COGS 118A, Spring 2019

Supervised Machine Learning Algorithms

Lecture 12: Cross-Validation and Nearest Neighborhood Classifier

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 will be provided.

Intuition about classification power

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

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

$$e_{training}^{(f)} \rightarrow 0$$

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

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

- 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.

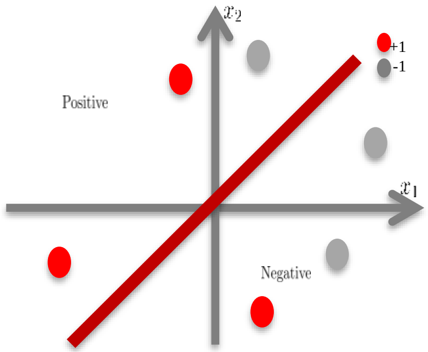
- If you are checking on n points, then there are 2^n possibilities to verify. Failing on any one of the situations will deem the classifier incapable of shattering n points.

- This is like a bank stress test.

VC dimension for the linear classifiers we have learned so far.

$$f(\mathbf{x}; \mathbf{w}, b) = \begin{cases} +1 & if\ \mathbf{w}^T\mathbf{x} + b \geq 0 \\ -1 & otherwise \end{cases}$$

$\mathbf{x}, \mathbf{w} \in \mathbb{R}^r,\ b \in \mathbb{R}$



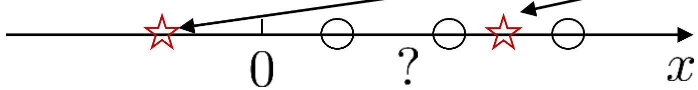
- Perceptron
- Logistic regression classifier
- Support vector machine (SVM)

Their VC dimension (h) is $r + 1$ in these linear classifier cases.

Understanding shattering

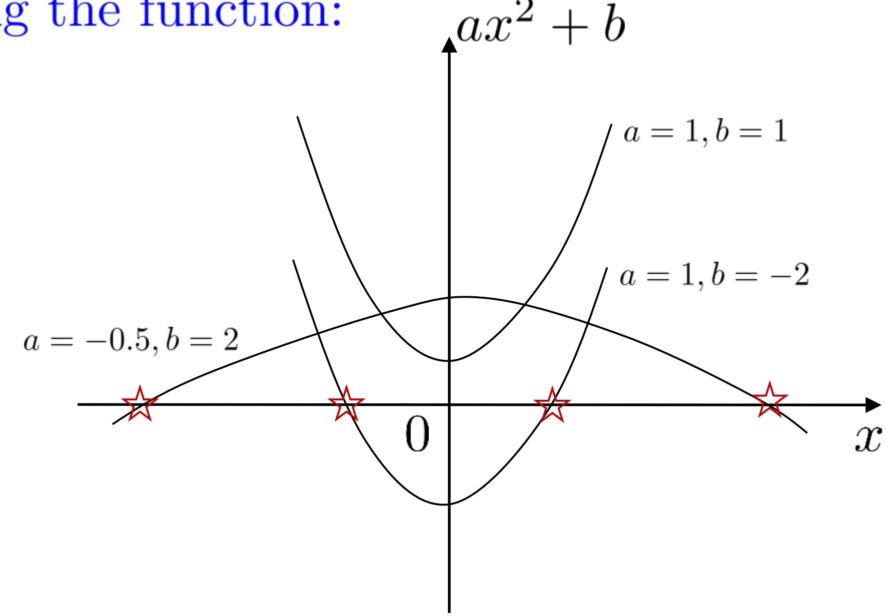
Example: What is the VC-dimension for $f(x; a, b) = sign(ax^2 + b)$, $x, a, b \in \mathbb{R}$?

Understanding the problem: Decision boundary consists of a set of points on the axis.



e.g. for $\forall x$ such that $ax^2 + b = 0$.

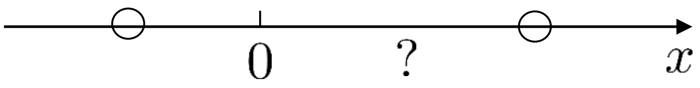
Understanding the function:

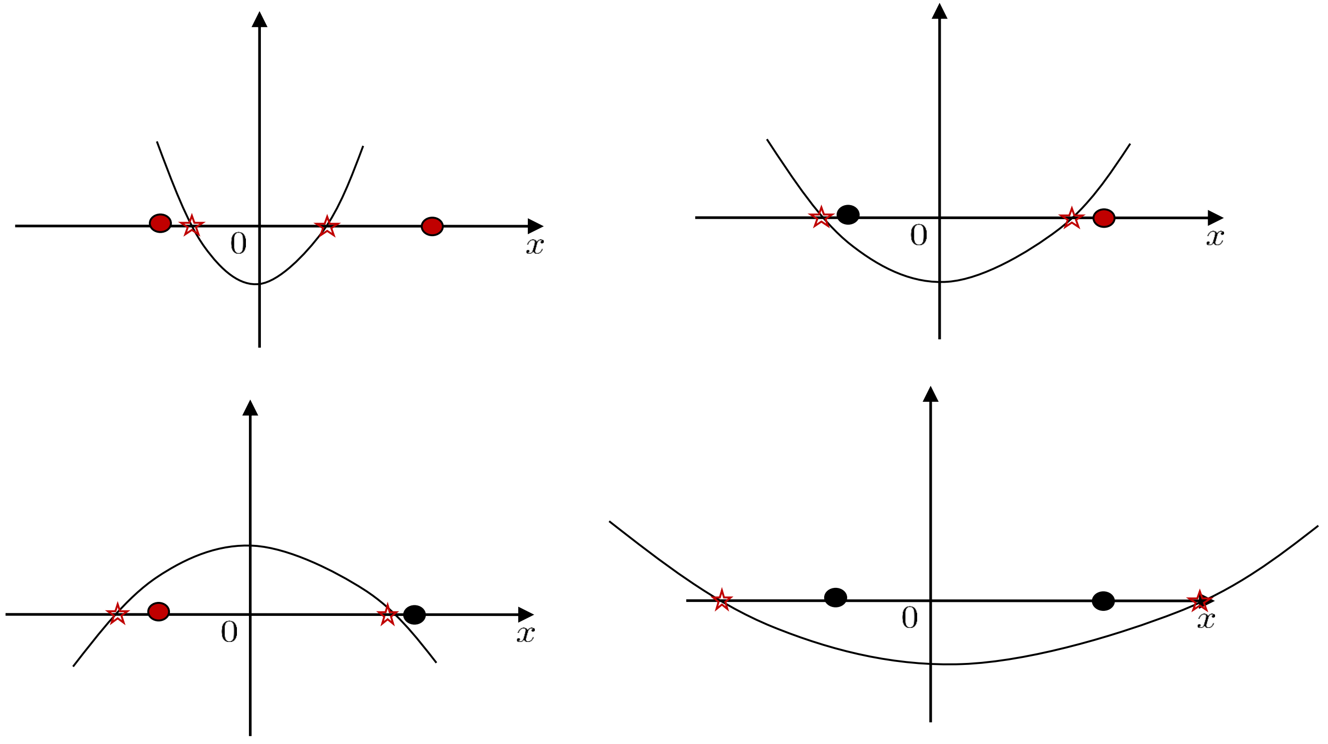


Understanding shattering

Example: What is the VC-dimension for $f(x; a, b) = sign(ax^2 + b)$, $x, a, b \in \mathbb{R}$?

Two points:





VC-dimension

Theory: The VC dimension (h) of the set of oriented hyperplanes in $\mathbb{R}^r$ is $r+1$, since we can always choose r+1 points, and then choose one of the points as origin, such that the position vectors of the remaining r points are linearly independent, but can never choose r+2 such points.

For a linear classifier:

$$f(\mathbf{x}; \mathbf{w}, b) = sign(\mathbf{w}^T\mathbf{x} + b),$$

$$\mathbf{x}, \mathbf{w} \in \mathbb{R}^r,\ b \in \mathbb{R}$$

Total number of parameters: $r+1$.

- VC dimension (h) reports the maximum number of points a classifier f can shatter.
- It is done by checking the number of shattering sequentially from 1,2,3,… until it fails.

VC Dimension

- The concept and theory of VC dimension, named after Vapnik and Chervonenkis, defines the maximum capability of a classifier f.

- VC dimension (h) reports the maximum number of points a classifier f can shatter.

- It is done by checking the number of shattering sequentially from 1,2,3,… until it fails.

- When checking on number n, you only need to find an existence of n non-overlapping points (no need to shatter all possible n points).

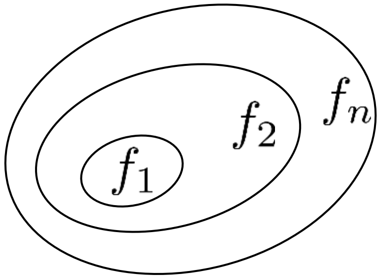
- However, once the n points are given, you need to make sure ALL possible labeling configurations for these n points can be well classified. Otherwise, it's a failure.

Structural Risk Minimization

Let: $\phi(f)$=the set of functions representable by f

Suppose: $\phi(f_1) \subseteq \phi(f_2) \subseteq \cdots \phi(f_n)$

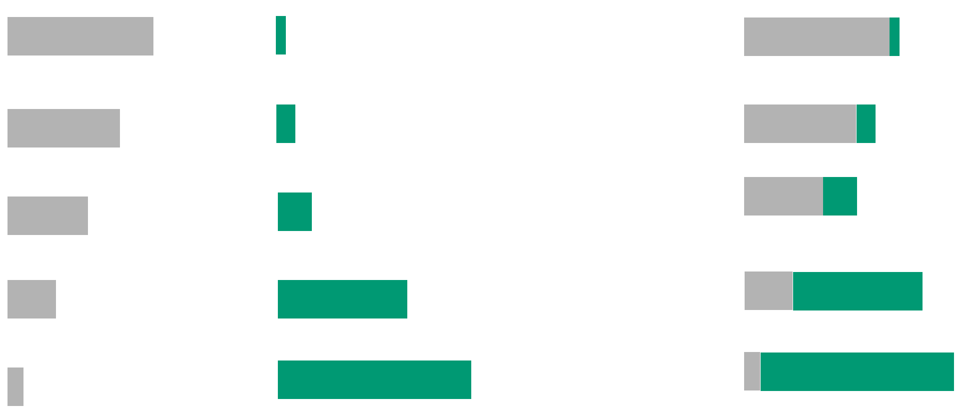
Then: $h(f_1) \leq h(f_2) \leq \cdots h(f_n)$

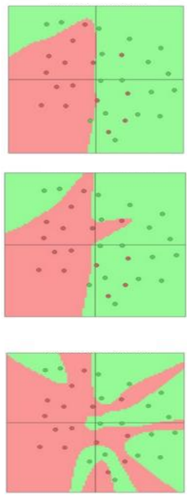


We are trying to decide which machine to use.

We train each machine and make a table: $e_{testing} \leq e_{training} + \sqrt{\frac{h(\log(2n/h+1)-\log(\eta/4)}{n}}$

	i	f_i	$e_{training}$	$\sqrt{\frac{h(\log(2n/h+1)-\log(\eta/4)}{n}}$ generalization	upper bound $e_{testing}$	choice
A	1	f_1				
B	2	f_2				
C	3	f_3				
D	4	f_4				
E	5	f_5				



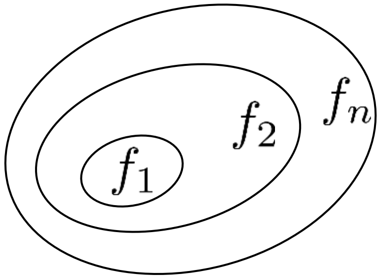


Structural Risk Minimization

Let: $\phi(f)$=the set of functions representable by f

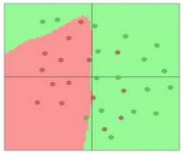
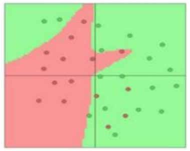
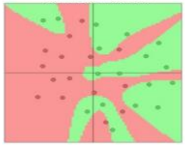
Suppose: $\phi(f_1) \subseteq \phi(f_2) \subseteq \cdots \phi(f_n)$

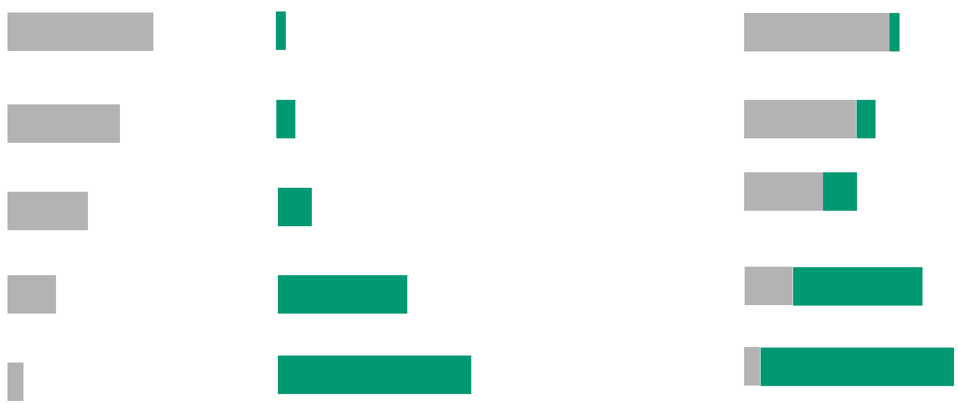
Then: $h(f_1) \leq h(f_2) \leq \cdots h(f_n)$



We are trying to decide which machine to use.

We train each machine and make a table: $e_{testing} \leq e_{training} + \sqrt{\frac{h(\log(2n/h+1)-\log(\eta/4)}{n}}$

	i	f_i	$e_{training}$	$\sqrt{\frac{h(\log(2n/h+1)-\log(\eta/4)}{n}}$ generalization	upper bound $e_{testing}$	choice	
A	1	f_1					
B	2	f_2					
C	3	f_3					
D	4	f_4					
E	5	f_5					



Recap: Classifier Complexity and VC Dimension

Math:

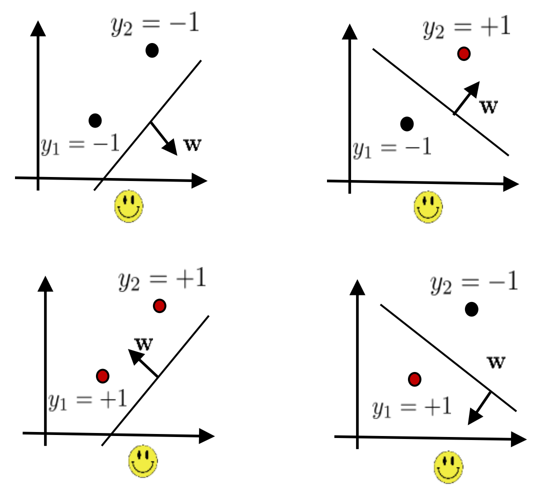
$$e_{testing} \leq e_{training} + \sqrt{\frac{h(\log(2n/h+1)) - \log(\eta/4)}{n}}$$

h : the complexity (VC dimension) of a classifier
n: the number of training samples

η: confidence level, can be ignored for the moment

Intituion:

- The concept and theory of VC dimension, named after Vapnik and Chervonenkis, defines the maximum capability of a classifier f.
- VC dimension (h) reports the maximum number of points a classifier f can shatter.
- It is done by checking the number of shattering sequentially from 1,2,3,… until it fails.



To obtain a good classifier

Which is the least effective when dealing with big data?

A. Increase your training data size.

B. Reduce your classifier complexity.

C. Design/learn good features.

To obtain a good classifier

1. Increase your training data size (big data)!

2. Reduce your classifier complexity.

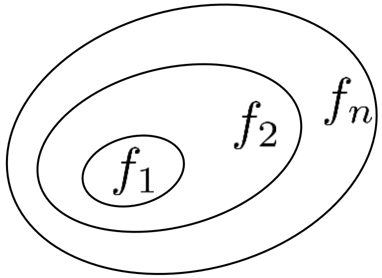
3. After the training dataset is given for the chosen classifier, perform optimization to find the optimal parameters/hyper-parameters.

Structural Risk Minimization

Let: $\phi(f)$=the set of functions representable by f

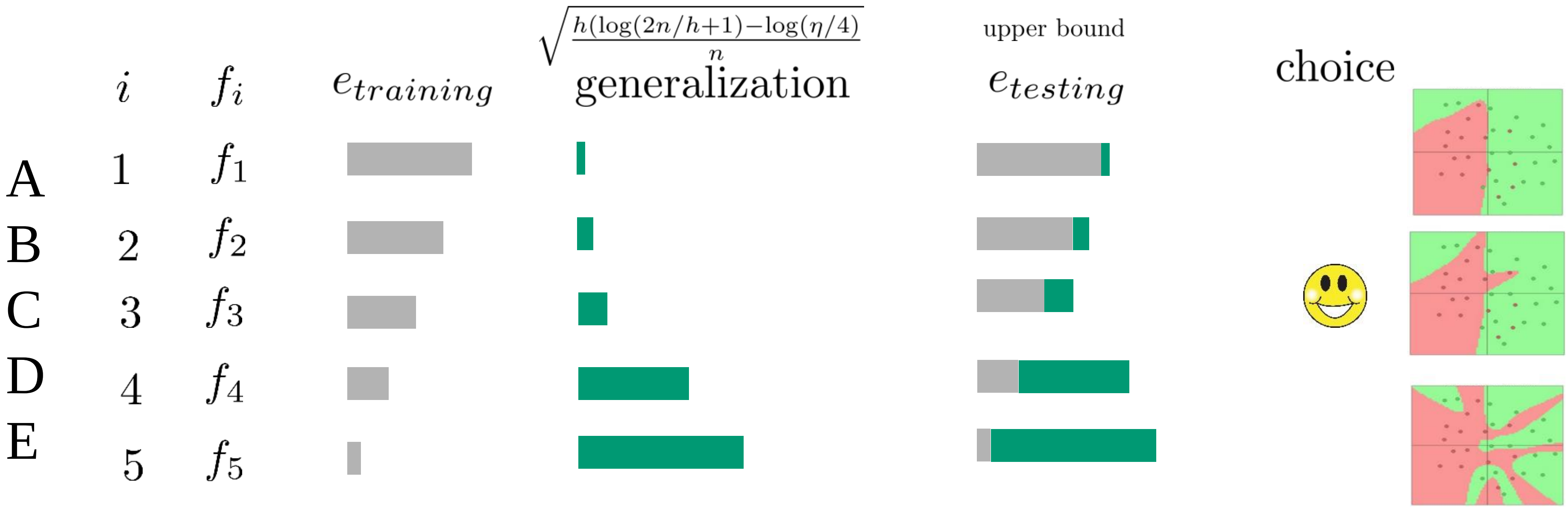
Suppose: $\phi(f_1) \subseteq \phi(f_2) \subseteq \cdots \phi(f_n)$

Then: $h(f_1) \leq h(f_2) \leq \cdots h(f_n)$



We are trying to decide which machine to use.

We train each machine and make a table: $e_{testing} \leq e_{training} + \sqrt{\frac{h(\log(2n/h+1)-\log(\eta/4))}{n}}$



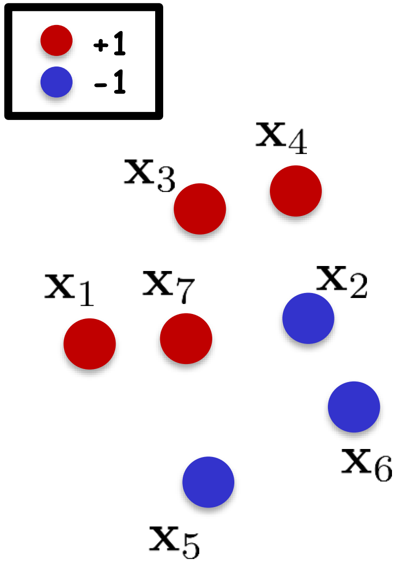
Cross-Validation

Why?

The VC dimension theory is nice but impractical in many real-world situations.

Cross-validation

(works for both regression and classification)



$$S_{training} =$$

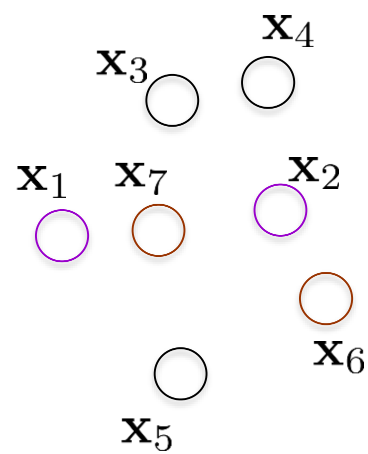
$$\{(\mathbf{x}_1, +1), (\mathbf{x}_2, -1), (\mathbf{x}_3, +1), (\mathbf{x}_4, +1), (\mathbf{x}_5, -1), (\mathbf{x}_6, -1), (\mathbf{x}_7, +1)\}$$

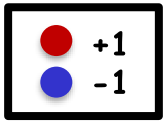
$$S_{training} = \{Sub_1, Sub_2, Sub_3\}$$

$$Sub_1 = \{(\mathbf{x}_1, +1), (\mathbf{x}_2, -1)\}$$

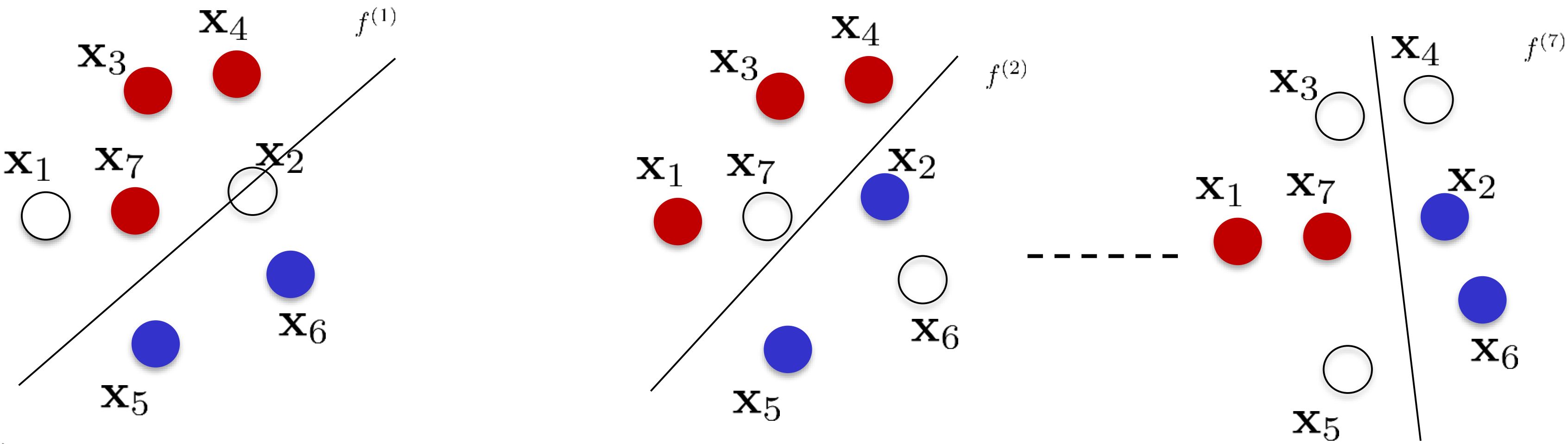
$$Sub_2 = \{(\mathbf{x}_6, -1), (\mathbf{x}_7, +1)\}$$

$$Sub_3 = \{(\mathbf{x}_3, +1), (\mathbf{x}_4, +1), (\mathbf{x}_5, -1)\}$$





Cross-validation
(works for both regression and classification)



$$S^{(1)}_{training} = \{(\mathbf{x}_3, +1), (\mathbf{x}_4, +1), (\mathbf{x}_5, -1), (\mathbf{x}_6, -1), (\mathbf{x}_7, +1)\}$$

Perform training to obtain $f^{(1)}$

$$S^{(1)}_{testing} = \{(\mathbf{x}_1, +1), (\mathbf{x}_2, -1), \}$$

$$e_{testing}(f^{(1)}) = \tfrac{1}{2}[\mathbf{1}(y_1 \neq f^{(1)}(\mathbf{x}_1)) + \mathbf{1}(y_2 \neq f^{(1)}(\mathbf{x}_2))]$$

$$S^{(2)}_{training} = \{(\mathbf{x}_1, +1), (\mathbf{x}_3, +1), (\mathbf{x}_4, +1), (\mathbf{x}_5, -1), (\mathbf{x}_2, -1\}$$

Perform training to obtain $f^{(2)}$

$$S^{(2)}_{testing} = \{(\mathbf{x}_6, -1), (\mathbf{x}_7, +1)\}$$

$$e_{testing}(f^{(2)})$$

$$S^{(3)}_{training} = \{(\mathbf{x}_1, +1), (\mathbf{x}_2, -1), (\mathbf{x}_7, +1), (\mathbf{x}_6, -1)\}$$

Perform training to obtain $f^{(k)}$

$$S^{(k)}_{testing} = \{(\mathbf{x}_3, +1), (\mathbf{x}_4, +1), (\mathbf{x}_5, -1)\}$$

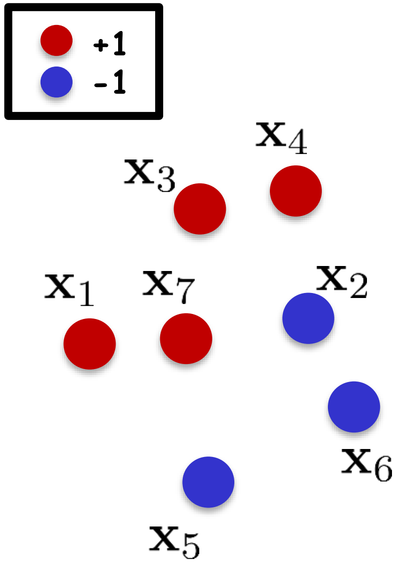
$$e_{testing}(f^{(k)})$$

We compute the cross-validation error by

$$\bar{e} = \frac{1}{k} \sum_i e_{testing}(f^{(i)})$$

$$var = \frac{1}{k} \sum_i (e_{testing}(f^{(i)}) - \bar{e})^2$$

K-fold cross-validation
(works for both regression and classification)



$$S_{training} =$$

$$\{(\mathbf{x}_1, +1), (\mathbf{x}_2, -1), (\mathbf{x}_3, +1), (\mathbf{x}_4, +1), (\mathbf{x}_5, -1), (\mathbf{x}_6, -1), (\mathbf{x}_7, +1)\}$$

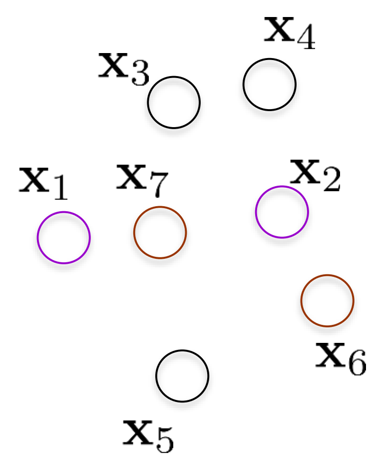
$$\Downarrow$$

$$S_{training} = \{Sub_1, Sub_2, Sub_3\}$$

$$Sub_1 = \{(\mathbf{x}_1, +1), (\mathbf{x}_2, -1)\}$$

$$Sub_2 = \{(\mathbf{x}_6, -1), (\mathbf{x}_7, +1)\}$$

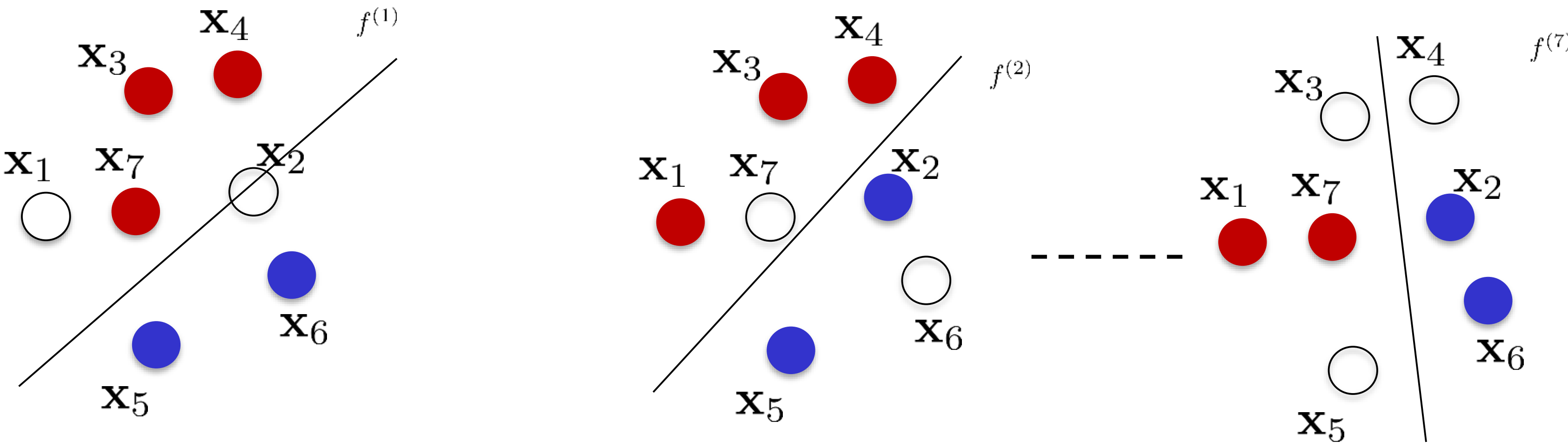
$$Sub_3 = \{(\mathbf{x}_3, +1), (\mathbf{x}_4, +1), (\mathbf{x}_5, -1)\}$$



For i=1 to k
Train classifier $f^{(i)}$ on a set that includes all the subsets but Sub_i and compute the corresponding training error $e_{train}(f^{(i)})$.
Compute the testing error $e(f^{(i)})$ on Sub_i.
Fine-tune the model and hyper-parameter to minimize: $\bar{e} = \frac{1}{k} \sum_i e(f^{(i)})$.

K-fold Cross-validation

(works for both regression and classification)



We use $\bar{e} = \frac{1}{k}\sum_i e(f^{(i)})$ and var to decide on:

- Which model (linear or nonlinear ones) we should use?
- How to fine-tune the hyper-parameter?
- Have we collected enough data for training?
- Is our hypothesis valid statistically significant?

We are supposed to have small values for both $\bar{e}$ and var if our hypothsis is statistically significant.

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.

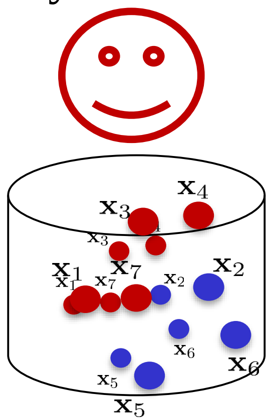
Cons:

It is time-consuming to compute.

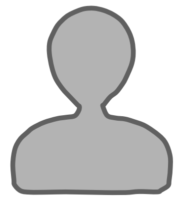
Not needed when your data is truly large: keep a hold-out dataset is sufficient.

How would you use cross-validation: example 1

your boss

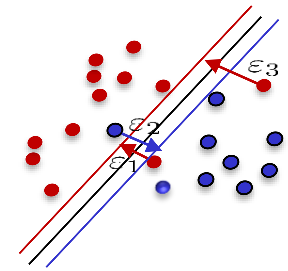


Labeled
Training Data



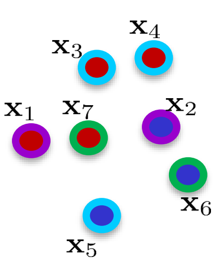
Your task:

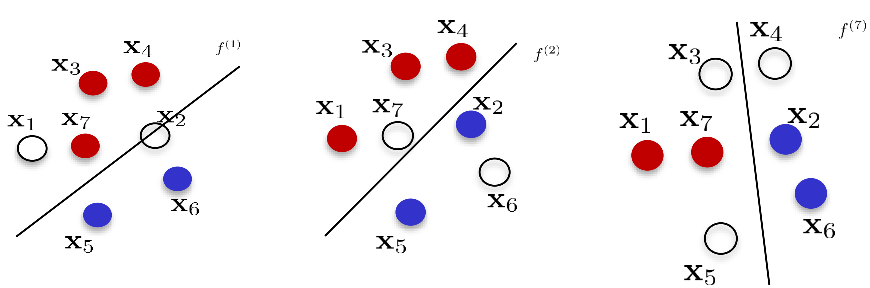
To obtain the "optimal" classifier using the given training data. Find the best hyper-parameter value for C.



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

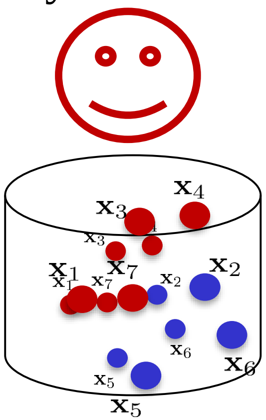
C. $\bar{e}_{C=1.0} = 0.15$

D. $\bar{e}_{C=10.0} = 0.10$

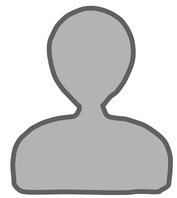
E. $\bar{e}_{C=100.0} = 0.25$

How would you use cross-validation: example 2

your boss



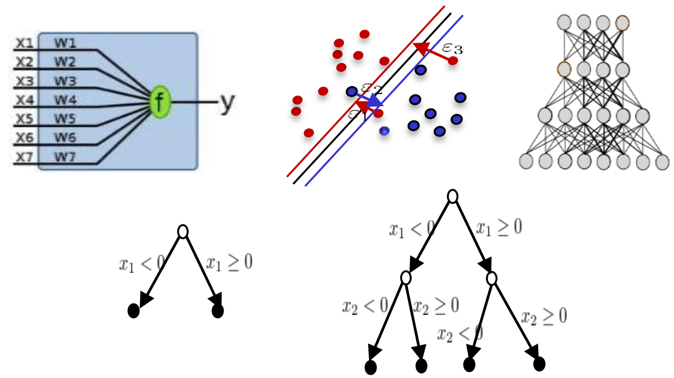
Labeled
Training Data

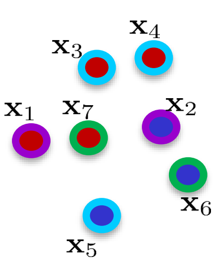


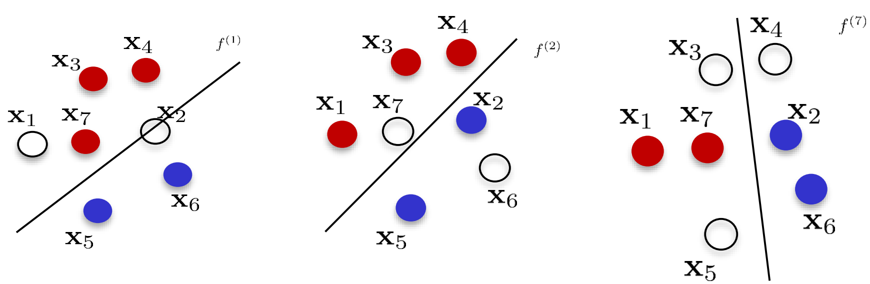
Your task:

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.

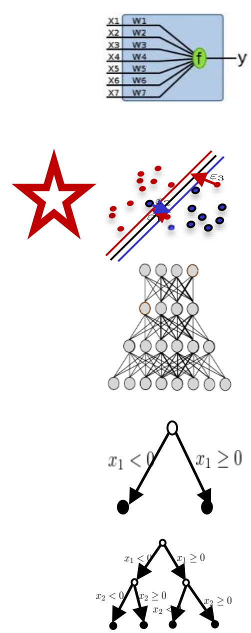






$e(f^{(1)})$ $e(f^{(2)})$ $e(f^{(3)})$

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



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

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

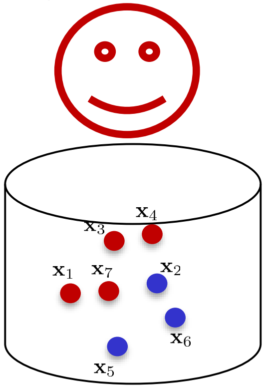
C. $\bar{e}_{NN} = 0.27$

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

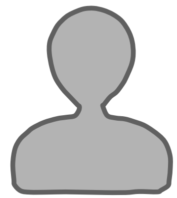
E. $\bar{e}_{Tree2} = 0.35$

Now you have chosen SVM

your boss



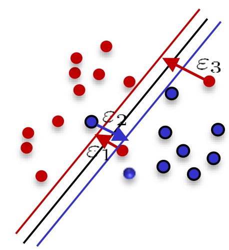
Labeled Training Data



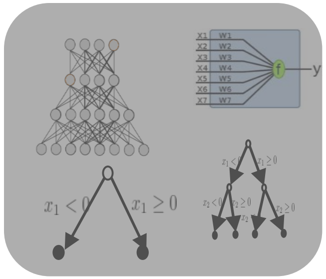
Your task:

To obtain the
“optimal” classifier
using the given
training data.

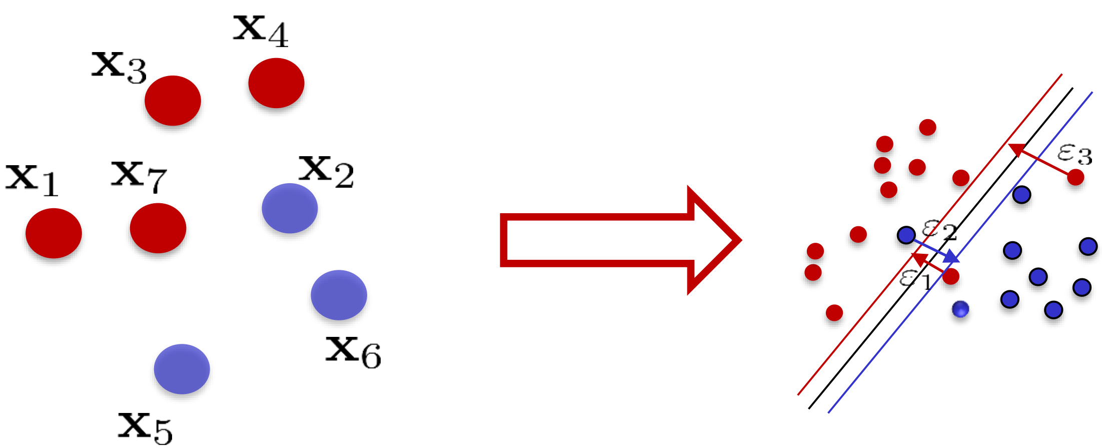
After cross-validation,
you have chosen SVM.



Yes



No



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.

Nearest Neighborhood Classifier

Chapter, “Non-parametric Techniques”, R. Duda, P. Hart, D. Stork, "Pattern Classification", second edition, 2000

Nonparametric estimation

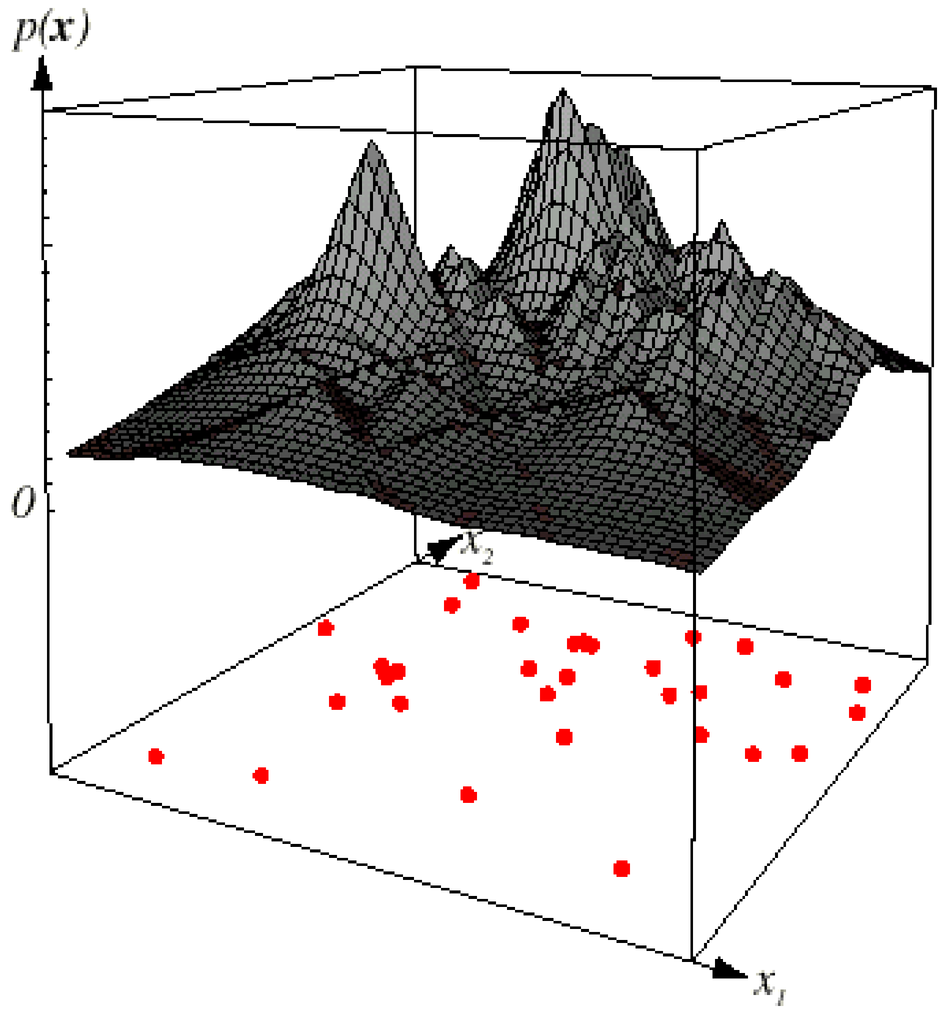
Parametric	Non-parametric
$y = f(x)$	$y = \sum_{k=1}^{K} \alpha_k f_k(x)$
Flooding? weather + month + location	Flooding? Every 12/06 in the history.

In practical applications, it is often difficult to know the parametric forms of underlying distributions (exemplar-based)

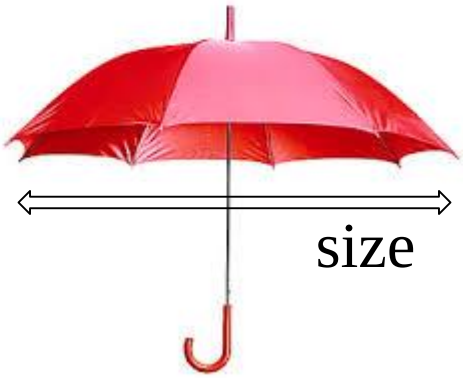
Parametric methods may lead to underfitting

Non-parametric models are direct and easier to implement.

K_n-Nearest Neighbor Estimation Examples – cont.



Understanding the Kernel



Kernel function $f(x)$.

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

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

Nonparametric Estimation

If we assume $p(\mathbf{x})$ to be continuous and the region R to be small, we have

$$P = \int_R p(\mathbf{x}) d\mathbf{x} \approx p(\mathbf{x}) \times V$$

where V refers to the volume of region R.

Overall:

$$p(\mathbf{x}) \cong \frac{k(\mathbf{x})}{l \times V(\mathbf{x})}$$

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

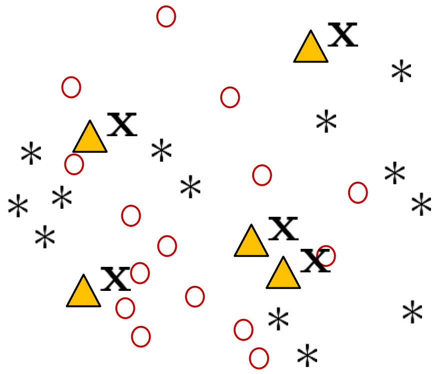
$k_l(\mathbf{x})$: number of samples within the Region.

$V_l(\mathbf{x})$: 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

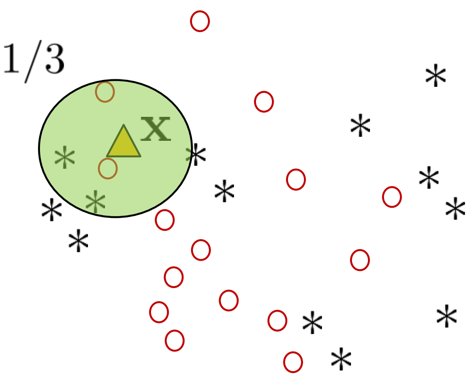
l: a hyper-parameter

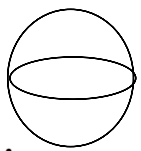
$k_l(\mathbf{x})$: number of samples within the Region.

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

○ $y = +1$ $\quad *$ $y = -1$

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





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

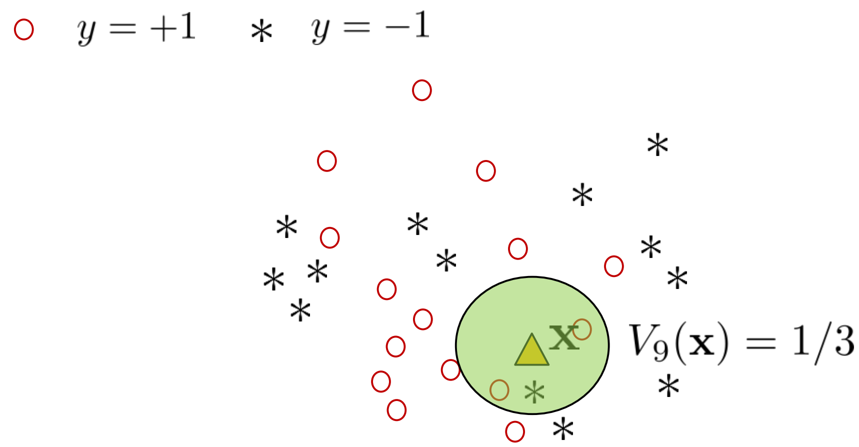
$$p_l(\mathbf{x}) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})}$$

$\mathbf{x}$: 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 $1/3$.

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

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

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

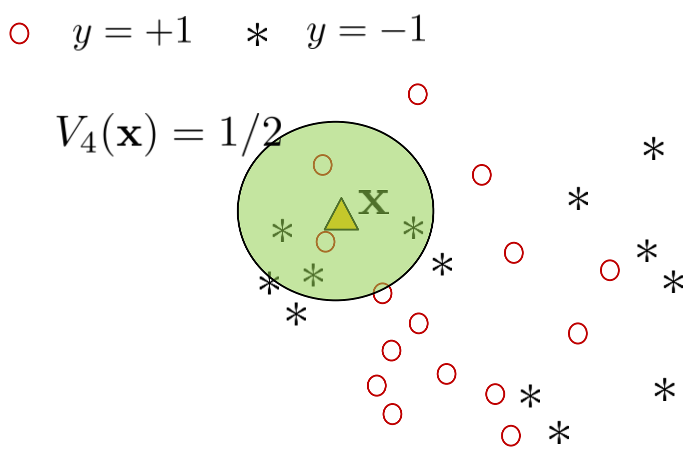
How to compute?
Strategy I

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

$\mathbf{x}$: 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.



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

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

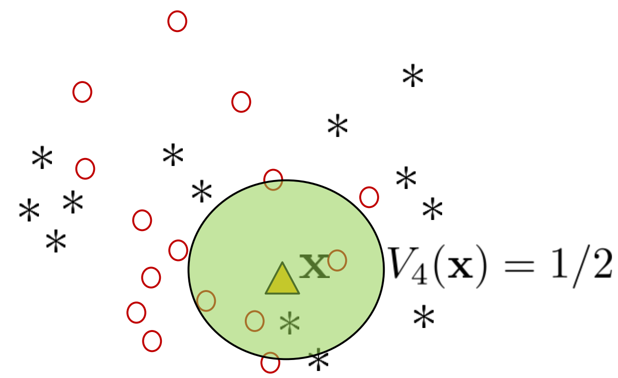
$\mathbf{x}$: a test/query data sample

l: a hyper-parameter

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

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

○ $y = +1$ $\ast$ $y = -1$



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

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

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

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

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

How to compute?

Strategy II

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

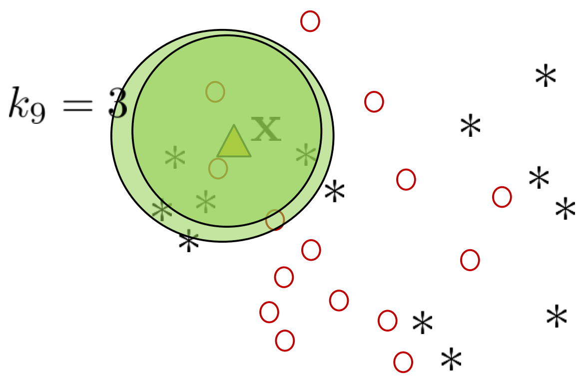
$\mathbf{x}$: a test/query data sample

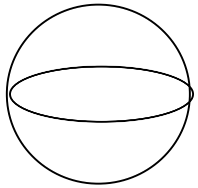
l: a hyper-parameter

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

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

○ $y = +1$ $\quad *$ $y = -1$





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

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

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

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

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

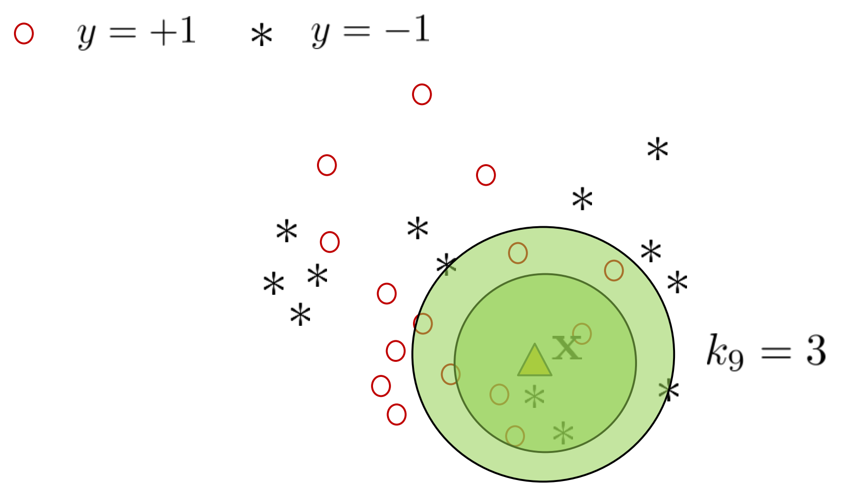
How to compute?
Strategy II

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

$\mathbf{x}$: 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.

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

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

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

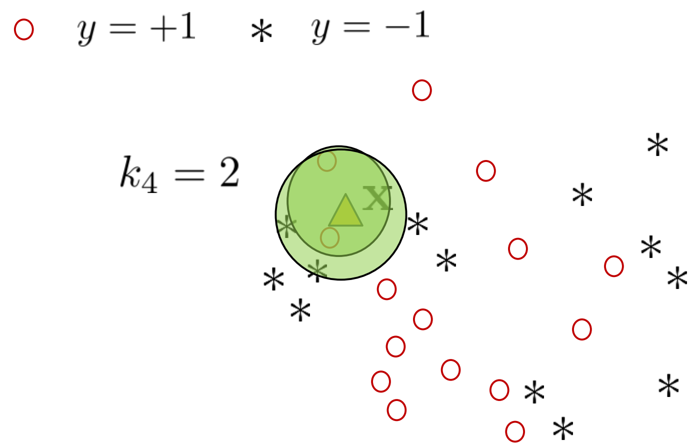
How to compute?
Strategy II

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

$\mathbf{x}$: 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.

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

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

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

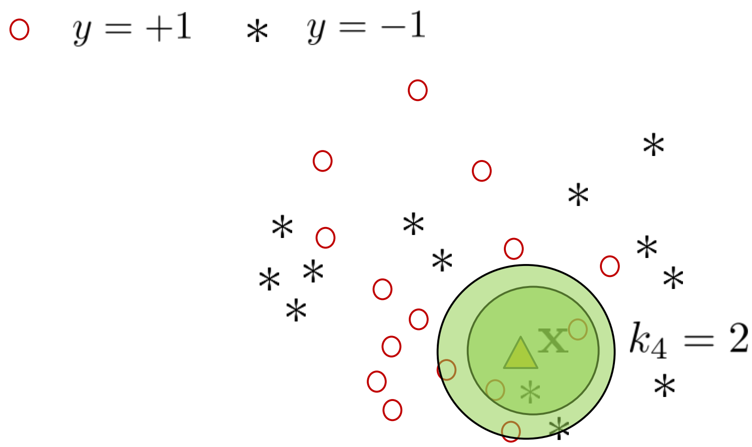
How to compute?
Strategy II

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

$\mathbf{x}$: 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.

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

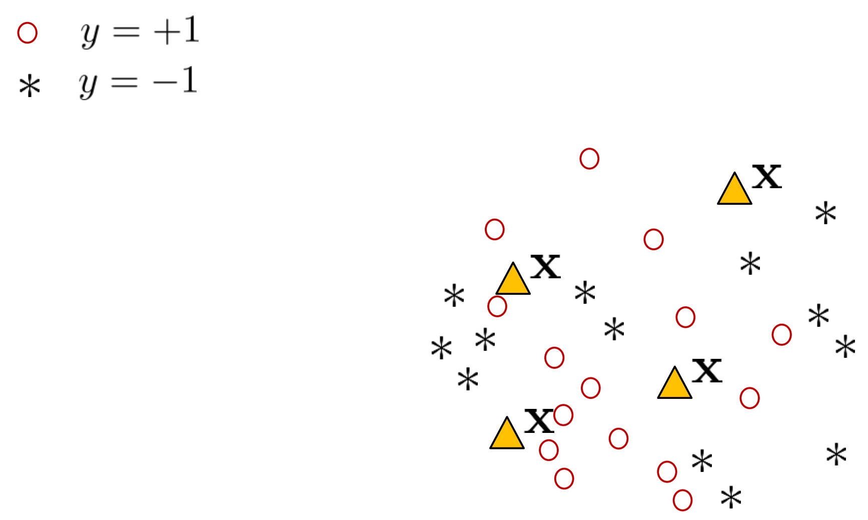
$$p_l(\mathbf{x}|y = -1) \cong \frac{k_l(\mathbf{x})}{l \times V_l(\mathbf{x})} \propto \frac{2}{4 \times 0.4} \propto \frac{2}{1.6}$$

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

The k-Nearest Neighbor Rule

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

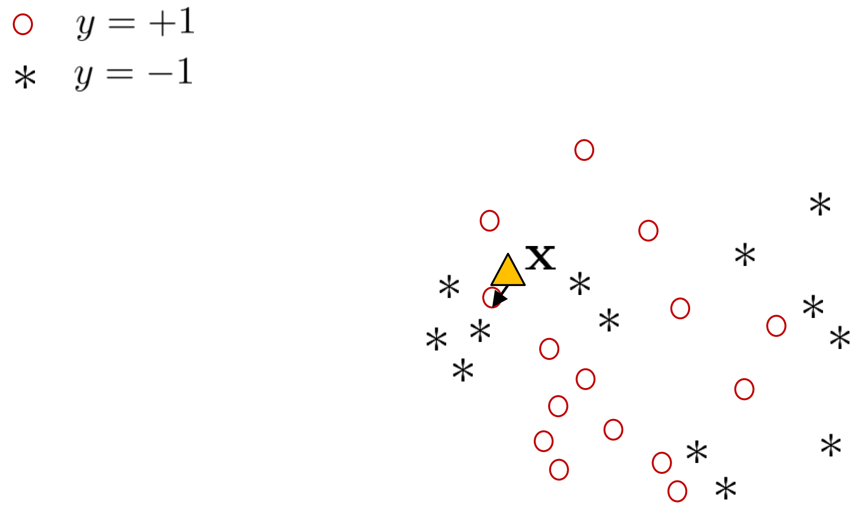
An extension of the nearest neighbor rule:
The k-nearest neighbor rule classifies $\mathbf{x}$ by assigning it the label most frequently represented among the k nearest samples In other words, given $\mathbf{x}$, we find the k nearest labeled samples. The label appeared most is assigned to $\mathbf{x}$.



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 $\mathbf{x}$.



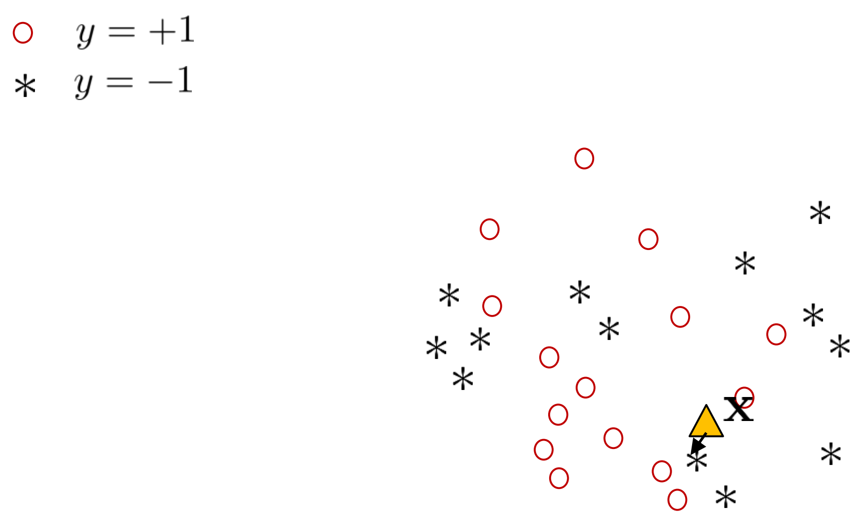
$k = 1$

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

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 $\mathbf{x}$.



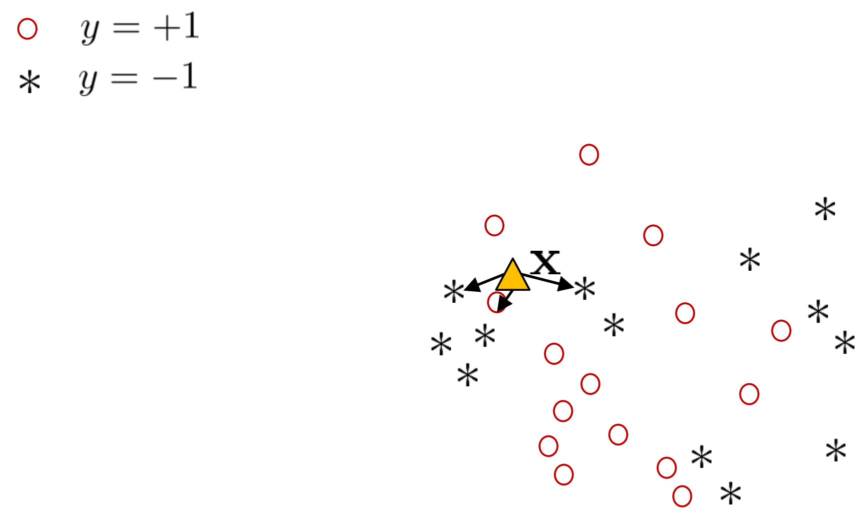
$k = 1$

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

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 $\mathbf{x}$.



$k = 3$

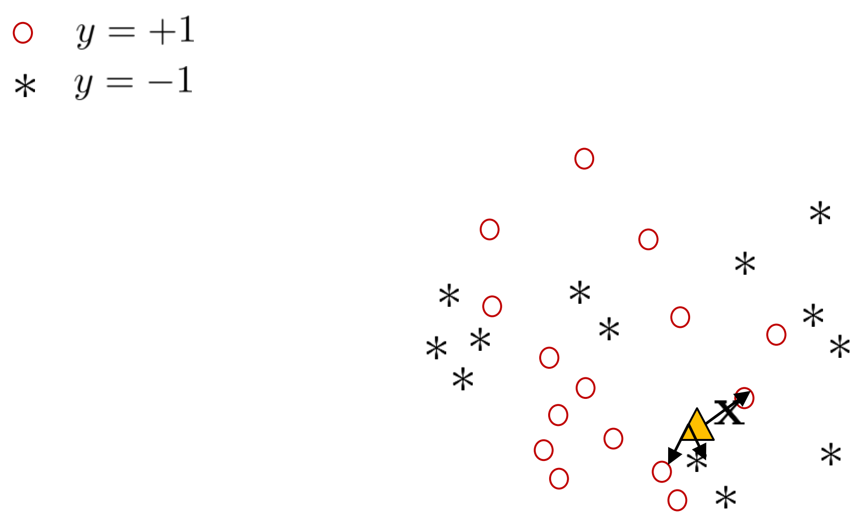
1 positives
2 negative

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

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 $\mathbf{x}$.



$k = 3$

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.