Naïve Bayes, Support Vector Machines

Danna Gurari

University of Texas at Austin

Spring 2021

https://www.ischool.utexas.edu/~dannag/Courses/IntroToMachineLearning/CourseContent.html

Review

- Last week:
  - Multiclass classification applications and evaluating models
  - Motivation for new ML era: need non-linear models
  - Nearest neighbor classification
  - Decision tree classification
  - Parametric versus non-parametric models

- Assignments (Canvas)
  - Problem set 3 due tonight
  - Problem set 4 out and due in two weeks (after spring break)
  - Lab assignment 2 out and due in three weeks

- Questions?

Today’s Topics

- Evaluating Machine Learning Models Using Cross-Validation
- Naïve Bayes
- Support Vector Machines

Today’s Topics

- Evaluating Machine Learning Models Using Cross-Validation
- Naïve Bayes
- Support Vector Machines

Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples

Input:				● ● ●		● ● ●
Label:	Hairy	Hairy	Not Hairy	● ● ●	Hairy	● ● ●

Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples

Classifier **predicts well** when test data **matches** training data. Lucky?

Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples

Classifier **predicts poorly** when test data **does not match** training data. Unlucky?

Goal: Design Models that **Generalize** Well to New, Previously Unseen Examples

How to know if good/bad evaluation scores happen from good/bad luck?

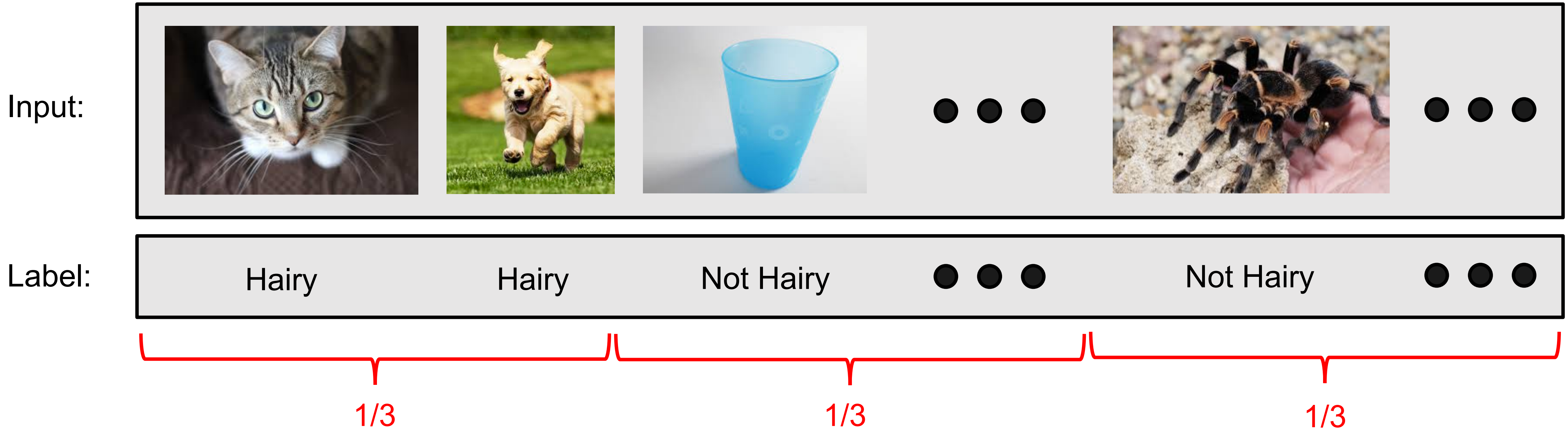
Evaluation of Classification Model

Cross-validation:

limit influence of chosen dataset split

Evaluation of Classification Model

e.g., 3-fold cross-validation



Cross-validation

Evaluation of Classification Model

e.g., 3-fold cross-validation

Fold 1:

– train on k–1 partitions

– test on k partitions

Fold 2:

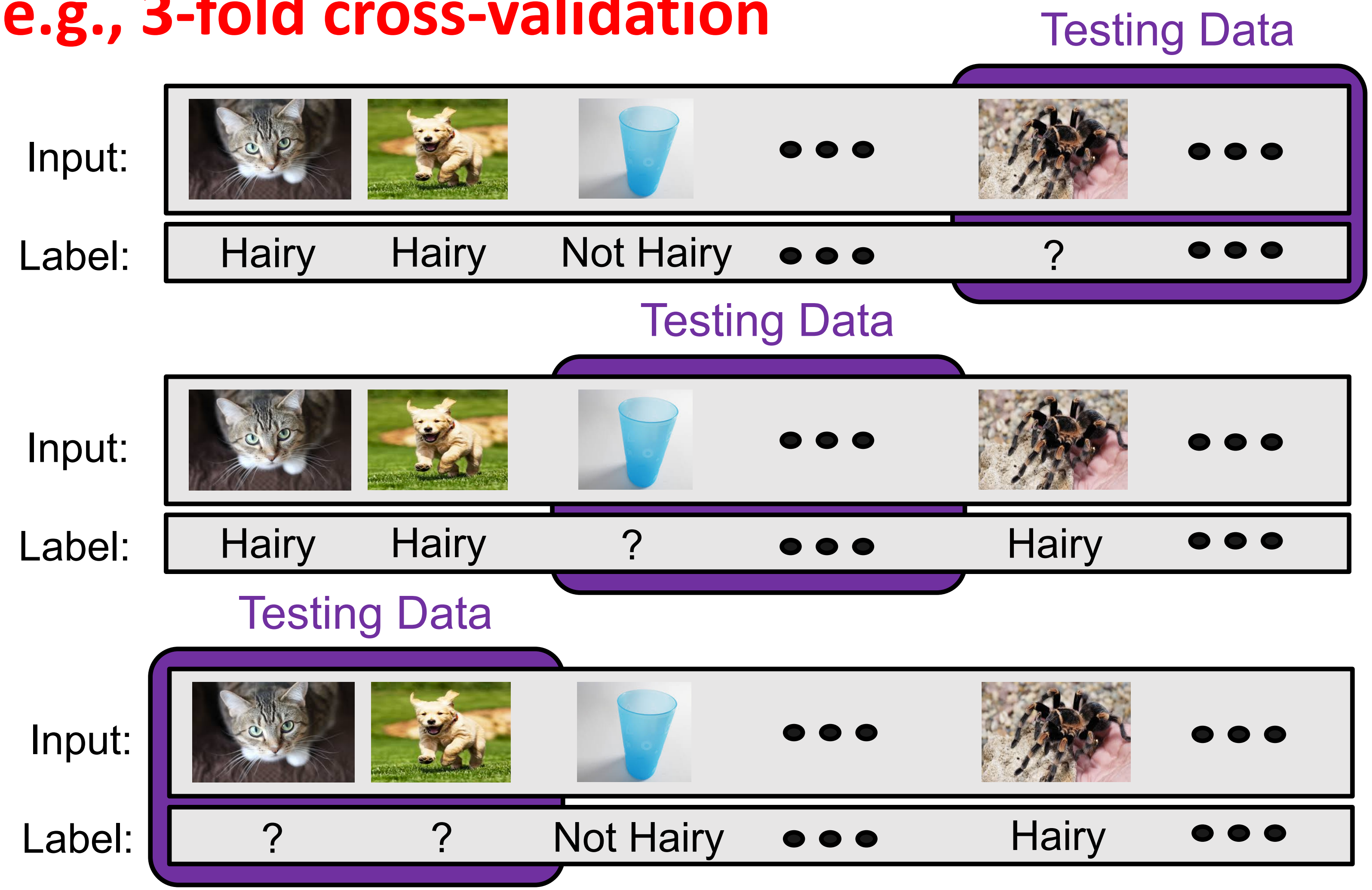
– train on k–1 partitions

– test on k partitions

Fold 3:

– train on k–1 partitions

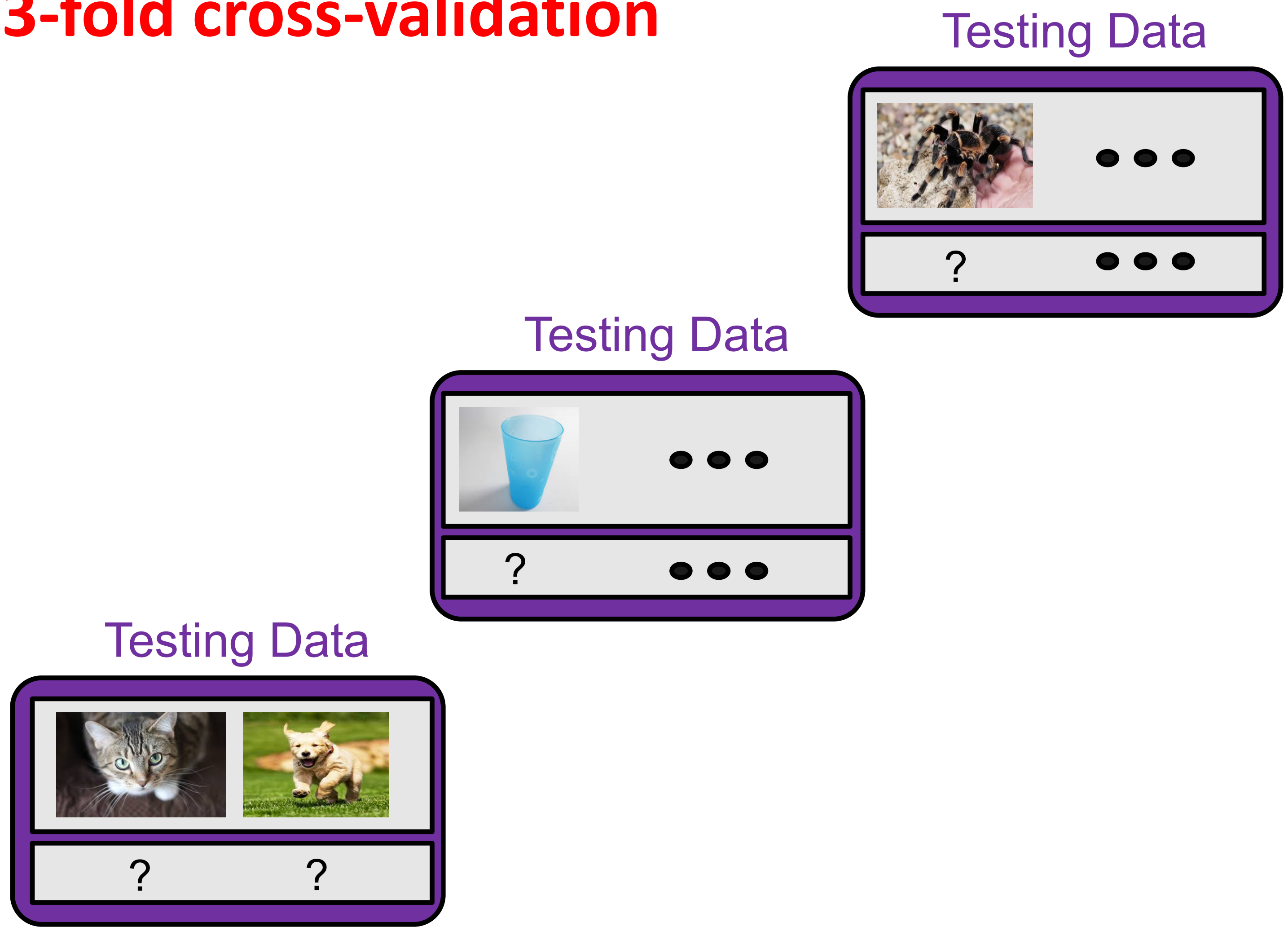
– test on k partitions



Evaluation of Classification Model

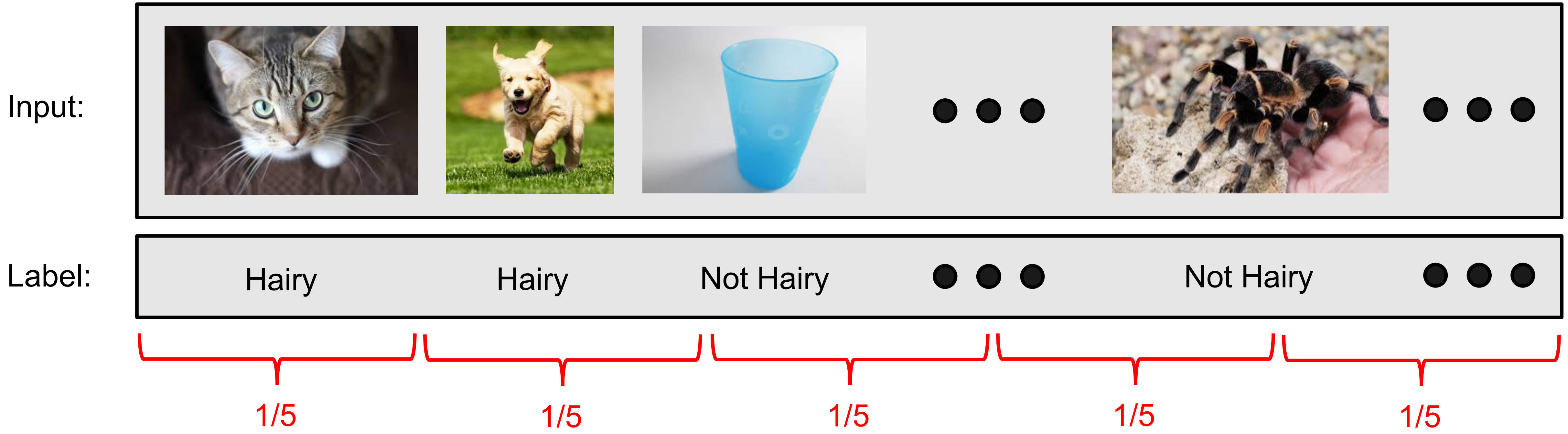
e.g., 3-fold cross-validation

Classifier accuracy: prediction accuracy across all folds of test data



Evaluation of Classification Model

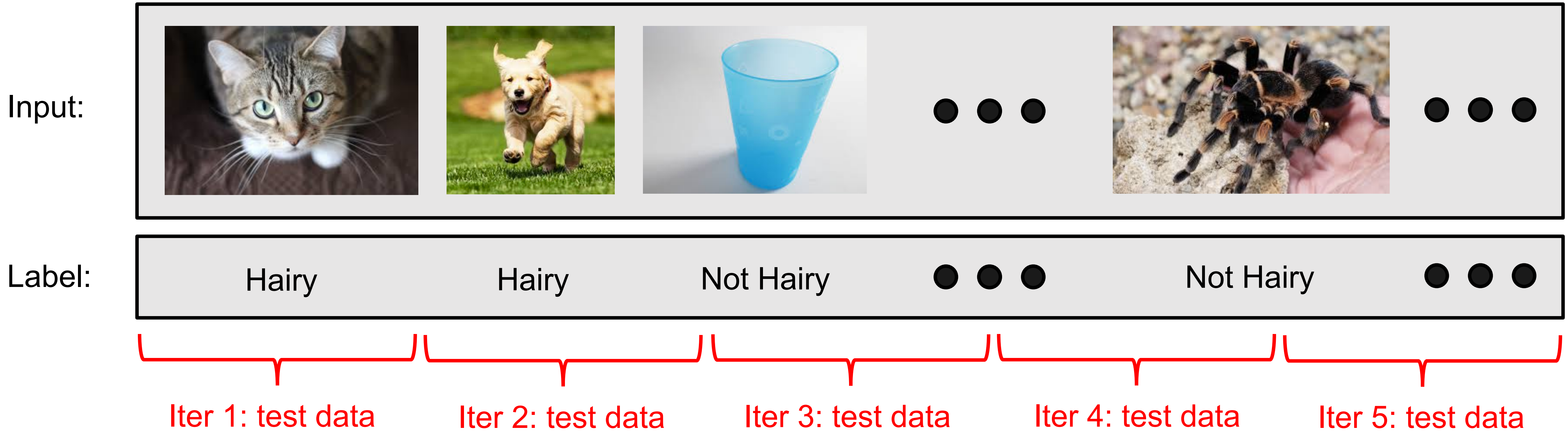
e.g., 5-fold cross-validation



How many partitions of the data to create?

Evaluation of Classification Model

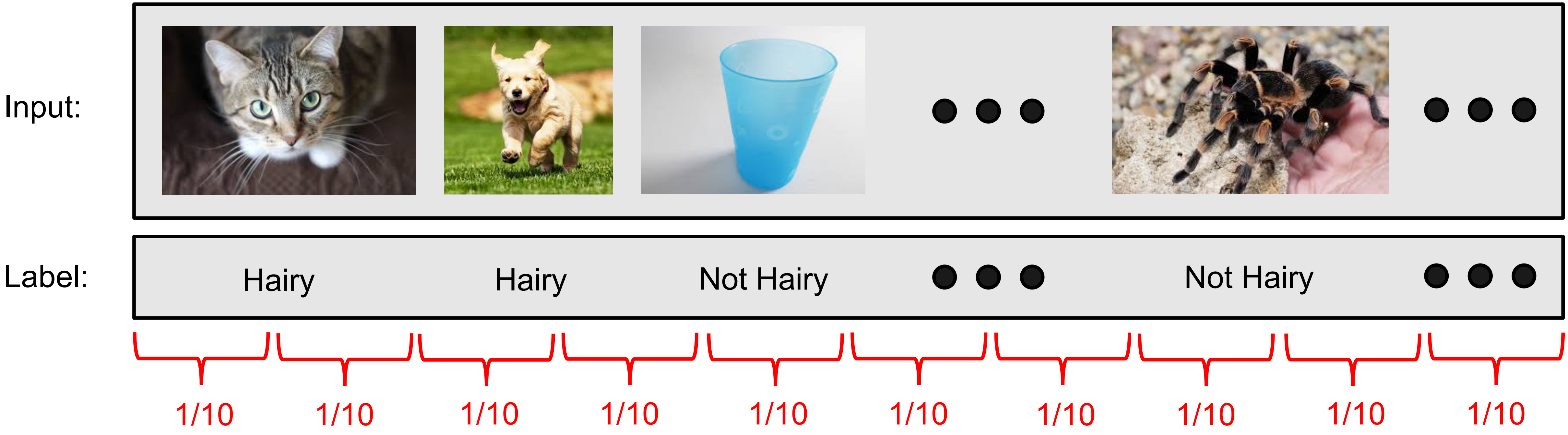
e.g., 5-fold cross-validation



How many iterations of train & test to run?

Evaluation of Classification Model

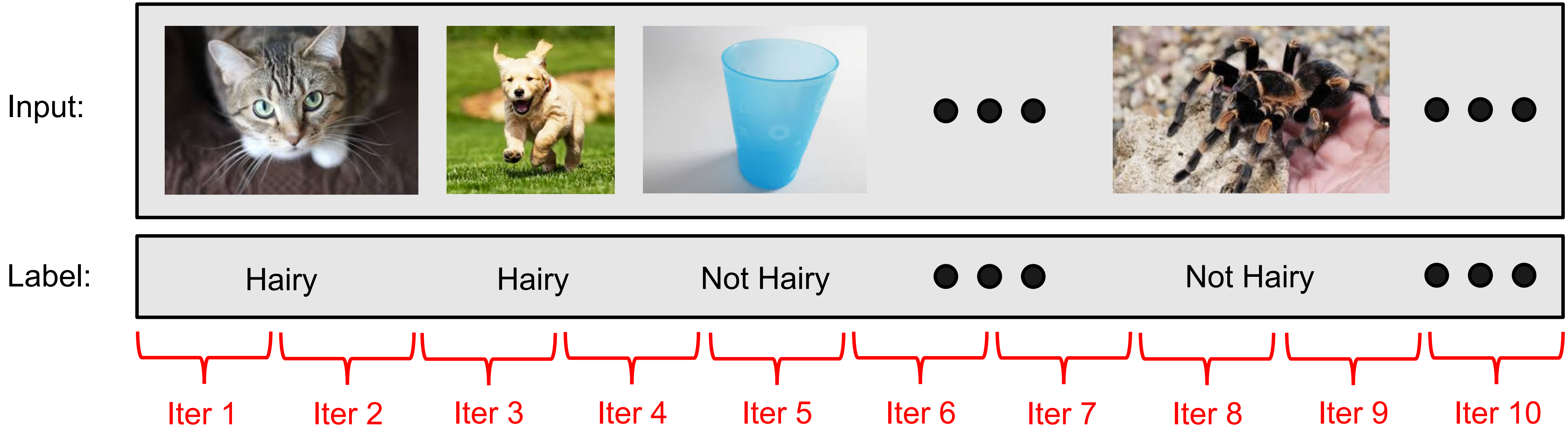
e.g., 10-fold cross-validation



How many partitions of the data to create?

Evaluation of Classification Model

e.g., 10-fold cross-validation



How many iterations of train & test to run?

Evaluation of Classification Model

e.g., k-fold cross-validation

Input:				● ● ●		● ● ●
Label:	Hairy	Hairy	Not Hairy	● ● ●	Not Hairy	● ● ●

What are the (dis)advantages of using larger values for “k”?

K-Fold Cross-Validation: How to Partition Data?

- e.g., 3-fold cross validation?

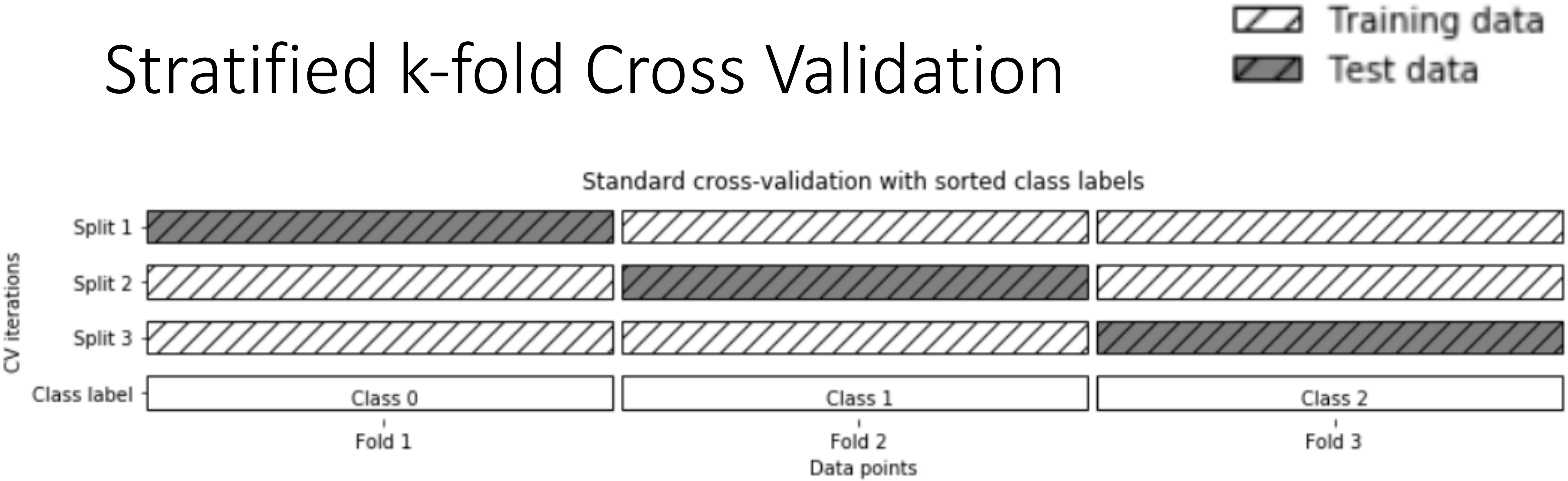
```
In [4]: iris.target

Out[4]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
               0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
               1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
               1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
               2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
               2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

Stratified k-fold Cross Validation

- e.g., 3-fold cross validation? Preserve class proportions in each fold to represent proportions in the whole dataset

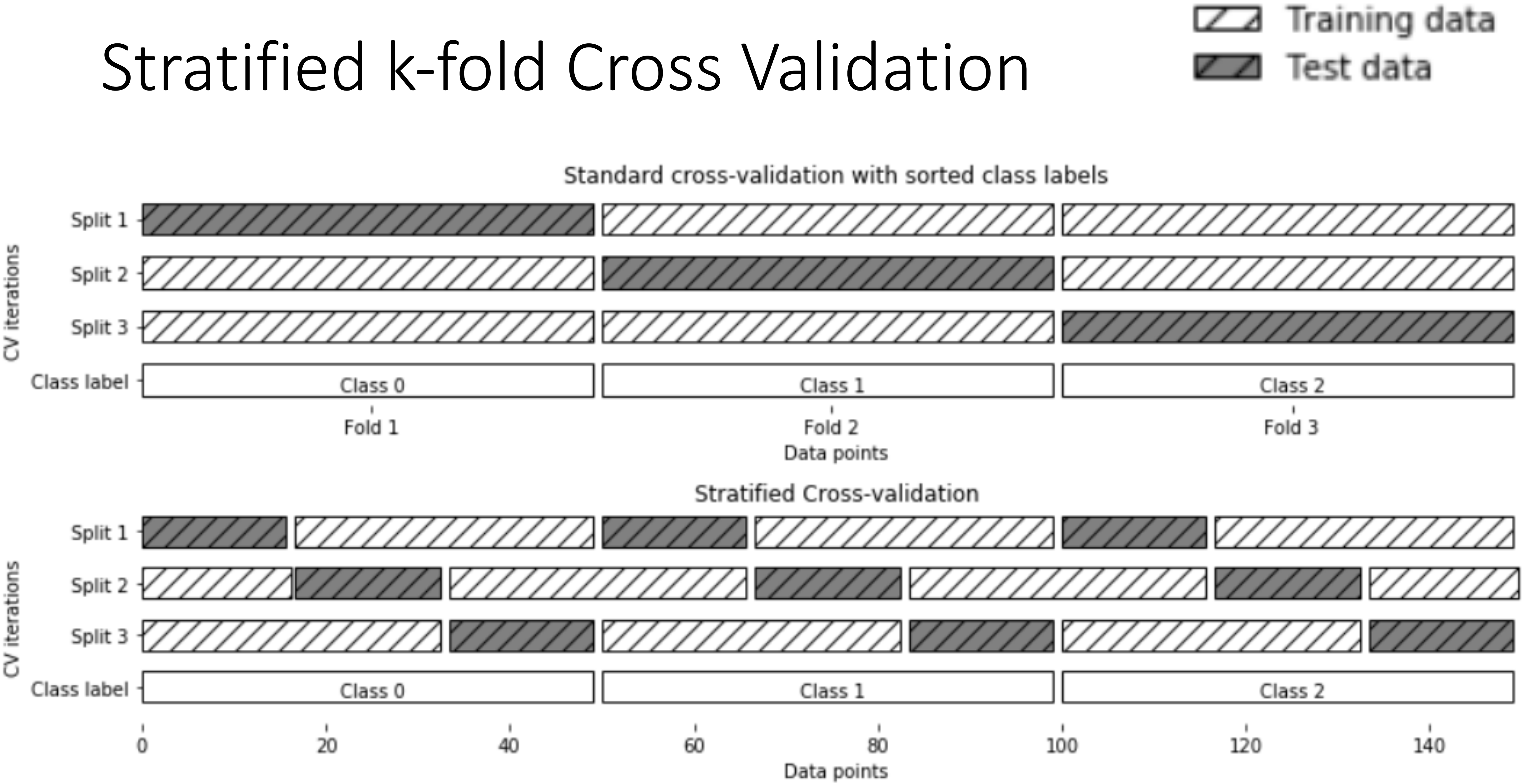
```
In [4]: iris.target

Out[4]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
               0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
               1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
               1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
               2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
               2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

Stratified k-fold Cross Validation



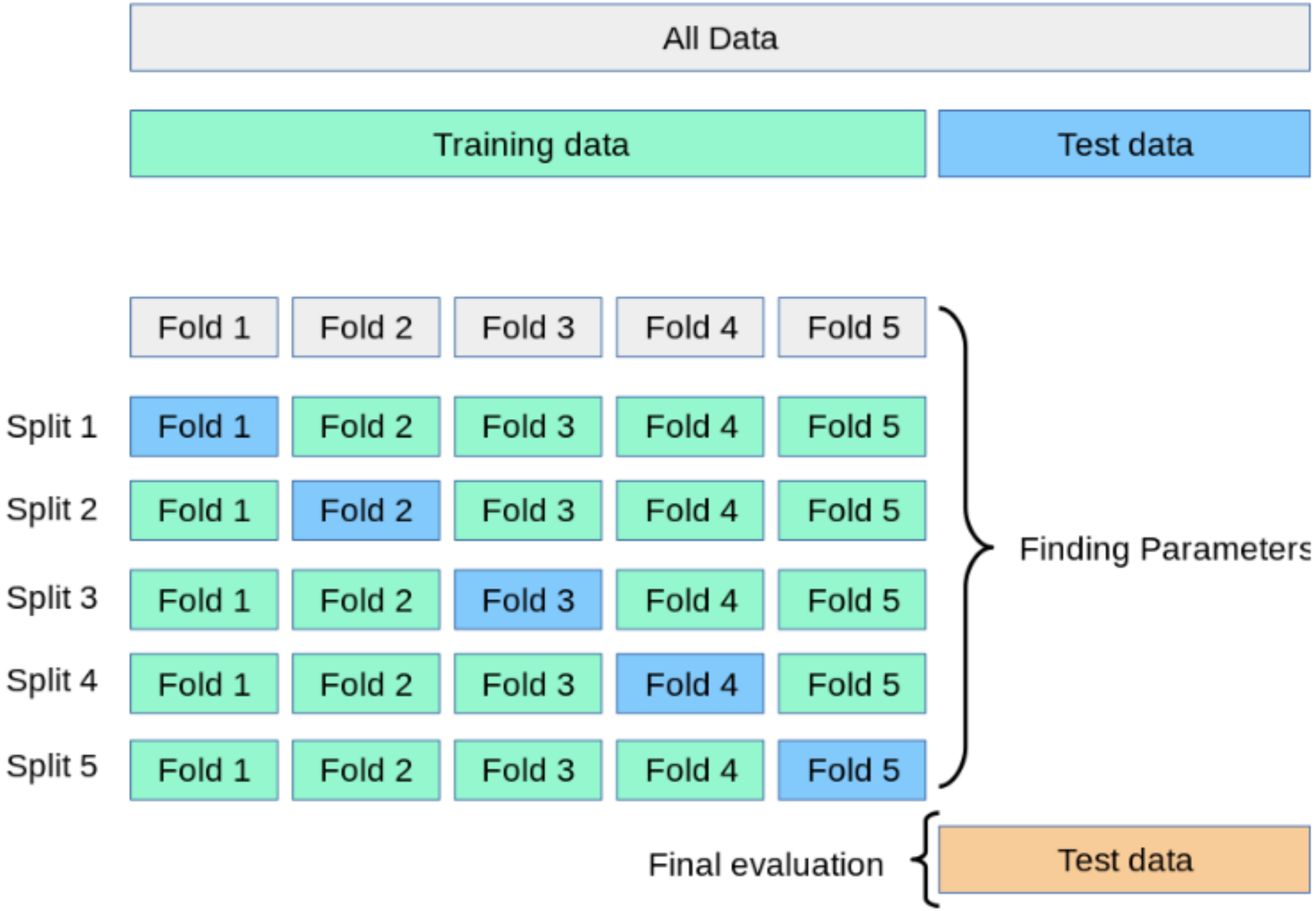
https://github.com/amueller/introduction_to_ml_with_python/blob/master/05-model-evaluation-and-improvement.ipynb

Stratified k-fold Cross Validation



https://github.com/amueller/introduction_to_ml_with_python/blob/master/05-model-evaluation-and-improvement.ipynb

Summary: K-Fold Cross Validation



https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

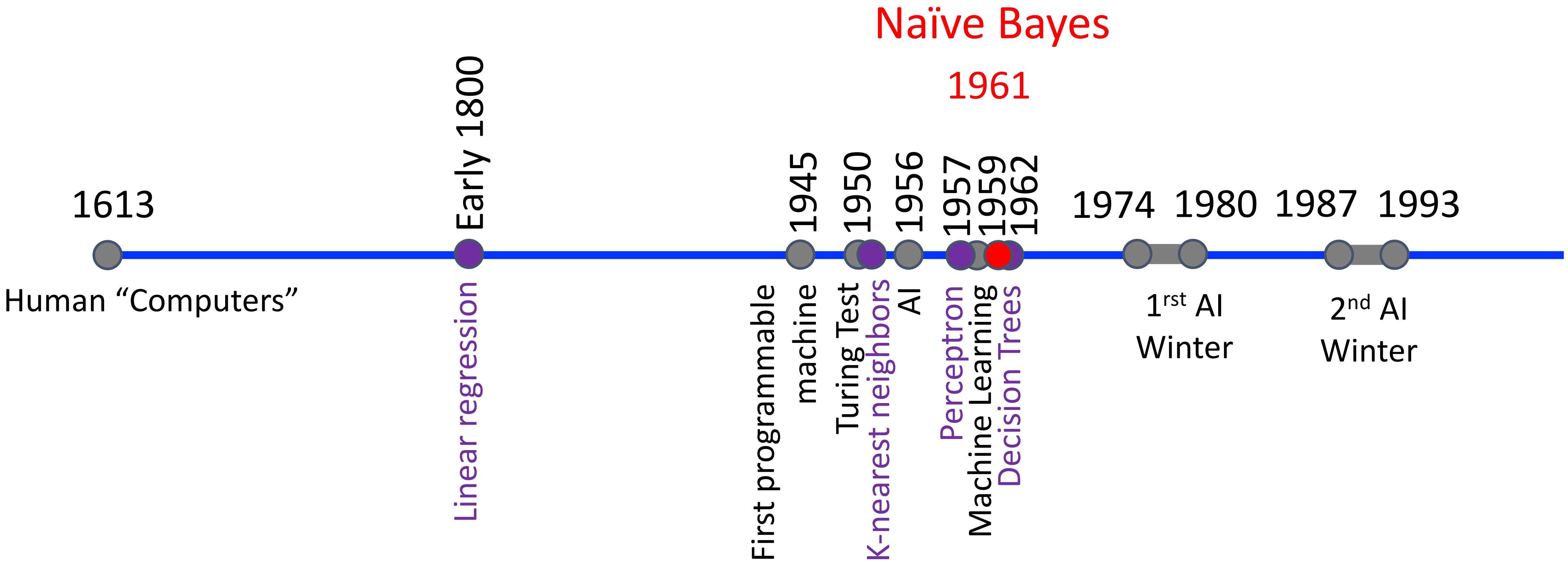
Group Discussion: Cross Validation

- Why would you choose cross validation over percentage split (train/val/test split)?
- Why would you choose percentage split (train/val/test split) over cross validation?
- What does high variance of test accuracy between different folds tell you?
- Does cross validation build a final model for use on new data?

Today’s Topics

- Evaluating Machine Learning Models Using Cross-Validation
- Naïve Bayes
- Support Vector Machines

Historical Context of ML Models



M. E. Maron. Automatic Indexing: An Experimental Inquiry. Journal of the ACM. 1961

Naïve Bayes

- Learns a model of the **joint probability** of the input features and each class, and then picks the most probable class

Background: Probability

- Joint probability: P(A, B)
  - i.e., probability of two events occurring simultaneously
- How to calculate joint probability?
- Can use **chain rule**: P(A, B) = P(B|A) * P(A)
  - P(A | B) is probability of an event occurring in the presence of a second event; i.e., conditional probability

Background: Probability Example

- Calculating joint probability using chain rule
  - P(A, B) = P(B|A) * P(A)
- e.g., Let:
  - A be the event of choosing from the blue bowl
  - B the event of choosing a Snickers
- P(B|A) = ?
- P(A, B) = 2/3 * 1/2 = 1/3

Naïve Bayes

- Learns a model of the **joint probability** of the input features and each class, and then **picks the most probable class**

Background: Bayes' Theorem

- Given chain rule:
    - P(A, B) = P(A|B) * P(B)
    - P(A, B) = P(B|A) * P(A)
- We can get:
    - P(A|B) * P(B) = P(B|A) * P(A)
- Rearranging:
    - P(A|B) = (P(B|A) * P(A))/P(B)
- Rewriting:

Need to solve this… more to follow

Need to solve this… more to follow

$$\boxed{P(C_i|features)} = \frac{\boxed{P(features|C_i)} * \boxed{P(C_i)}}{\boxed{P(features)}}$$

Want to find class with the largest probability

Constant for all classes… so can ignore this!

Naïve Bayes

- Learns a model of the **joint probability** of the **input features** and **each class**, and then **picks the most probable class**

Naïve Bayes: Naively Assumes Features Are Class Conditionally Independent

If we assume independence then

P(A,B)=P(A)P(B)

However, in many cases such an assumption maybe too strong (more later in the class)

$$P(C_i|features) = \boxed{P(features|C_i)} * P(C_i)$$

$$P(features|C_i) = \prod_{j=1}^{m} P(x_j|C_i)$$

$$P(features|C_i) = P(x_1|C_i) * P(x_2|C_i) * ... * P(x_m|C_i)$$

$$P(C_i|features) = \boxed{P(x_1|C_i) * P(x_2|C_i) * ... * P(x_m|C_i)} * P(C_i)$$

Naïve Bayes: Different Generative Models Can Yield the Observed Features

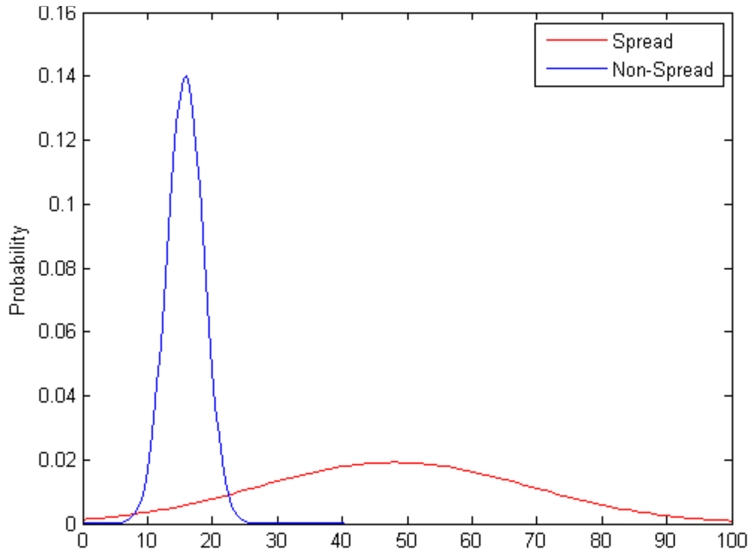
Key Decision: How to compute probability of each feature given the class?

Recall: Want to find class with the largest probability

$$\boxed{P(C_i|features)} = \boxed{P(x_1|C_i)} * \boxed{P(x_2|C_i)} * \ldots * \boxed{P(x_m|C_i)} * P(C_i)$$

Naïve Bayes: Different Generative Models Can Yield the Observed Features

- **Gaussian** Naïve Bayes (typically used for “continuous”-valued features)
  - Assume data drawn from a Gaussian distribution: mean + standard deviation



$$P(C_i|features) = \boxed{P(x_1|C_i) * P(x_2|C_i) * \ldots * P(x_m|C_i)} * P(C_i)$$

Naïve Bayes: Different Generative Models Can Yield the Observed Features

- **Multinomial** Naïve Bayes (typically used for "discrete"-valued features)
  - Assume count data and computes fraction of entries belonging to the category

e.g.,

Movie	Type	Length	Liked?
m1	Comedy	Short	Yes
m2	Drama	Medium	Yes
m3	Comedy	Medium	No
m4	Drama	Long	No
m5	Drama	Medium	Yes
m6	Drama	Short	No
m7	Comedy	Short	Yes
m8	Drama	Medium	Yes

$$P(C_i|features) = \boxed{P(x_1|C_i) * P(x_2|C_i) * \ldots * P(x_m|C_i)} * P(C_i)$$

Gaussian Naïve Bayes: Example

x_1

e.g.,

IMDb Rating	Liked?
7.2	Yes
9.3	Yes
5.1	No
6.9	No
8.3	Yes
4.5	No
8.0	Yes
7.5	Yes

- P(Liked) = ?
  - 5/8 = 0.625

$$P(C_i|features) = P(x_1|C_i) * \boxed{P(C_i)}$$

Gaussian Naïve Bayes: Example

e.g.,

x_1

IMDb Rating	Liked?
7.2	Yes
9.3	Yes
5.1	No
6.9	No
8.3	Yes
4.5	No
8.0	Yes
7.5	Yes

- P(Liked) = ?
  - 5/8 = 0.625
- P(Not Liked) = ?
  - 3/8 = 0.375

$$P(C_i|features) = P(x_1|C_i) * \boxed{P(C_i)}$$

Gaussian Naïve Bayes: Example

x_1

e.g.,

IMDb Rating	Liked?
7.2	Yes
9.3	Yes
5.1	No
6.9	No
8.3	Yes
4.5	No
8.0	Yes
7.5	Yes

- P(Liked) = 5/8 = 0.625
- P(Not Liked) = 3/8 = 0.375
- P(IMDb Rating | Liked): Mean and Standard Deviation?
  - Mean = 8.06
  - Standard Deviation = 0.81

$$P(C_i|features) = \boxed{P(x_1|C_i)} * P(C_i)$$

Gaussian Naïve Bayes: Example

e.g.,

x_1

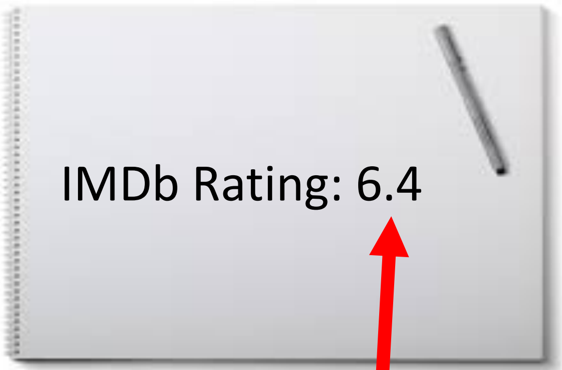
IMDb Rating	Liked?
7.2	Yes
9.3	Yes
5.1	No
6.9	No
8.3	Yes
4.5	No
8.0	Yes
7.5	Yes

- P(Liked) = 5/8 = 0.625
- P(Not Liked) = 3/8 = 0.375
- P(IMDb Rating | Liked)
  - Mean = 8.06
  - Standard Deviation = 0.81
- P(IMDb Rating | Not Liked): Mean and Standard Deviation?
  - Mean = 5.5
  - Standard Deviation = 1.25

$$P(C_i|features) = \boxed{P(x_1|C_i)} * P(C_i)$$

Gaussian Naïve Bayes: Example

Test Example



- P(Liked | Features)

e.g.,

x_1

IMDb Rating	Liked?
7.2	Yes
9.3	Yes
5.1	No
6.9	No
8.3	Yes
4.5	No
8.0	Yes
7.5	Yes

- P(Liked) = 5/8 = 0.625
- P(Not Liked) = 3/8 = 0.375
- P(IMDb Rating | Liked)
  - Mean = 8.06
  - Standard Deviation = 0.81
- P(IMDb Rating | Not Liked)
  - Mean = 5.5
  - Standard Deviation = 1.25

$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

(Can Use: https://planetcalc.com/4986/)

$$P(C_i|features) = \boxed{P(x_1|C_i)} * P(C_i)$$

Gaussian Naïve Bayes: Example

e.g.,

x_1

IMDb Rating	Liked?
7.2	Yes
9.3	Yes
5.1	No
6.9	No
8.3	Yes
4.5	No
8.0	Yes
7.5	Yes

- P(Liked) = 5/8 = 0.625
- P(Not Liked) = 3/8 = 0.375
- P(IMDb Rating | Liked)
  - Mean = 8.06
  - Standard Deviation = 0.81
- P(IMDb Rating | Not Liked)
  - Mean = 5.5
  - Standard Deviation = 1.25

Test Example

- P(Liked | Features)
  - = 0.06 * 0.625

$$P(C_i|features) = P(x_1|C_i) * P(C_i)$$

Gaussian Naïve Bayes: Example

e.g.,

x_1

IMDb Rating	Liked?
7.2	Yes
9.3	Yes
5.1	No
6.9	No
8.3	Yes
4.5	No
8.0	Yes
7.5	Yes

- P(Liked) = 5/8 = 0.625
- P(Not Liked) = 3/8 = 0.375
- P(IMDb Rating | Liked)
  - Mean = 8.06
  - Standard Deviation = 0.81
- P(IMDb Rating | Not Liked)
  - Mean = 5.5
  - Standard Deviation = 1.25

Test Example

- P(Liked | Features)
  - = 0.06 * 0.625
  - = 0.0375
- P(Not Liked | Features)
  - = 0.25 * 0.375
  - = 0.09

Which class is the most probable?

$$P(C_i|features) = P(x_1|C_i) * P(C_i)$$

Multinomial Naïve Bayes: Example

	x_1	x_2	
Movie	Type	Length	Liked?
m1	Comedy	Short	Yes
m2	Drama	Medium	Yes
m3	Comedy	Medium	No
m4	Drama	Long	No
m5	Drama	Medium	Yes
m6	Drama	Short	No
m7	Comedy	Short	Yes
m8	Drama	Medium	Yes

- P(Liked) = 5/8 = 0.625
- P(Not Liked) = 3/8 = 0.375
- P(Comedy | Liked) = ?
  - 2/5 = 0.4
- P(Comedy | Not Liked) = ?
  - 1/3 = 0.333
- P(Drama | Liked) = ?
  - 3/5 = 0.6
- P(Drama | Not Liked) =
  - 2/3 = 0.666

$$P(C_i|features) = \boxed{P(x_1|C_i)} * P(x_2|C_i) * P(C_i)$$

Multinomial Naïve Bayes: Example

	x_1	x_2	
Movie	Type	Length	Liked?
m1	Comedy	Short	Yes
m2	Drama	Medium	Yes
m3	Comedy	Medium	No
m4	Drama	Long	No
m5	Drama	Medium	Yes
m6	Drama	Short	No
m7	Comedy	Short	Yes
m8	Drama	Medium	Yes

- P(Short | Liked) = ?
  - 2/5 = 0.4
- P(Short | Not Liked) = ?
  - 1/3 = 0.333
- P(Medium | Liked) = ?
  - 3/5 = 0.6
- P(Medium | Not Liked) = ?
  - 1/3 = 0.333
- P(Long | Liked) = ?
  - 0/5 = 0
- P(Long | Not Liked) = ?
  - 1/3 = 0.333

$$P(C_i|features) = P(x_1|C_i) * \boxed{P(x_2|C_i)} * P(C_i)$$

Multinomial Naïve Bayes: Example

Test Example

Which class is the most probable?

	x_1	x_2	
Movie	Type	Length	Liked?
m1	Comedy	Short	Yes
m2	Drama	Medium	Yes
m3	Comedy	Medium	No
m4	Drama	Long	No
m5	Drama	Medium	Yes
m6	Drama	Short	No
m7	Comedy	Short	Yes
m8	Drama	Medium	Yes

- P(Liked) = 0.63
- P(Not Liked) = 0.38
- P(Comedy | Liked) = 0.4
- P(Comedy | Not Liked) = 0.33
- P(Drama | Liked) = 0.6
- P(Drama | Not Liked) = 0.67
- P(Short | Liked) = 0.4
- P(Short | Not Liked) = 0.33
- P(Medium | Liked) = 0.6
- P(Medium | Not Liked) = 0.33
- P(Long | Liked) = 0
- P(Long | Not Liked) = 0.33

$$P(C_i|features) = P(x_1|C_i) * P(x_2|C_i) * P(C_i)$$

P(Liked | Features) = 0.4 x 0.6 x 0.63 = 0.15

P(Not Liked | Features) = 0.33 x 0.33 x 0.38 = 0.04

Test Example

Type: Comedy
Length: Long

Multinomial Naïve Bayes: Example

	x_1	x_2	
Movie	Type	Length	Liked?
m1	Comedy	Short	Yes
m2	Drama	Medium	Yes
m3	Comedy	Medium	No
m4	Drama	Long	No
m5	Drama	Medium	Yes
m6	Drama	Short	No
m7	Comedy	Short	Yes
m8	Drama	Medium	Yes

Which class is the most probable?

- P(Liked) = 0.63
- P(Not Liked) = 0.38
- P(Comedy | Liked) = 0.4
- P(Comedy | Not Liked) = 0.33
- P(Drama | Liked) = 0.6
- P(Drama | Not Liked) = 0.67
- P(Short | Liked) = 0.4
- P(Short | Not Liked) = 0.33
- P(Medium | Liked) = 0.6
- P(Medium | Not Liked) = 0.33
- P(Long | Liked) = 0
- P(Long | Not Liked) = 0.33

To avoid zero, assume training data is so large that adding one to each count makes a negligible difference

$$P(C_i|features) = P(x_1|C_i) * P(x_2|C_i) * P(C_i)$$

What are Naïve Bayes' Strengths

- Training is relatively fast
- Once trained, prediction is fast
- Can work well in the absence of a large dataset
- Requires little memory (a few computed statistics)

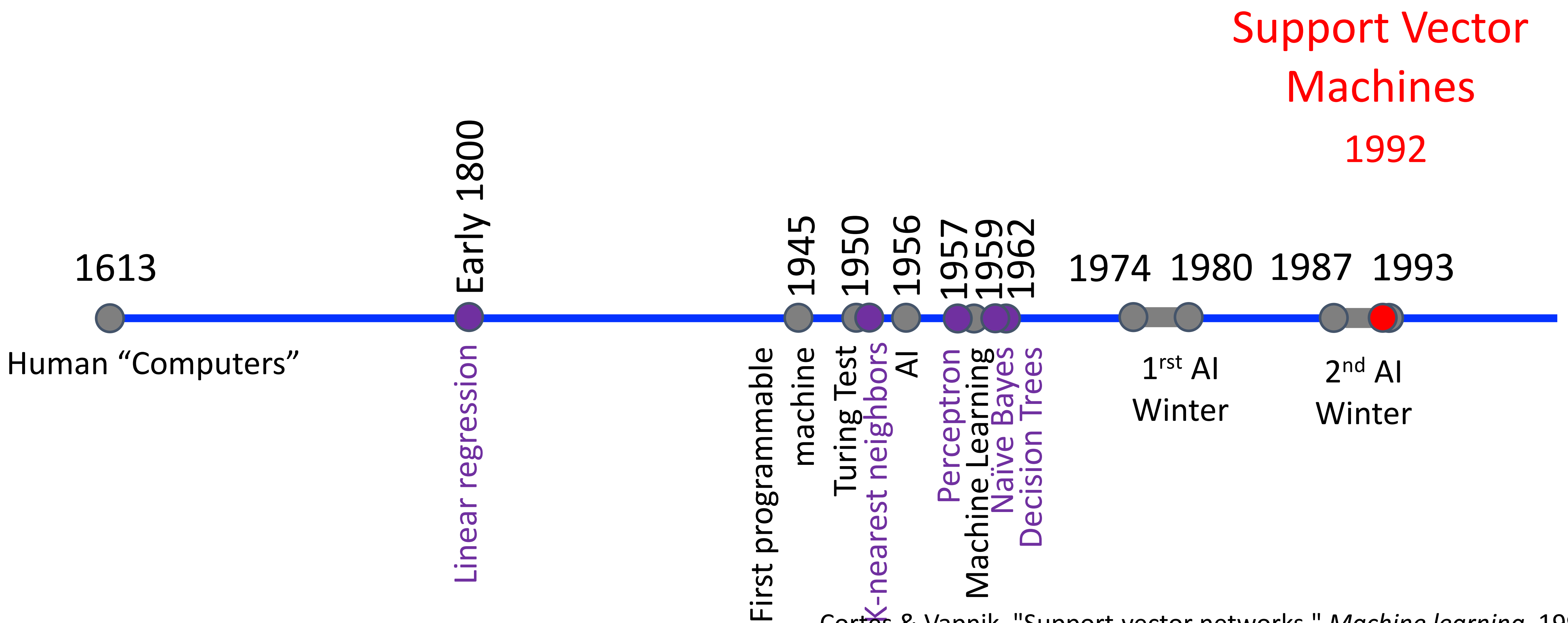
What are Naïve Bayes' Weaknesses

- Makes a strong, often unrealistic assumption that the presence of a each feature is completely unrelated to the presence of other features

Today’s Topics

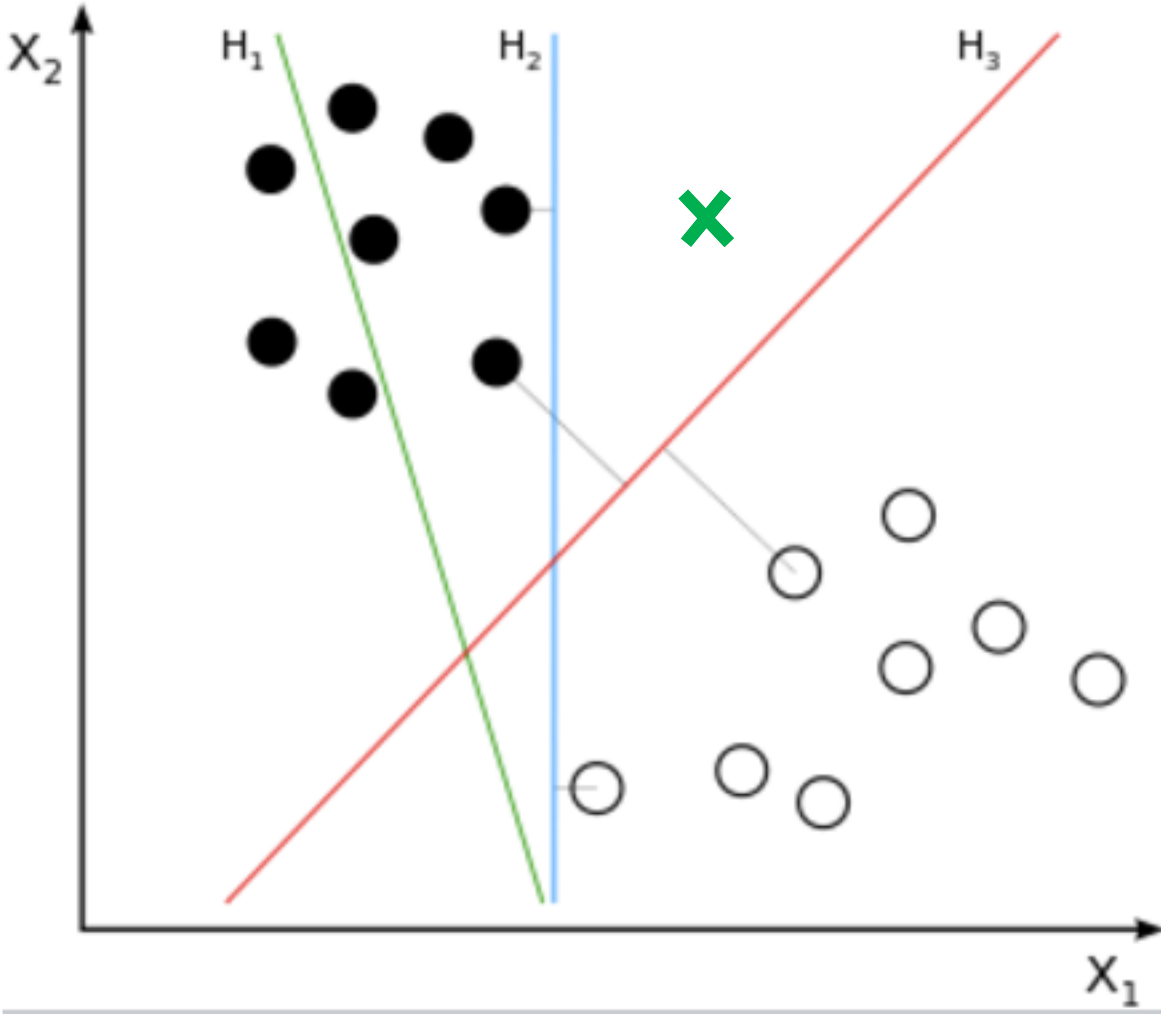
- Evaluating Machine Learning Models Using Cross-Validation
- Naïve Bayes
- Support Vector Machines

Historical Context of ML Models



Cortes & Vapnik. "Support-vector networks." *Machine learning,* 1995.
Boser, Guyon, & Vapnik. "A training algorithm for optimal margin classifiers." *Workshop on Computational learning theory,* 1992.

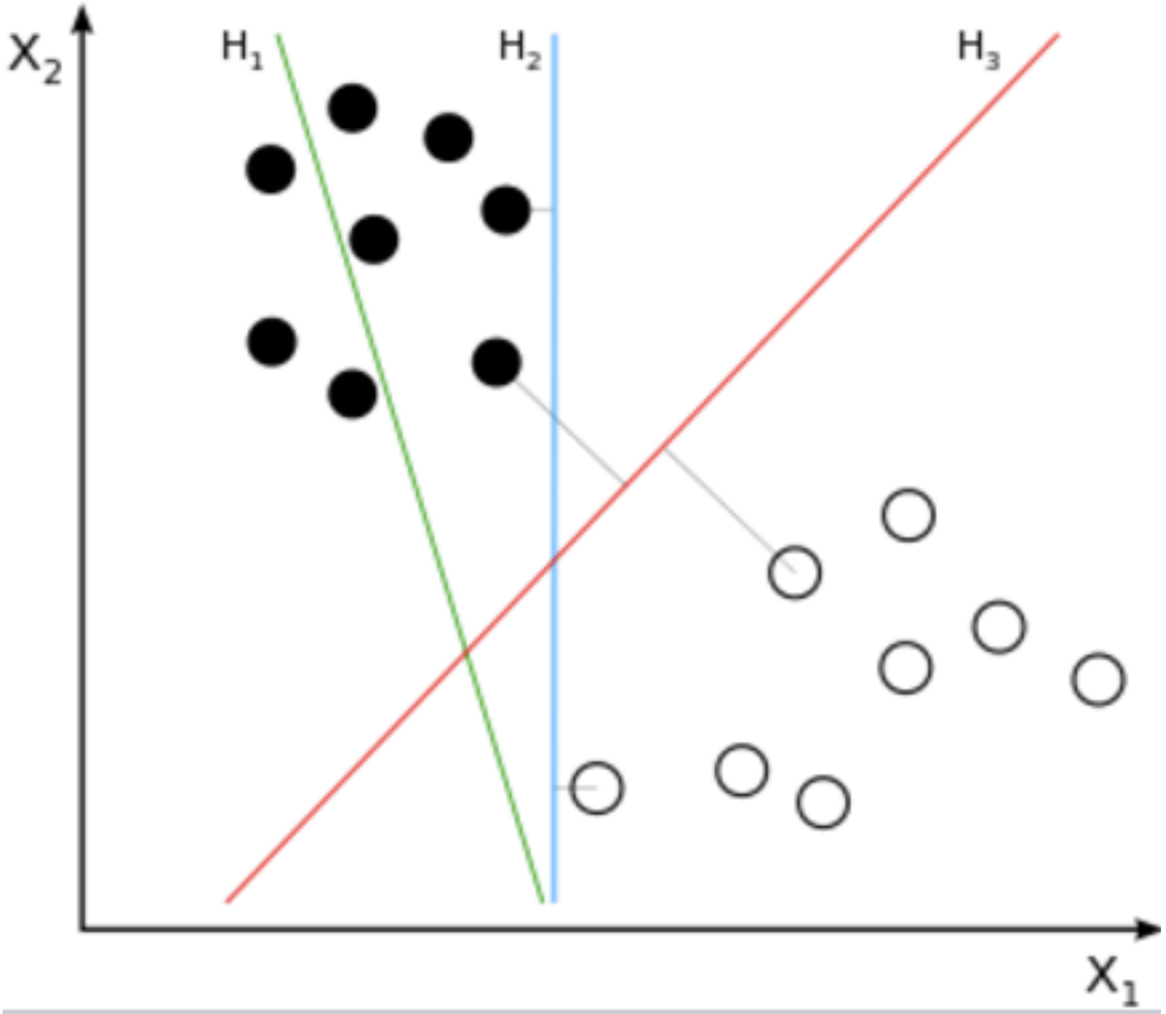
Support Vector Machine (SVM) Motivation



https://en.wikipedia.org/wiki/Linear_separability

To which class would each hyperplane (decision boundary) assign the new data point?

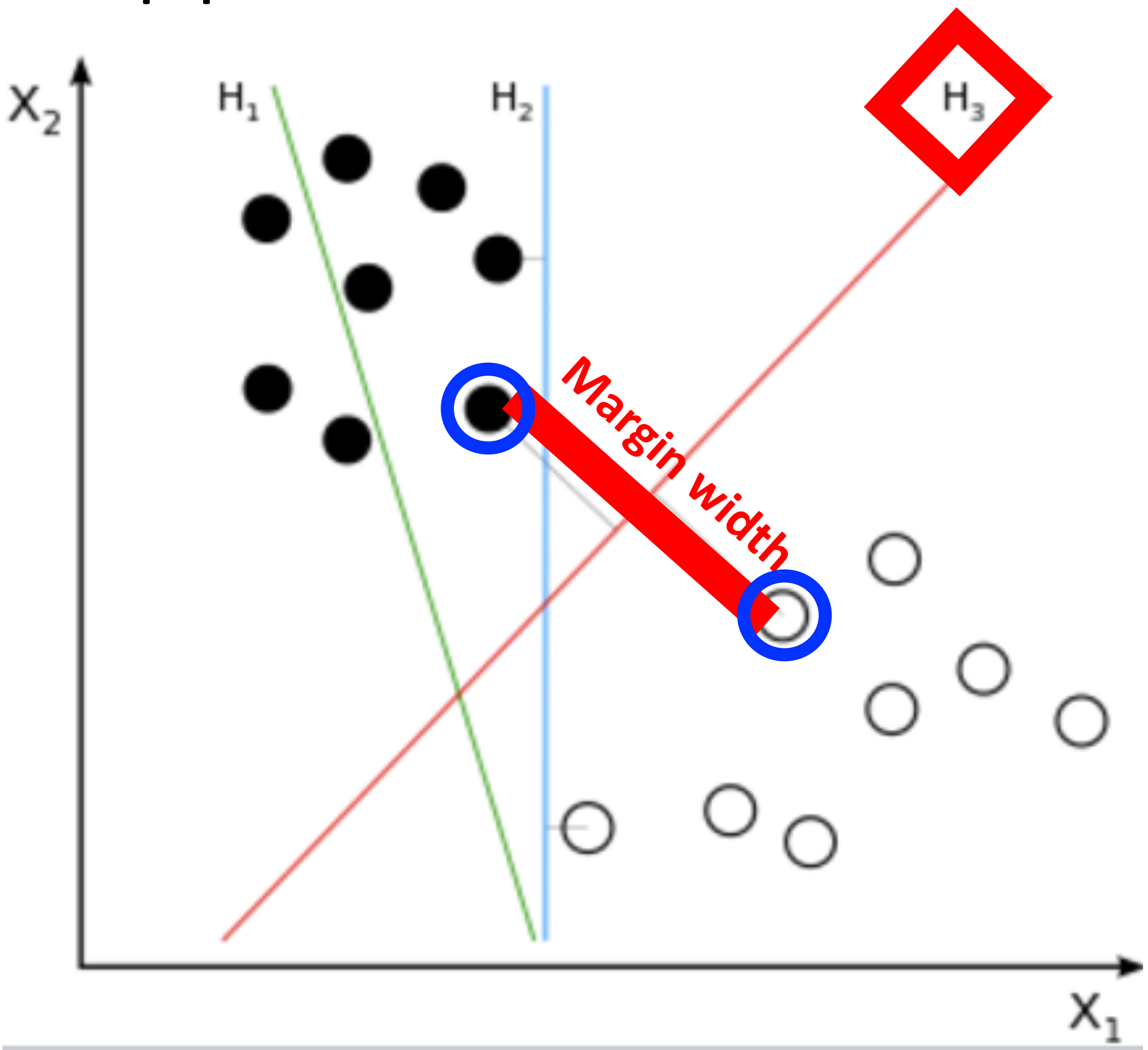
Support Vector Machine (SVM) Motivation



https://en.wikipedia.org/wiki/Linear_separability

Which hyperplane would you choose to separate data?

Support Vector Machine (SVM) Motivation

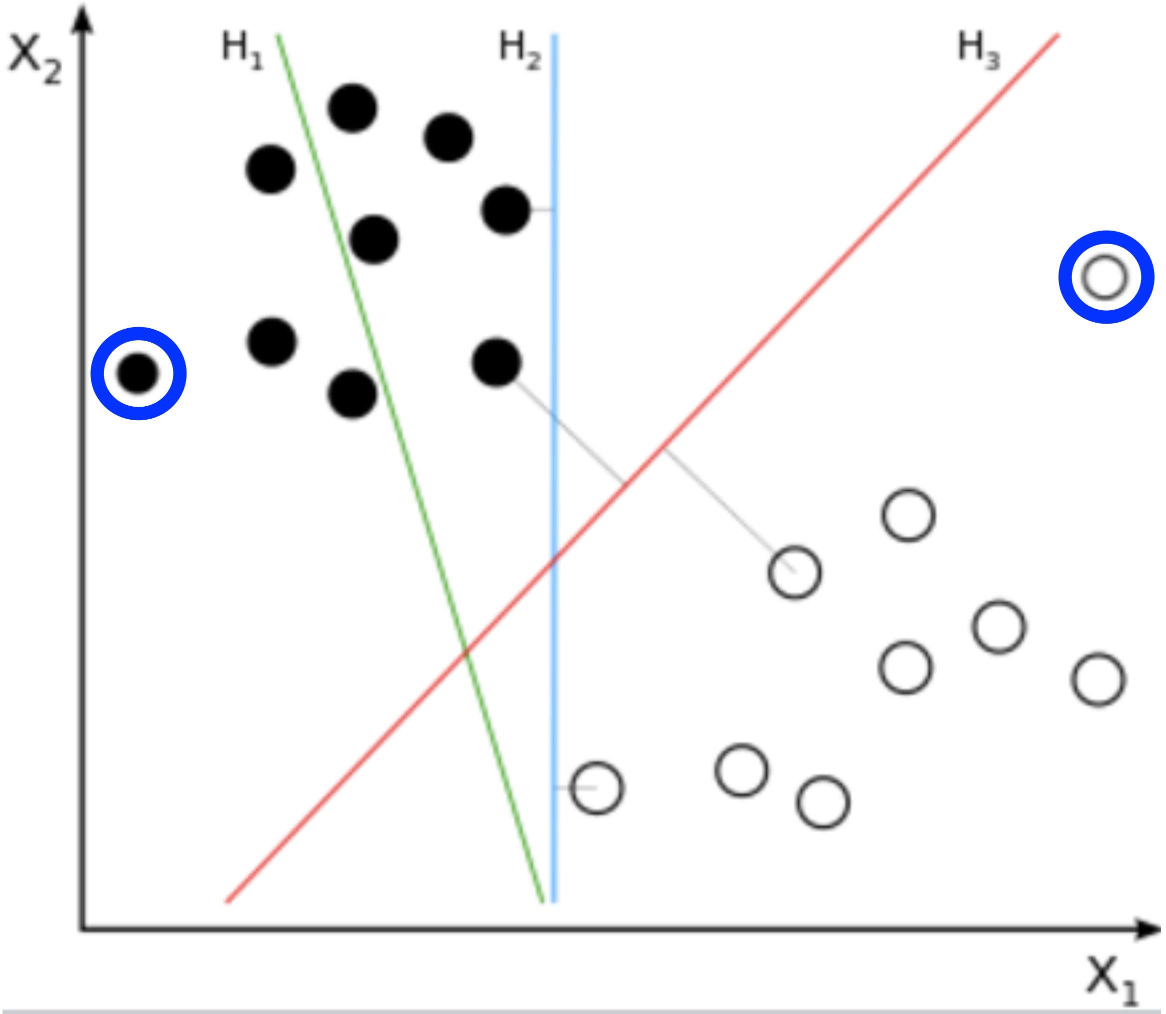


https://en.wikipedia.org/wiki/Linear_separability

Idea: choose hyperplane that maximizes the "margin" width.

Margin: distance between the separating hyperplane and training samples ("**support vectors**") closest to the hyperplane.

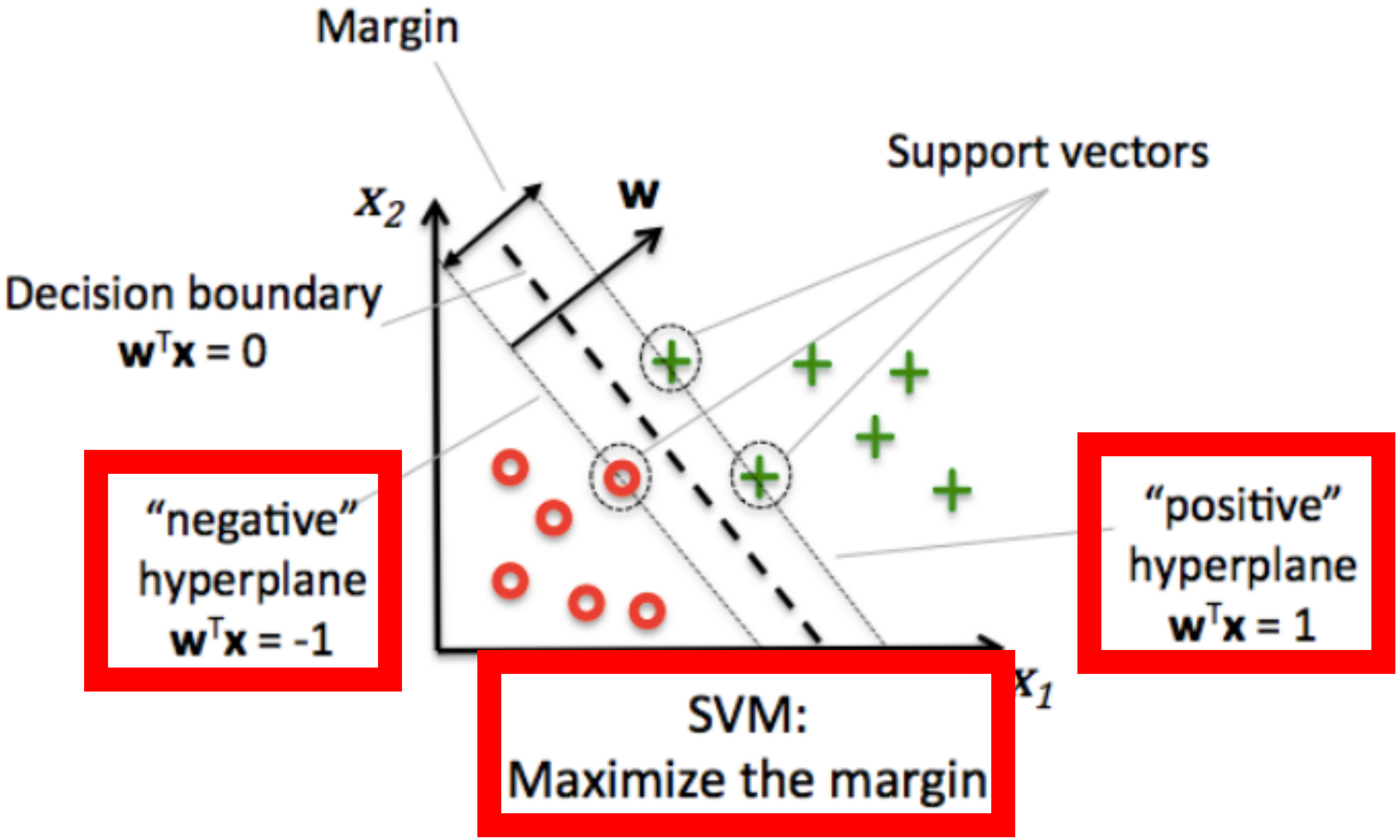
Support Vector Machine (SVM) Motivation



https://en.wikipedia.org/wiki/Linear_separability

When trying to maximize the margin, what happens to the choice of line when you add outliers to the dataset?

Support Vector Machine (SVM): Formalizing Definition



$$w_0x_0 + w_1x_1 + \cdots + w_mx_m = \sum_{j=0}^{m} \boldsymbol{x}_j \boldsymbol{w}_j = \boldsymbol{w}^T \boldsymbol{x}$$

Distance Between Parallel Lines Tutorial: http://web.mit.edu/zoya/www/SVM.pdf

<u>Derivation of Margin Length</u>

- Subtract two equations from each other:

$$\boldsymbol{w}^T \left(\boldsymbol{x}_{pos} - \boldsymbol{x}_{neg} \right) = 2$$

- Normalize by length of **w** to compute margin length:

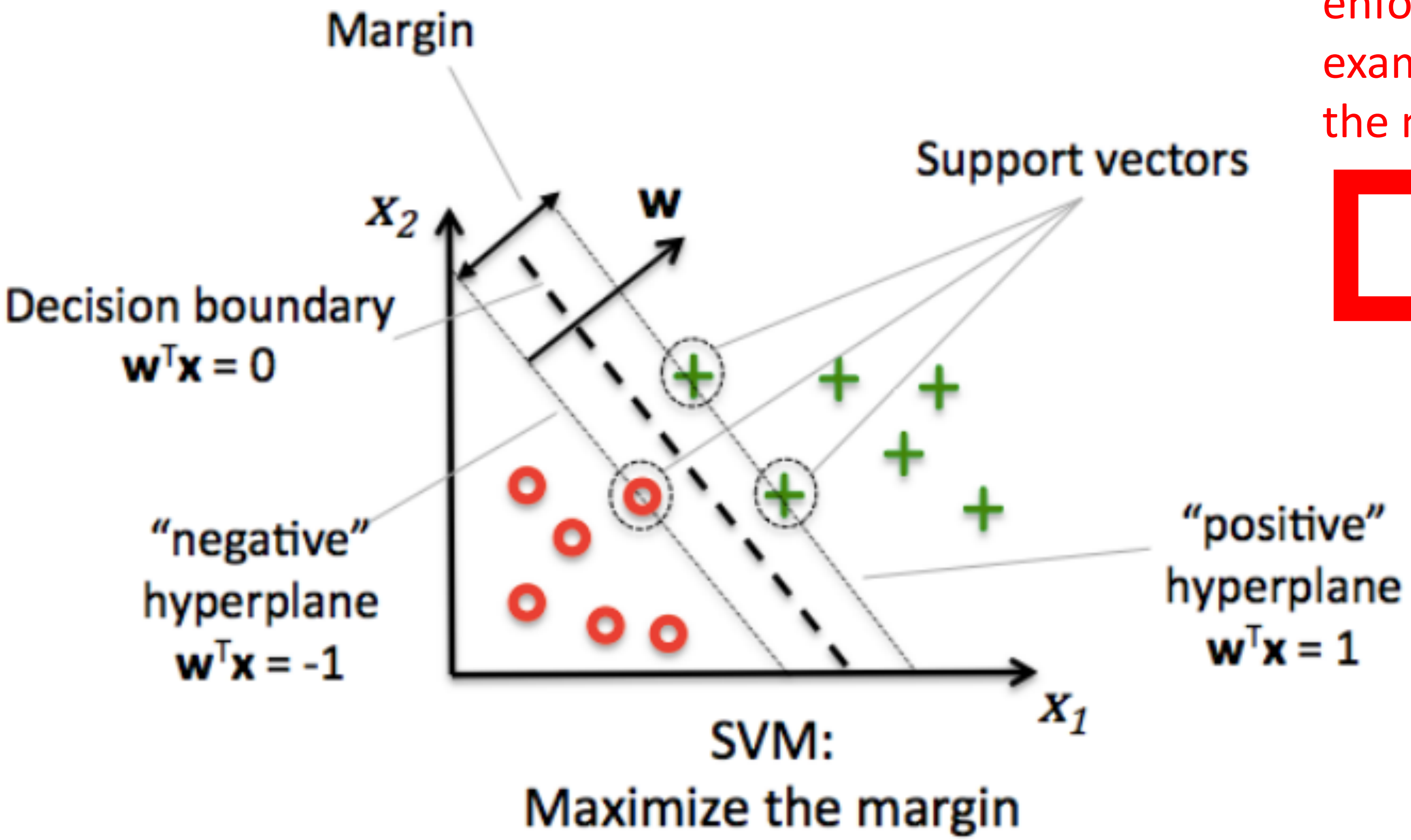
$$\frac{\boldsymbol{w}^T \left(\boldsymbol{x}_{pos} - \boldsymbol{x}_{neg} \right)}{\|\boldsymbol{w}\|} = \frac{2}{\|\boldsymbol{w}\|}$$

where:

$$\|\boldsymbol{w}\| = \sqrt{\sum_{j=1}^{m} w_j^2}$$

Python Machine Learning; Raschkka & Mirjalili

Support Vector Machine (SVM): Formalizing Definition



Same as finding parameters (w, w_0) that maximizes margin:

$$\min_{\mathbf{w}, w_0} \frac{1}{2} ||\mathbf{w}||^2$$

Constraint that enforces that all examples fall outside of the margin space:

$$s.t. \forall i \quad y^{(i)}\left(w_0 + \boldsymbol{w}^T \boldsymbol{x}^{(i)}\right) \geq 1,$$

$$w_0 + \boldsymbol{w}^T \boldsymbol{x}^{(i)} \geq 1 \ if \ y^{(i)} = 1$$

$$w_0 + \boldsymbol{w}^T \boldsymbol{x}^{(i)} < -1 \ if \ y^{(i)} = -1$$

$$w_0 x_0 + w_1 x_1 + \cdots + w_m x_m = \sum_{j=0}^{m} \boldsymbol{x}_j \boldsymbol{w}_j = \boldsymbol{w}^T \boldsymbol{x}$$

Python Machine Learning; Raschkka & Mirjalili

Support Vector Machine (SVM): Training a Classifier

Same as finding parameters that maximizes margin:

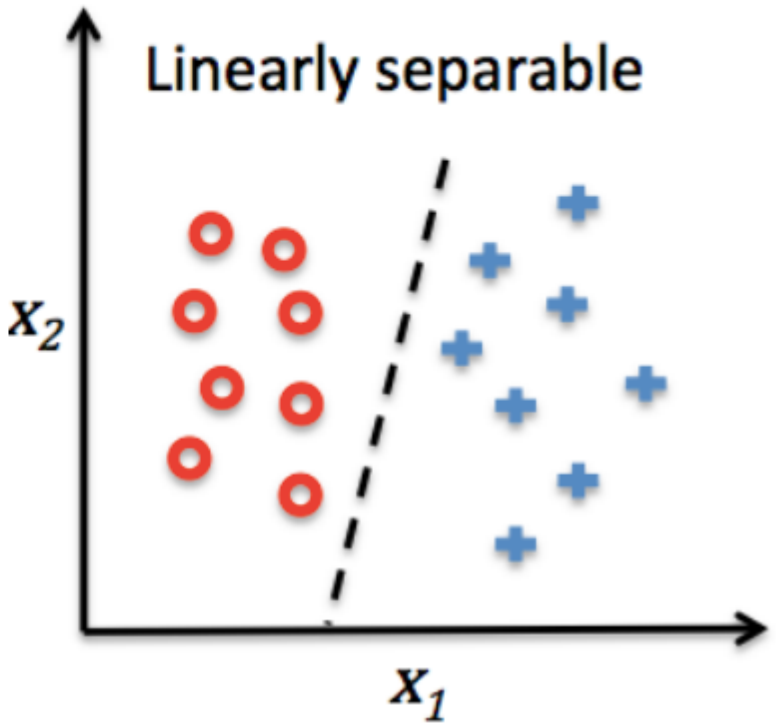
$$\boxed{\min_{\mathbf{w},w_0} \frac{1}{2}||\mathbf{w}||^2}$$

Constraint that enforces that all examples fall outside of the margin space:

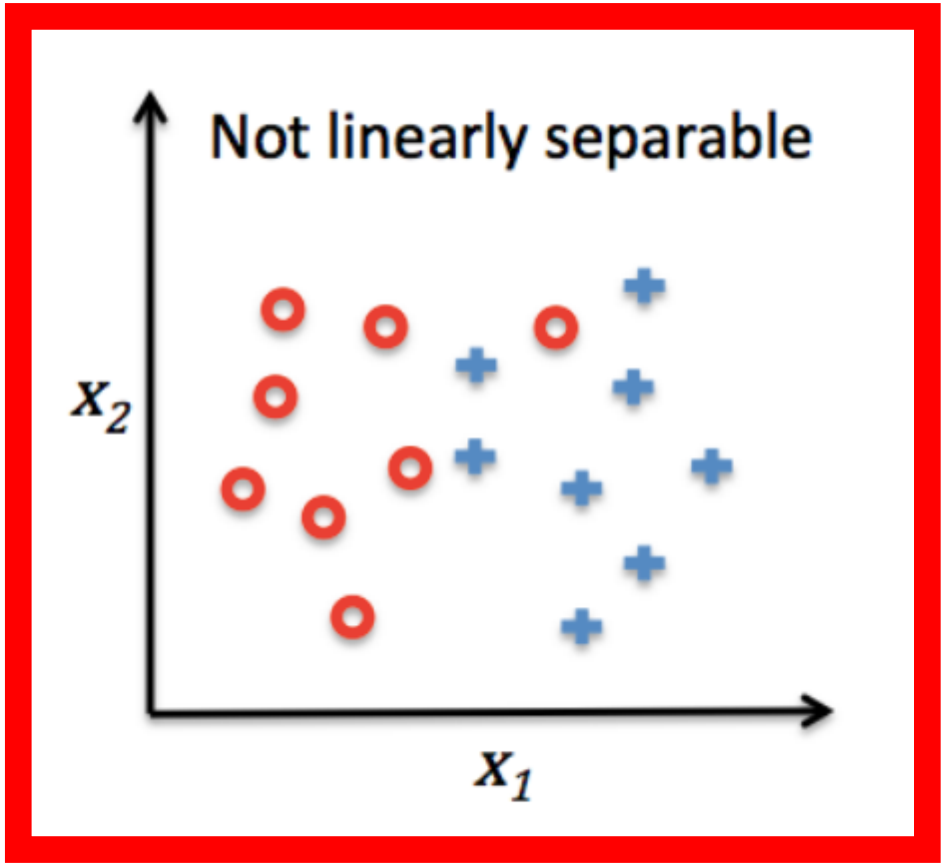
$$\boxed{s.t. \forall i \quad y^{(i)}\left(w_0 + \boldsymbol{w}^T \boldsymbol{x}^{(i)}\right) \geq 1,}$$

Can be solved with a quadratic programming solver… learn more about this at:
- “The Nature of Statistical Learning and Theory, by Vladimir Vapnik
- A Tutorial on Support Vector Machines for Pattern Recognition by Chris J. C. Burges’

What if the Decision Boundary is Not Linear?



Hard-Margin Classification



Soft-Margin Classification

Python Machine Learning; Raschkka & Mirjalili

Soft-Margin Classification

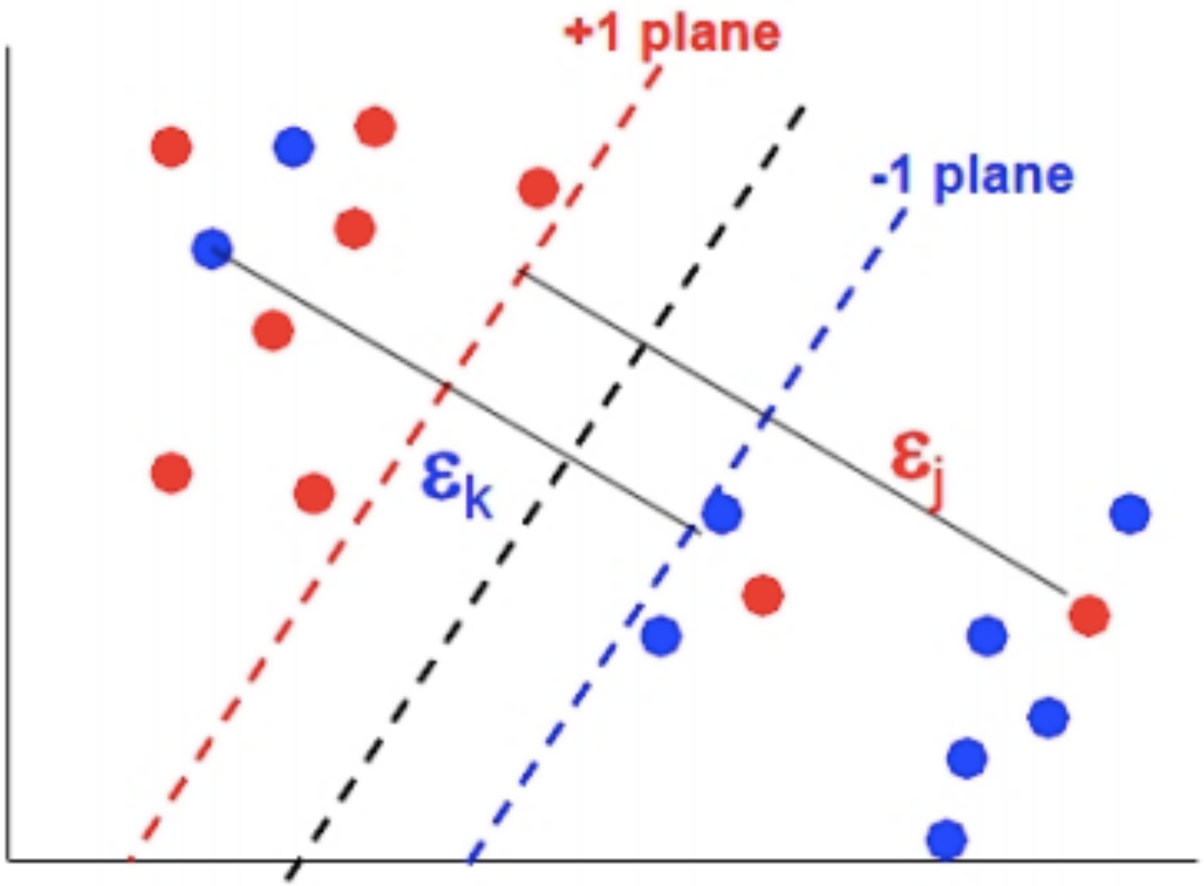
Introduce "slack" variable:

$$\boldsymbol{w}^T \boldsymbol{x}^{(i)} \geq 1 - \xi^{(i)} \quad \text{if} \quad y^{(i)} = 1$$

$$\boldsymbol{w}^T \boldsymbol{x}^{(i)} \leq -1 + \xi^{(i)} \quad \text{if} \quad y^{(i)} = -1$$

$$\min \frac{1}{2}||\mathbf{w}||^2 + \lambda \sum_{i=1}^{N} \xi_i$$

$$\text{s.t} \quad \xi_i \geq 0; \quad \forall i \quad t^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)}) \geq 1 - \xi_i = 0$$



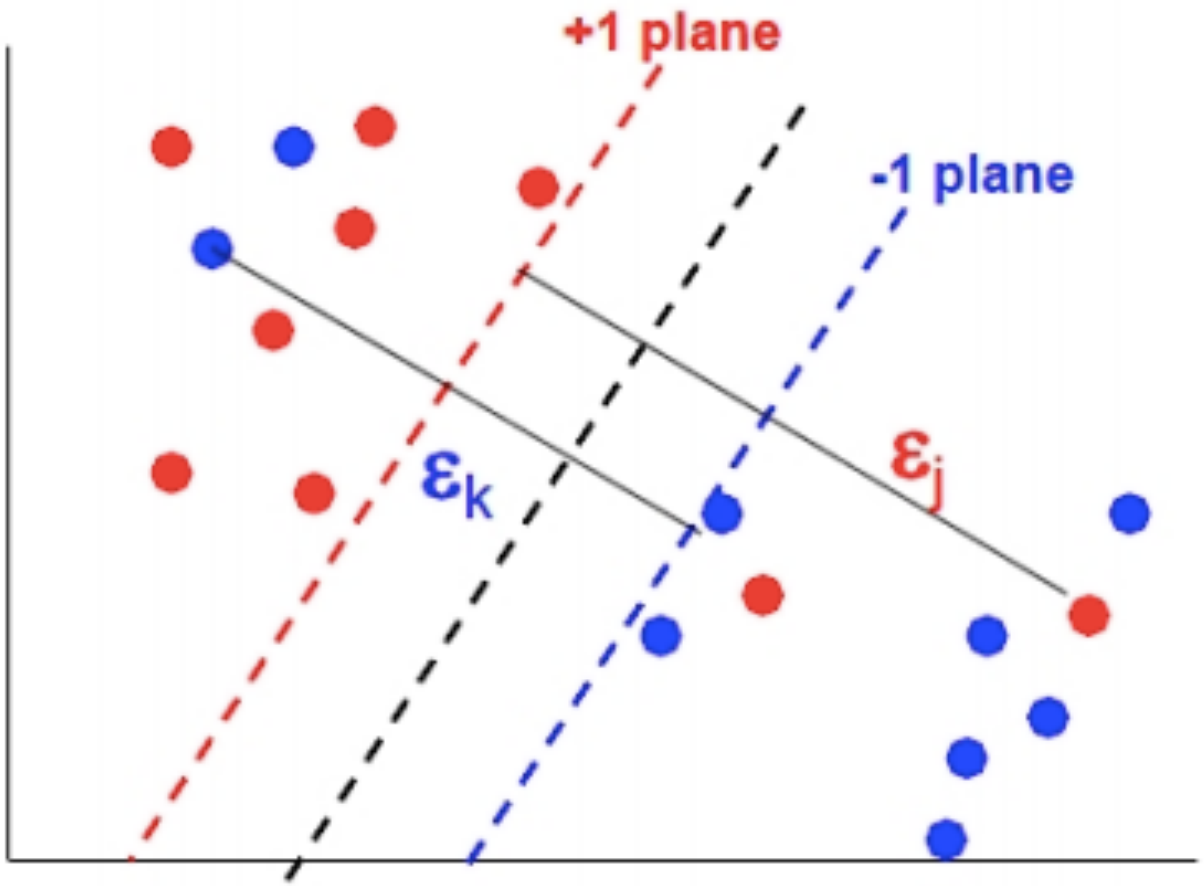
Python Machine Learning; Raschkka & Mirjalili

Soft-Margin Classification

Controls how much slack is allowed:

$$\min \frac{1}{2}||\mathbf{w}||^2 + \boxed{\lambda} \sum_{i=1}^{N} \xi_i$$

$$\text{s.t} \quad \xi_i \geq 0; \quad \forall i \quad t^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)}) \geq 1 - \xi_i = 0$$



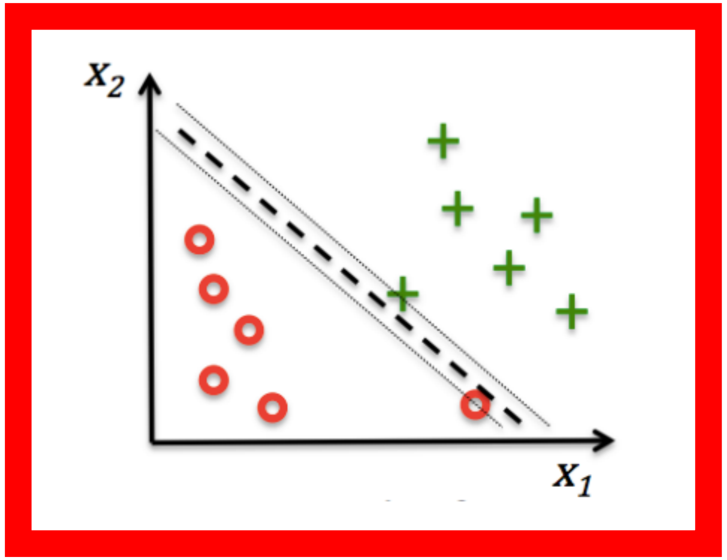
Python Machine Learning; Raschkka & Mirjalili

Soft-Margin Classification

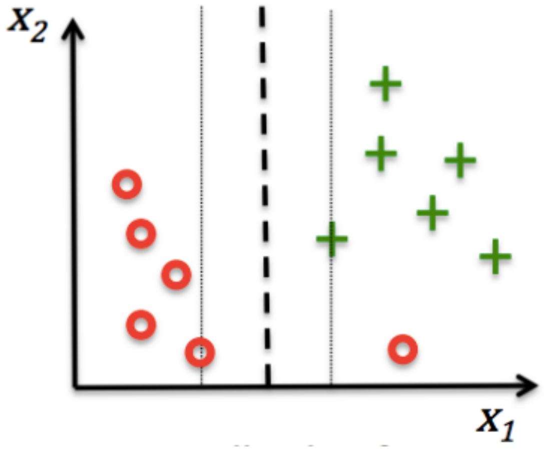
$$\min \frac{1}{2}||\mathbf{w}||^2 + \boxed{\lambda} \sum_{i=1}^{N} \xi_i$$

$$\text{s.t} \quad \xi_i \geq 0; \quad \forall i \;\; t^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)}) \geq 1 - \xi_i = 0$$

A



B



(Increases priority placed on minimizing error)

Which plot shows when the slack variable is **larger**?

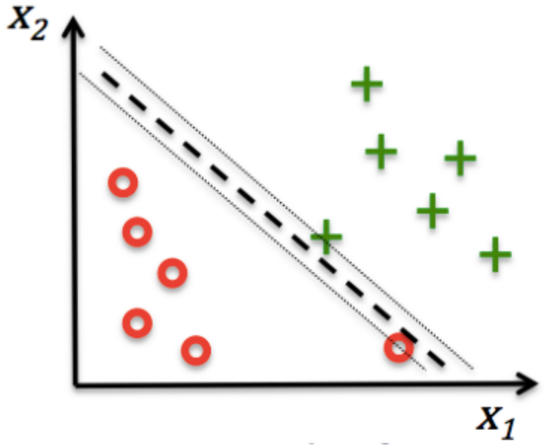
Python Machine Learning; Raschkka & Mirjalili

Soft-Margin Classification

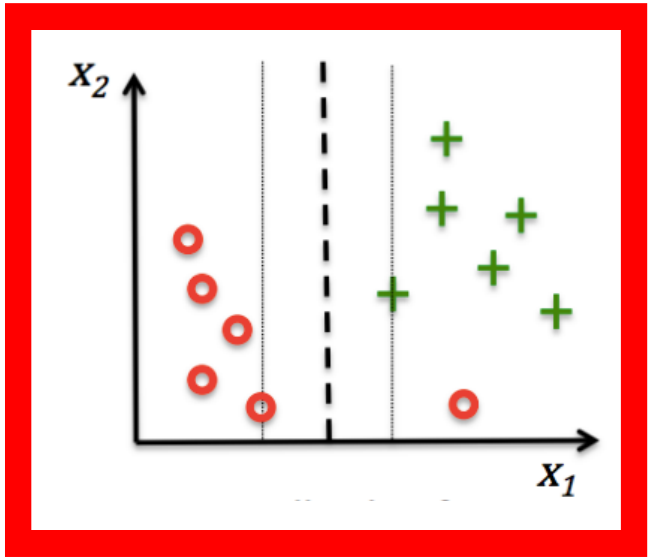
$$\min \frac{1}{2}||\mathbf{w}||^2 + \boxed{\lambda} \sum_{i=1}^{N} \xi_i$$

$$\text{s.t} \quad \xi_i \geq 0; \quad \forall i \quad t^{(i)}(\mathbf{w}^T \mathbf{x}^{(i)}) \geq 1 - \xi_i = 0$$

A



B

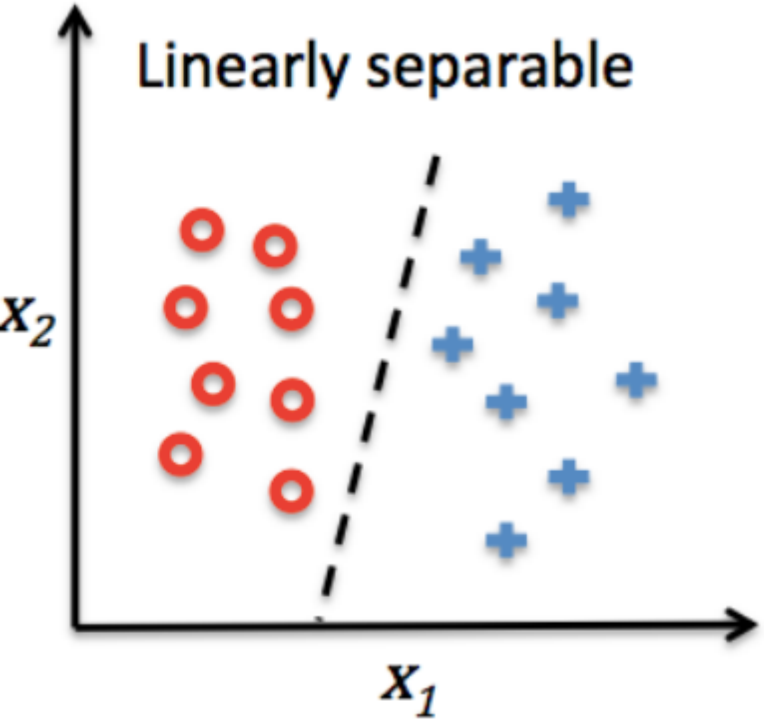


(Increases priority placed on maximizing margin)

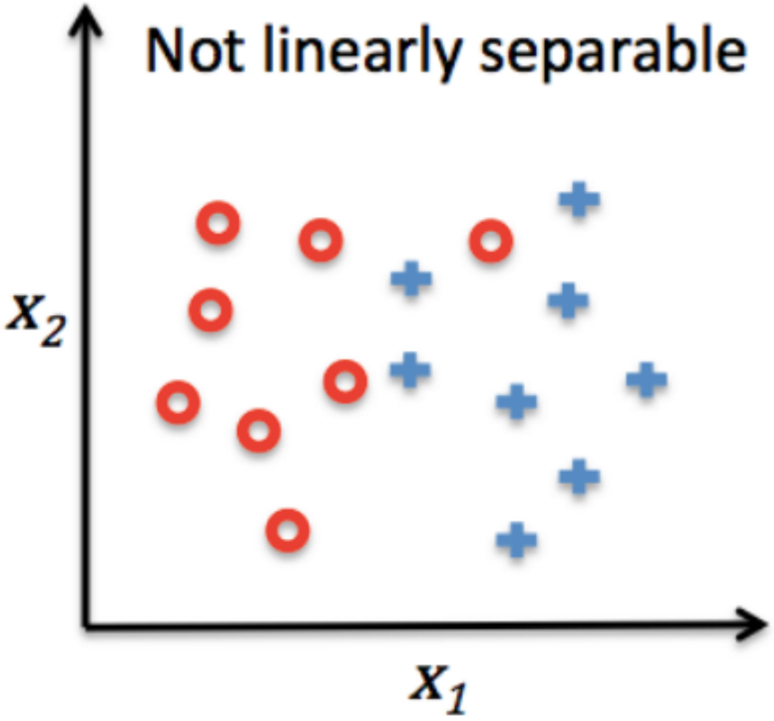
Which plot shows when the slack variable is **smaller**?

Python Machine Learning; Raschkka & Mirjalili

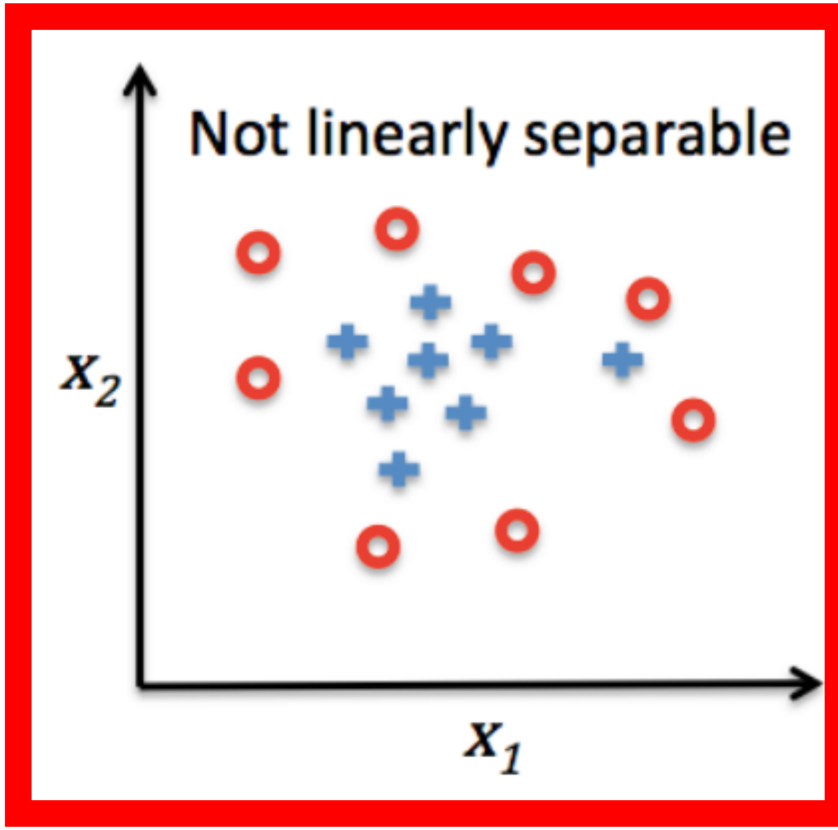
What if the **Decision Boundary is Not Linear?**



Hard-Margin Classification



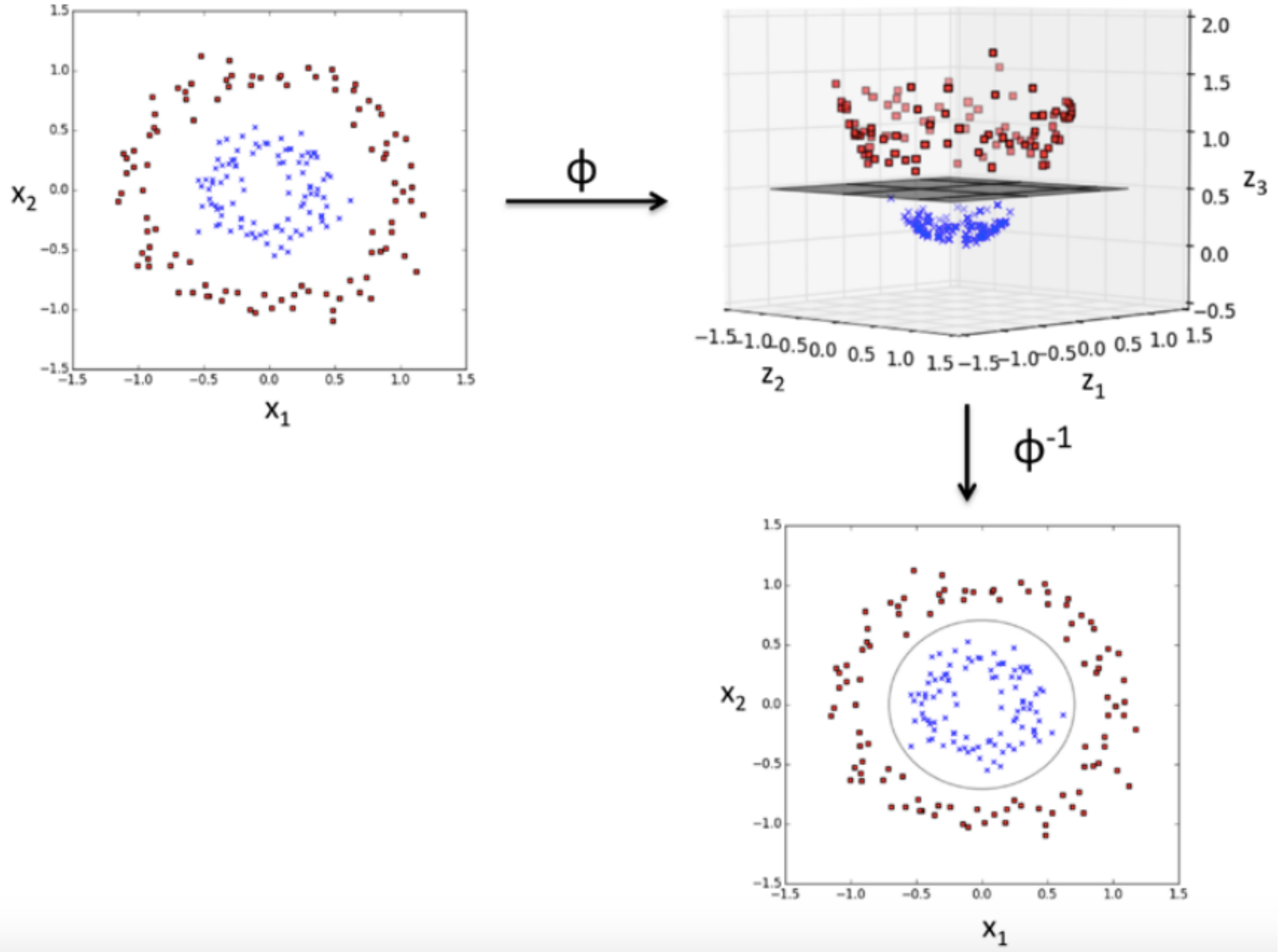
Soft-Margin Classification



Python Machine Learning; Raschkka & Mirjalili

Kernelized Support Vector Machines

- Recall polynomial regression?
  - Project features to higher order space
- Kernels efficiently project features to higher order spaces
- What conversion to use? e.g.,
  - Polynomial kernel
  - Gaussian Radial Basis Function kernel



Python Machine Learning; Raschkka & Mirjalili

What are SVM’s Strengths

- Insensitive to outliers (only relies on support vectors to choose dividing line)
- Once trained, prediction is fast
- Requires little memory (rely on a few support vectors)
- Work well with high-dimensional data

What are SVM’s Weaknesses

- Prohibitive computational costs for large datasets
- Performance heavily dependent on soft margin value for non-linear classification
- Does not have a direct probabilistic interpretation

Today’s Topics

- Evaluating Machine Learning Models Using Cross-Validation
- Naïve Bayes
- Support Vector Machines