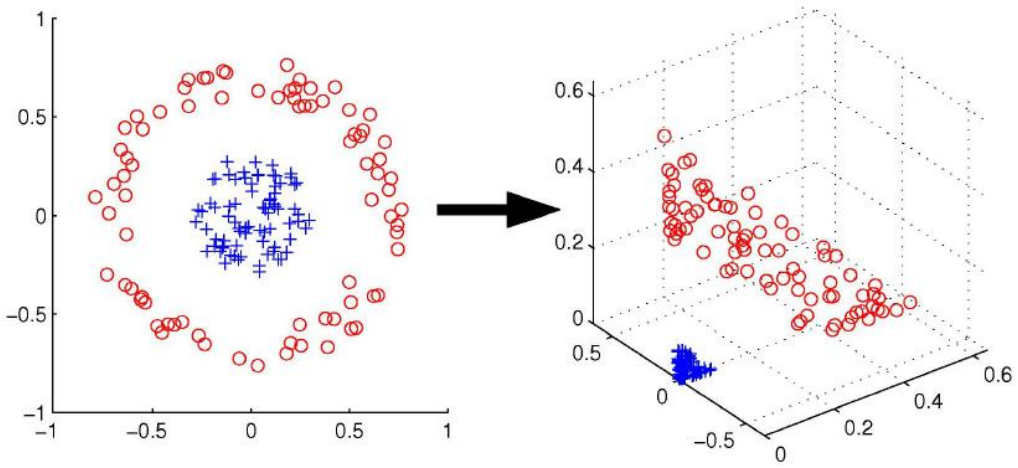
separable

The kernel trick

non-linear mapping to F

1. high-D space
2. infinite-D countable space :
3. function space (Hilbert space)

$$\Phi : \mathbf{x} \rightarrow \phi(\mathbf{x}), \Re^d \rightarrow F$$



example: $(x_1, x_2) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1 \times x_2)$

SVMs: the kernel trick

Problem: the dimension of $\Phi(\mathbf{x})$ can be very large, making w hard to represent explicitly in memory, and hard for the QP to solve.

The Representer theorem (Kimeldorf & Wahba, 1971) shows that (for SVMs as a special case):

$$w = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i)$$

for some variables α. Instead of optimizing w directly we can thus optimize α.

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x})$$

We call $K(\mathbf{x}_i, \mathbf{x}) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) =< \phi(\mathbf{x}_i), \phi(\mathbf{x}) >$ the kernel function.

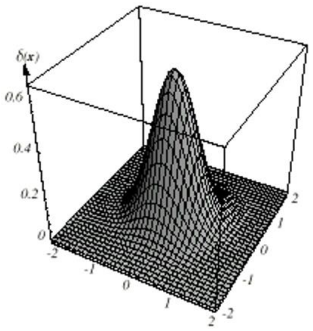
Defining kernels

$$sign(\sum_i \alpha_i \times K(\mathbf{x}_i, \mathbf{x}))$$

There are two strategies to compute $K(\mathbf{x}_i, \mathbf{x}))$

2. If we do not know the projection function, then e.g.

$$K(\mathbf{x}_1, \mathbf{x}_2) = e^{-||\mathbf{x}_1 - \mathbf{x}_2||^2/c}$$

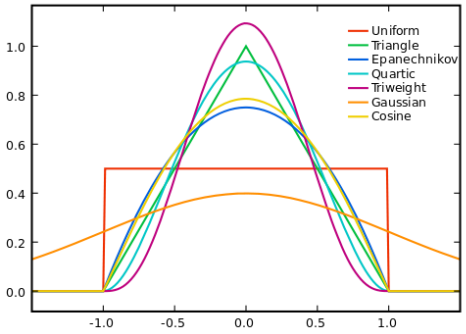


More kernel functions:
http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/

$$K(\mathbf{x}_1, \mathbf{x}_2) = (< \mathbf{x}_1, \mathbf{x}_2 > +\theta)^d$$

$$K(\mathbf{x}_1, \mathbf{x}_2) = tanh(\alpha < \mathbf{x}_1, \mathbf{x}_2 > +\theta)$$

$$K(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{\sqrt{||\mathbf{x}_1 - \mathbf{x}_2||^2 + c^2}}$$



https://en.wikipedia.org/wiki/Kernel_(statistics)

Understanding the kernel

Turn a parametric (explicit) representation into a non-parametric (implicit) form.

$$sign(\mathbf{w}^T\mathbf{x} + b) \longrightarrow sign(\textstyle\sum_i \alpha_i \times K(\mathbf{x}_i, \mathbf{x}))$$

Next: to see how is the solution in Kernel SVM obtained.

This can be simply illustrated in the Ridge Regression Classifier.

An example

Let's look at a simpler case (ridge regression) with constant λ:

Find: $\arg\min_{\mathbf{w}} \quad \lambda||\mathbf{w}||^2 + \sum_{i=1}^{n}(y_i - \mathbf{w} \cdot \mathbf{x}_i)^2$

A. It is convex and also has a closed-form (analytic) solution.

B. It is convex but without a closed-form solution.

C. It is non-convex but can be solved using gradient descent.

D. It is non-convex and has no clear solutions.

An example

Let's look at a simpler case (ridge regression) with constant λ:

Find: $\arg\min_{\mathbf{w}} \quad \lambda||\mathbf{w}||^2 + \sum_{i=1}^{n}(y_i - \mathbf{w} \cdot \mathbf{x}_i)^2$

A. It is convex and also has a closed-form (analytic) solution. Using gradient descent is fine too.

B. It is convex but without a closed-form solution.

C. It is non-convex but can be solved using gradient descent.

D. It is non-convex and has no clear solutions.

Ridge regression and kernel

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

$$\mathbf{w}^* = (X^T X + \lambda I_m)^{-1} X^T Y$$

$$\mathbf{w}^* = \arg min_{\mathbf{w}} = (X\mathbf{w} - Y)^T (X\mathbf{w} - Y) + \lambda \mathbf{w}^T \mathbf{w}$$

$$(X^T X + \lambda I)\mathbf{w}^* = X^T Y$$

$$\mathbf{w}^* = \tfrac{1}{\lambda}(X^T Y - X^T X \mathbf{w}^*)$$

$$\Downarrow$$

$$= X^T \mathbf{a}$$

$$\Downarrow$$

$$\mathbf{a} = \tfrac{1}{\lambda}(Y - X\mathbf{w}^*)$$

$$\Downarrow$$

$$\lambda \mathbf{a} = (Y - X X^T \mathbf{a})$$

$$\Downarrow$$

$$X X^T \mathbf{a} + \lambda \mathbf{a} = Y$$

$$\Downarrow$$

$$(X X^T + \lambda I)\mathbf{a} = Y$$

$$\Downarrow$$

$$\mathbf{a} = (X X^T + \lambda I)^{-1} Y$$

$$\Downarrow$$

$$\mathbf{w}^* = X^T (X X^T + \lambda I_n)^{-1} Y$$

Explanations of duality

Let's look at a simpler case (ridge regression) with constant λ:

Find: $\arg\min_{\mathbf{w}} \quad \lambda||\mathbf{w}||^2 + \times \sum_{i=1}^{n}(y_i - \mathbf{w} \cdot \mathbf{x}_i)^2$

$$\mathbf{w}^* = \sum_i \alpha_i \times \mathbf{x}_i$$

Learded classifier: $\text{sign}(\mathbf{w}^* \cdot \mathbf{x}) = sign(\sum_i \alpha_i \times \mathbf{x}_i \cdot \mathbf{x})$

Explanations of duality

$$\mathbf{w}^* = \sum_i \alpha_i \times \mathbf{x}_i$$

Learded classifier: $\text{sign}(\mathbf{w} \cdot \mathbf{x}) = sign(\sum_i \alpha_i \times \mathbf{x}_i \cdot \mathbf{x})$

A magic here is in training:
we only need to know $\mathbf{x_i} \cdot \mathbf{x_j} =< \mathbf{x_i}, \mathbf{x_j} >, \forall i, j$

In testing:
we only need to know $\mathbf{x_i} \cdot \mathbf{x} =< \mathbf{x_i}, \mathbf{x} >, \forall i$

The original feature representation of $\mathbf{x}_i$ and $\mathbf{x}$ can be implicit.

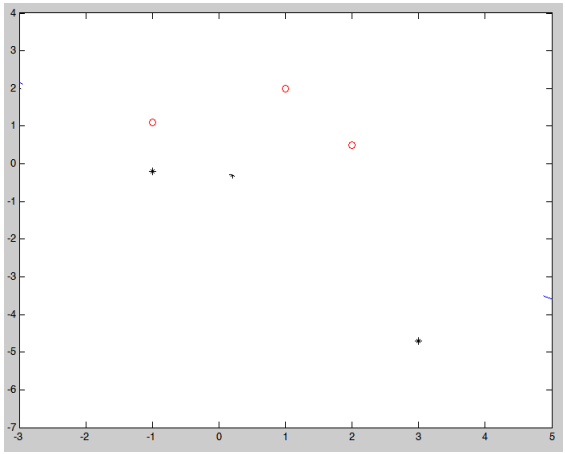
The difference between

$X^T X$ and $X X^T$

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

Feature space: $\mathbf{w}^* = (X^T X + \lambda I_m)^{-1} X^T Y$

Kernel space: $\mathbf{w}^* = X^T (X X^T + \lambda I_n)^{-1} Y$



$$X = \begin{pmatrix} 1.0 & 2.0 \\ 2.0 & 0.5 \\ -1.0 & 1.1 \\ -1.0 & -0.2 \\ 3.0 & -4.5 \\ 0.2 & -0.29 \end{pmatrix}$$

$$Y = \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix}$$

$$X^T X = \begin{pmatrix} 1.0 & 2.0 & -1.0 & -1.0 & 3.0 & 0.2 \\ 2.0 & 0.5 & 1.1 & -0.2 & -4.5 & -0.29 \end{pmatrix} \begin{pmatrix} 1.0 & 2.0 \\ 2.0 & 0.5 \\ -1.0 & 1.1 \\ -1.0 & -0.2 \\ 3.0 & -4.5 \\ 0.2 & -0.29 \end{pmatrix} = \begin{pmatrix} 16.04 & -11.46 \\ -11.45 & 25.83 \end{pmatrix}$$

$$X X^T = \begin{pmatrix} 1.0 & 2.0 \\ 2.0 & 0.5 \\ -1.0 & 1.1 \\ -1.0 & -0.2 \\ 3.0 & -4.5 \\ 0.2 & -0.29 \end{pmatrix} \begin{pmatrix} 1.0 & 2.0 & -1.0 & -1.0 & 3.0 & 0.2 \\ 2.0 & 0.5 & 1.1 & -0.2 & -4.5 & -0.29 \end{pmatrix} = \begin{pmatrix} 5. & 3 & 1.2 & -1.4 & -6. & -0.38 \\ 3. & 4.25 & -1.45 & -2.1 & 3.75 & 0.255 \\ 1.2 & -1.45 & 2.21 & 0.78 & -7.95 & -0.52 \\ -1.4 & -2.1 & 0.78 & 1.04 & -2.1 & -0.14 \\ -6. & 3.75 & -7.95 & -2.1 & 29.25 & 1.91 \\ -0.38 & 0.26 & -0.52 & -0.14 & 1.9 & 0.12 \end{pmatrix}$$

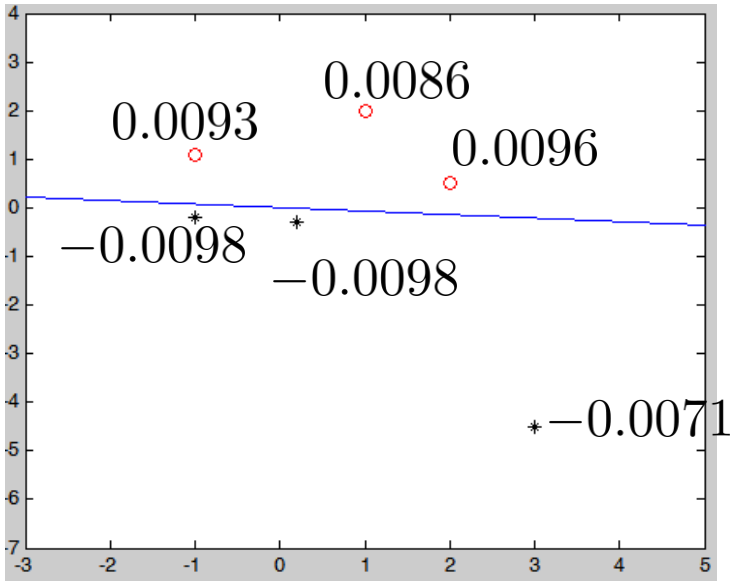
An example

Let's look at a simpler case (ridge regression) with constant λ:

$$\text{Find: } \arg\min_{\mathbf{w}} \quad \lambda||\mathbf{w}||^2 + \sum_{i=1}^{n}(y_i - \mathbf{w}\cdot\mathbf{x}_i)^2$$

$$\mathbf{w}^* = X^T(G + \lambda I_n)^{-1}Y$$

$$\mathbf{w}^* = \sum_i \alpha_i \times \mathbf{x}_i$$



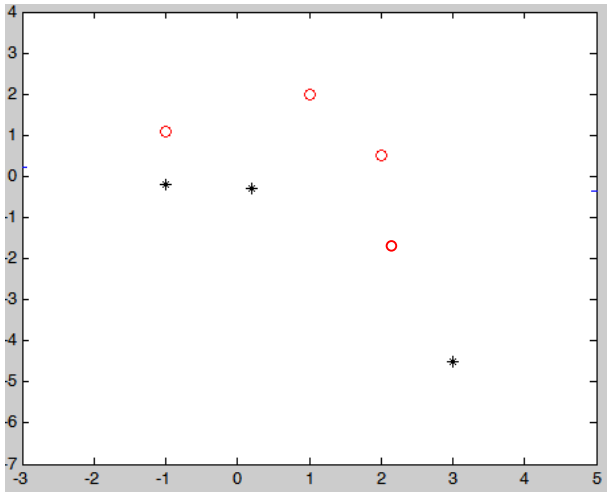
$$X = \begin{pmatrix} 1.0 & 2.0 \\ 2.0 & 0.5 \\ -1.0 & 1.1 \\ -1.0 & -0.2 \\ 3.0 & -4.5 \\ 0.2 & -0.29 \end{pmatrix} \qquad Y = \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix} \qquad \lambda = 100$$

```
w=X'*inv(X*X'+100*eye(6))*Y;
```

$$w = \begin{pmatrix} 0.0051 \\ 0.0687 \end{pmatrix} \qquad \alpha = \begin{pmatrix} 0.0086 \\ 0.0096 \\ 0.0093 \\ -0.0098 \\ -0.0071 \\ -0.0098 \end{pmatrix}$$

A simpler case:
ridge regression

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$



Ideally : $\arg\min_{\mathbf{w}} C \times (\#training\ errors) + \frac{1}{2}||\mathbf{w}||^2$

Instead: $\arg\min_{\mathbf{w}} \quad C \times \sum_{i=1}^{n}(y_i - \mathbf{w} \cdot \mathbf{x}_i)^2 + ||\mathbf{w}||^2$

$$\mathbf{w}^* = \arg min_{\mathbf{w}} = C \times (X\mathbf{w} - Y)^T(X\mathbf{w} - Y) + \mathbf{w}^T\mathbf{w}$$

$$\frac{\partial[C \times (X\mathbf{w} - Y)^T(X\mathbf{w} - Y) + \mathbf{w}^T\mathbf{w}]}{\partial \mathbf{w}} \rightarrow 0$$

$$\mathbf{w}^* = X^T(C \times (XX^T) + I)^{-1}Y$$

I is an identity matrix.
Serving as a regularizer.

Explanations of duality

Let's look at a simpler case (ridge regression) with constant λ:

Find: $\arg\min_{\mathbf{w}} \quad \lambda||\mathbf{w}||^2 + \sum_{i=1}^{n}(y_i - \mathbf{w}\cdot\mathbf{x}_i)^2$

Let: $X = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_n)^T$ be the entire input data matrix.

Let: $Y = (y_1, y_2, .., y_n)^T$ be the taining labels.

We often use I to denote an identity matrix.

$$I_m = \begin{pmatrix} 1, 0, ..., 0 \\ 0, 1, ..., 0 \\ ... \\ 0, 0, ..., 1 \end{pmatrix}, m \times m$$

solution: $\mathbf{w}^* = (X^T X + \lambda I_m)^{-1} X^T Y$

$= X^T (G + \lambda I_n)^{-1} Y$

$$I_n = \begin{pmatrix} 1, 0, ..., 0 \\ 0, 1, ..., 0 \\ ... \\ 0, 0, ..., 1 \end{pmatrix}, n \times n$$

Explanations of duality

Let's look at a simpler case (ridge regression) with constant λ:

Find: $\arg\min_{\mathbf{w}} \quad \lambda||\mathbf{w}||^2 + \sum_{i=1}^{n}(y_i - \mathbf{w}\cdot\mathbf{x}_i)^2$

$$I_m = \begin{pmatrix} 1,0,...,0 \\ 0,1,...,0 \\ ... \\ 0,0,...,1 \end{pmatrix}, m \times m$$

$$\text{solution: } \mathbf{w} \quad = X^T(G + \lambda I_n)^{-1}Y$$

$$I_n = \begin{pmatrix} 1,0,...,0 \\ 0,1,...,0 \\ ... \\ 0,0,...,1 \end{pmatrix}, n \times n$$

where $G = XX^T$ is a Gram-matrix of dimension $n \times n$ (n is the number of samples), $G_{ij} = \mathbf{x}_i\mathbf{x}_j^T$

$$\mathbf{w} = \sum_i \alpha_i \times \mathbf{x}_i$$

In the end, the best parameter is a linear combination of the data samples with learned contributions (importance of each data point).

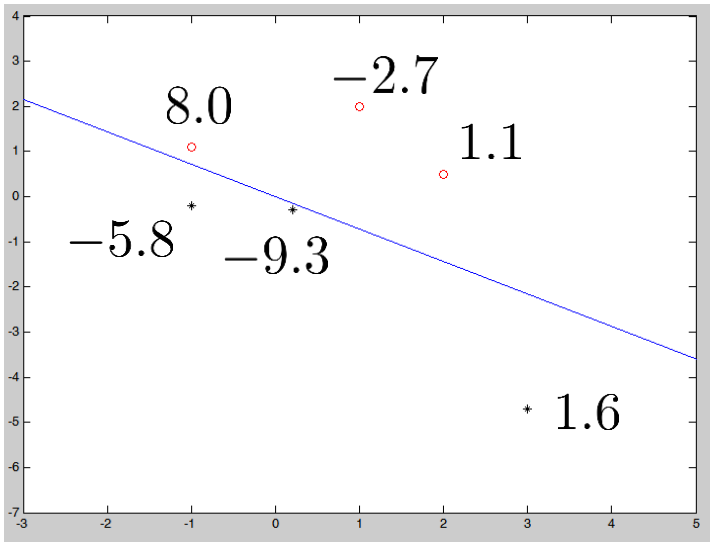
An example

Let's look at a simpler case (ridge regression) with constant λ:

Find: $\arg\min_{\mathbf{w}} \quad \lambda||\mathbf{w}||^2 + \sum_{i=1}^{n}(y_i - \mathbf{w}\cdot\mathbf{x}_i)^2$

$$\mathbf{w}^* = X^T(G + \lambda I_n)^{-1}Y$$

$$\mathbf{w}^* = \sum_i \alpha_i \times \mathbf{x}_i$$



$$X = \begin{pmatrix} 1.0 & 2.0 \\ 2.0 & 0.5 \\ -1.0 & 1.1 \\ -1.0 & -0.2 \\ 3.0 & -4.5 \\ 0.2 & -0.29 \end{pmatrix} \qquad Y = \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix} \qquad \lambda = 0.1$$

```
w*=X'*inv(X*X'+0.1*eye(6))*Y;
```

$$\mathbf{w}^* = \begin{pmatrix} 0.3245 \\ 0.4746 \end{pmatrix} \qquad \mathbf{a} = \begin{pmatrix} -2.7 \\ 1.1 \\ 8.0 \\ -5.8 \\ 1.6 \\ -9.3 \end{pmatrix}$$

SVM

Primal: Find: $\arg\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C \times \sum_{i=1}^{n}(1 - y_i \times (\mathbf{w} \cdot \mathbf{x}_i + b))_+$

Dual: Find $\arg\max_{\alpha_1,\ldots\alpha_n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} \alpha_j \alpha_i Q_{ij}$

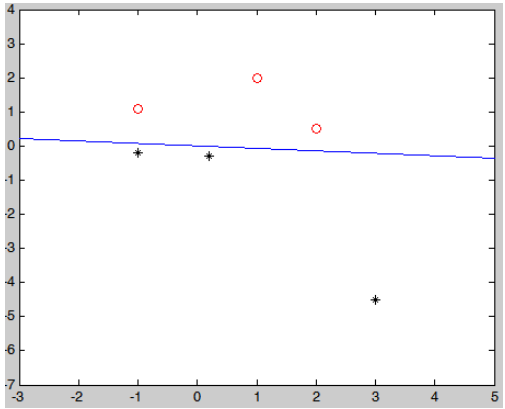
where $Q_{ji} = y_j y_i K(\mathbf{x}_j, \mathbf{x}_i)$ note: $\mathbf{x}_j \cdot \mathbf{x}_i$ is replaced by a more general form, kernel

Subject to constraints: $0 \leq \alpha_i \leq C, \forall i$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i$$

$$b^* = y_k(1 - \varepsilon_k) - \mathbf{w}^* \cdot \mathbf{x}_k \quad \text{where } k = \arg\max_k \alpha_k$$

Note α_i^* and y_i are scalar. $\mathbf{x}_i$ is data vector.

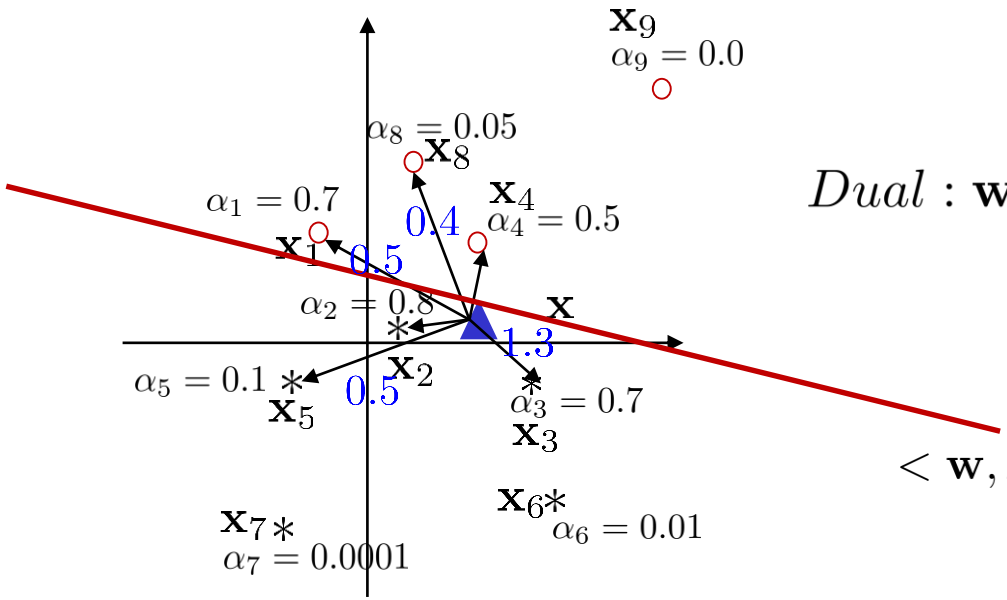


Most α_is are 0 and we only save non-zero data samples, which are the support vectors of our learned classifier.

Learned classifier: $sign(\sum_i \alpha_i y_i \times K(\mathbf{x}_i, \mathbf{x}))$

Understanding SVM?

Find: $\arg\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 + C \times \sum_{i=1}^{n}(1 - y_i \times (\mathbf{w} \cdot \mathbf{x}_i + b))_+$



$$Dual : \mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i$$

$$< \mathbf{w}, \mathbf{x} > +b = 0$$

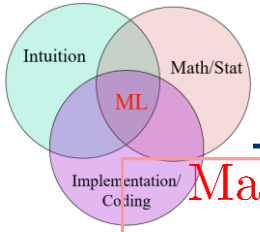
In testing: given an input data $\mathbf{x}$, make the prediction based on

$$sign(< \mathbf{w}, \mathbf{x} > +b) = sign(\sum_{i=1}^{n} \alpha_i y_i < \mathbf{x}_i, \mathbf{x} >)$$

$$< \mathbf{w}, \mathbf{x} > +b = 0.7 \times (+1) \times 0.5 + 0.8 \times (-1) \times 1.5 + 0.7 \times (-1) \times 1.3 + 0.5 \times (+1) \times 0.7 + 0.1 \times (-1) \times 0.5 + 0.05 \times (+1) \times 0.4$$

$$= -1.44$$

$$sign(< \mathbf{w}, \mathbf{x} > +b) = -1$$



Recap: Kernel-based Support Vector Machine

Math:

Training : Minimize $\mathcal{L}(\mathbf{w}, b) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))_+$

$\Longrightarrow$ Find $\arg\max_{\alpha_1,\ldots\alpha_n} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n}\alpha_j\alpha_i Q_{ij}$

$\mathbf{w}^* = \sum_{i=1}^{n}\alpha_i\mathbf{x}_i$

$b^* = y_k(1 - \varepsilon_k) - \mathbf{w}\cdot\mathbf{x}_k$ where $k = \arg\max_i \alpha_i$

Ridge regression: $\alpha = (C \times (XX^T) + I)^{-1}Y$

Testing : Learned classifier: $sign(\sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x}))$

Pros:

- It is very robust.
- Works very well in practice.
- Mathematically well-defined and can be extended to many places.

Cons:

- No intrinsic feature selection stage.
- May not be able to deal with large amount training data with high dimension due to its kernel.