Learning (II)

Mingsheng Long

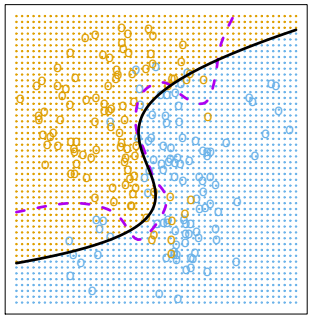
Tsinghua University

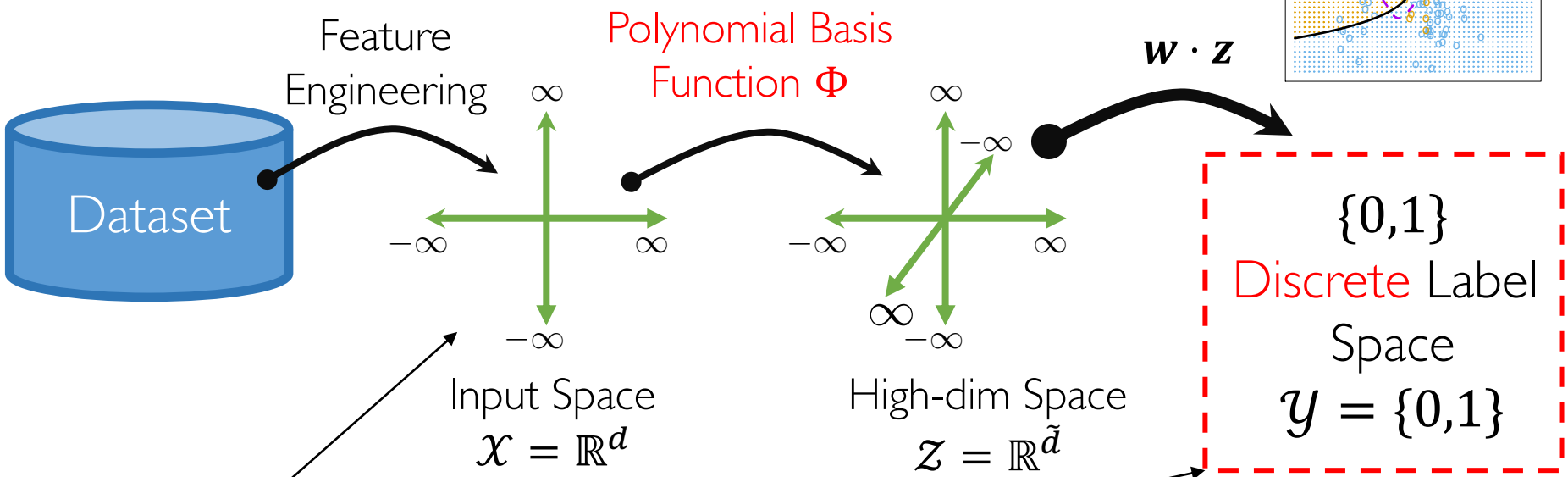
Spring 2023

Outline

- **Linear Classification**
  - **Logistic Regression**
  - Softmax Regression
- Support Vector Machine (SVM)
  - Soft-SVM
  - Kernel Soft-SVM
- Decision Tree
- Random Forest

2-class Classification: An Example: Spam Filtering





From: mingsheng@tsinghua.edu.cn
Date: 5:20 p.m. September 14, 2020
Subject: Class announcement

Hello students,
Welcome to Machine Learning!
Here's what...

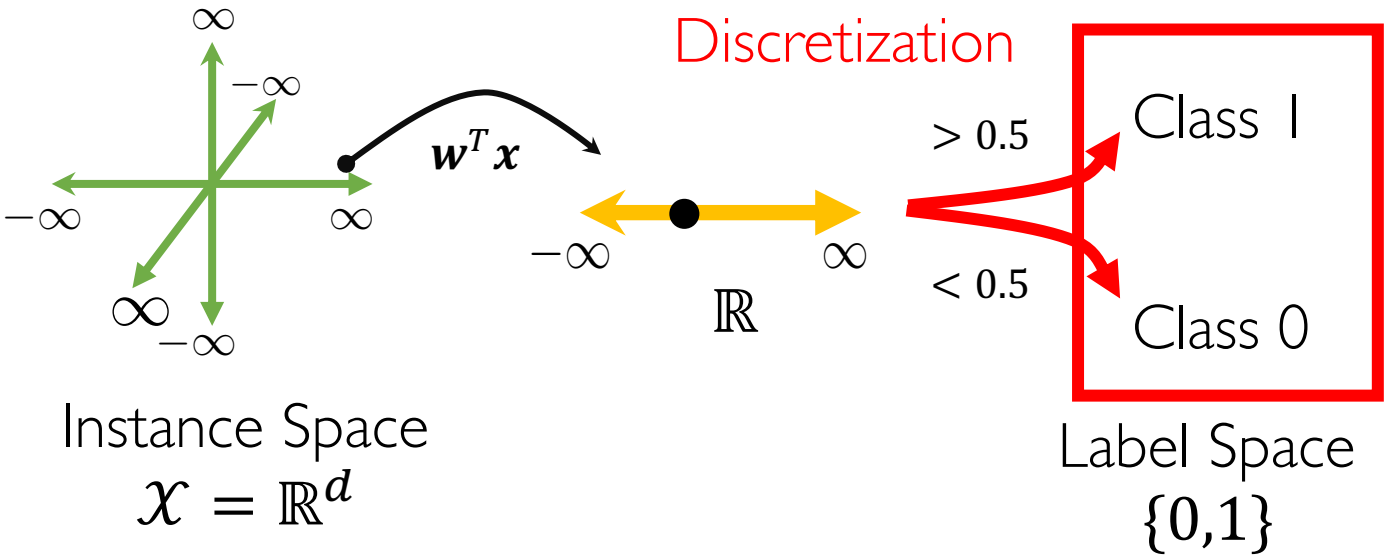
Category	Not-spam
FracOfAlpha	1.0
Sent at midnight	0
Subject len	18
Contains 'dollar'	0

Can we simply use squared loss?

$$\min_{\boldsymbol{w}} \sum_{i=1}^{n} (h_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2 + \lambda \Omega(\boldsymbol{w})$$

Regression to Classification

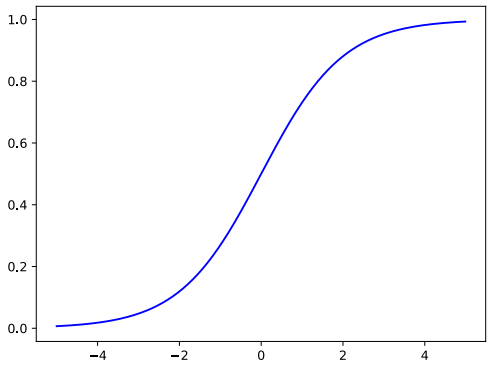
- How to design loss function over linear hypothesis for classification?
- Naïve idea: $\min_{\boldsymbol{w}} \sum_{i=1}^{n} (h_{\boldsymbol{w}}(\boldsymbol{x}_i) - y_i)^2 + \lambda\Omega(\boldsymbol{w})$ where $y_i \in \{0,1\}$
  - Discretize the continuous output $h_{\boldsymbol{w}}(\boldsymbol{x}_i)$ to be $\{0,1\}$
- During test part:



Logistic Regression: Logistic Function

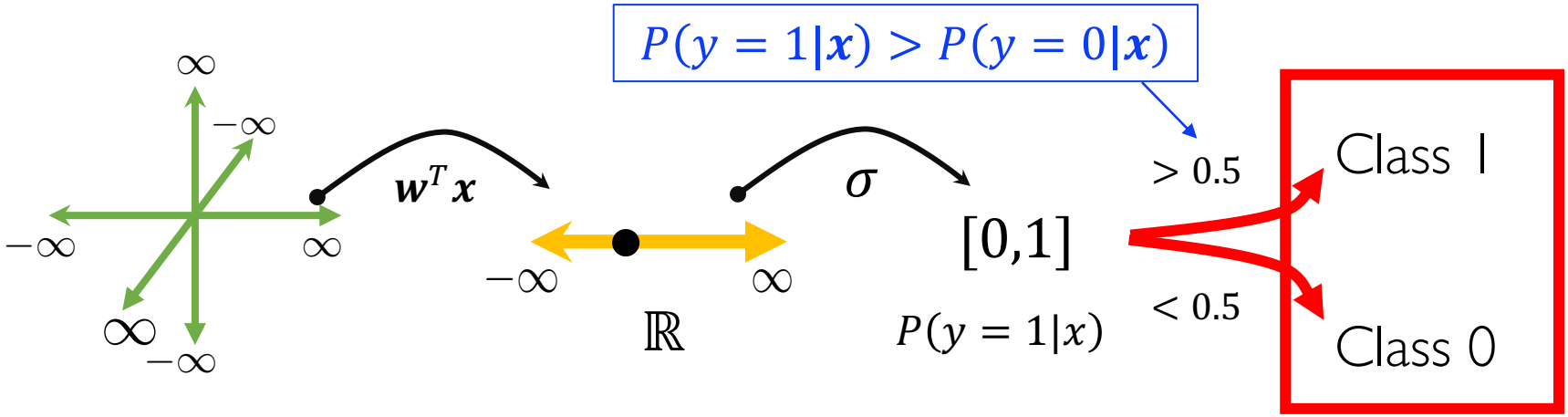
- To map the output of $h(\boldsymbol{x})$ into $[0,1]$, we use sigmoid function:

$$\sigma(t) = \frac{1}{1+e^{-t}}$$



  - Sigmoid maps $\mathbb{R}$ to $[0,1]$.
  - $t \to +\infty, \sigma(t) \to 1$.

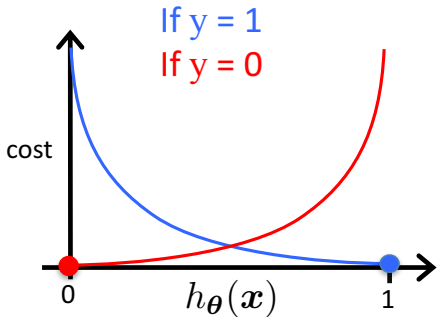
- View $\sigma(h(\boldsymbol{x})) = p(y=1|\boldsymbol{x})$ as the probability to label $\boldsymbol{x}$ as $y=1$.



Logistic Regression: Cross-Entropy Loss

$$\ell(h(\boldsymbol{x}_i), y_i) = \begin{cases} -\log[\sigma(\boldsymbol{w}^T \boldsymbol{x}_i)] & y_i = 1 \\ -\log[1 - \sigma(\boldsymbol{w}^T \boldsymbol{x}_i)] & y_i = 0 \end{cases}$$

sigma是之前预测的概率

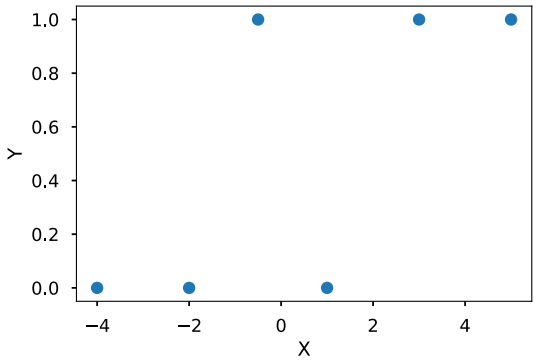


Check. If $y_i = 1$,

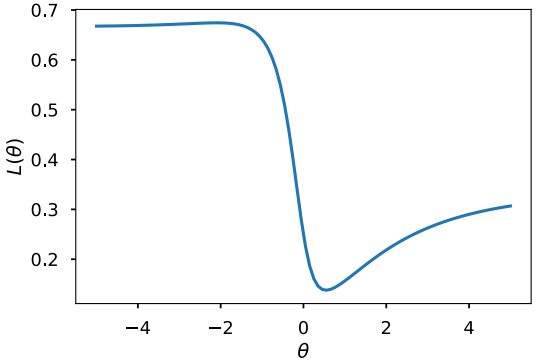
- $\sigma(\boldsymbol{w}^T \boldsymbol{x}_i) \to 0$, loss goes to infinity;
- $\sigma(\boldsymbol{w}^T \boldsymbol{x}_i) \to 1$, loss goes to 0.

$P(y = 1|\boldsymbol{x}_i)$

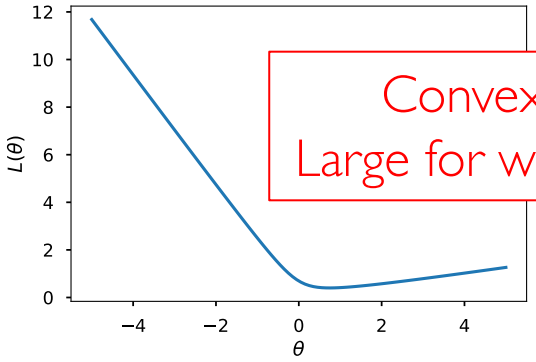
Toy data



Squared Loss Surface



Cross-Entropy Surface



Logistic Regression: Regularization

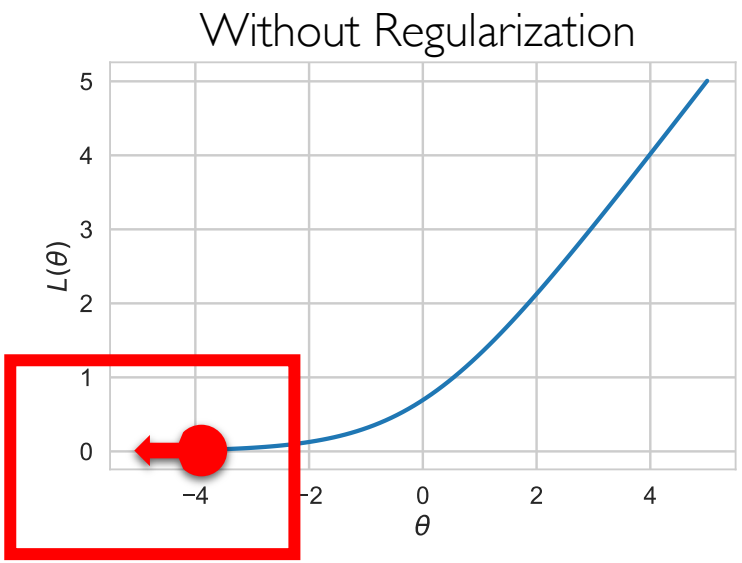
- Prevent weights from diverging on linearly separable data

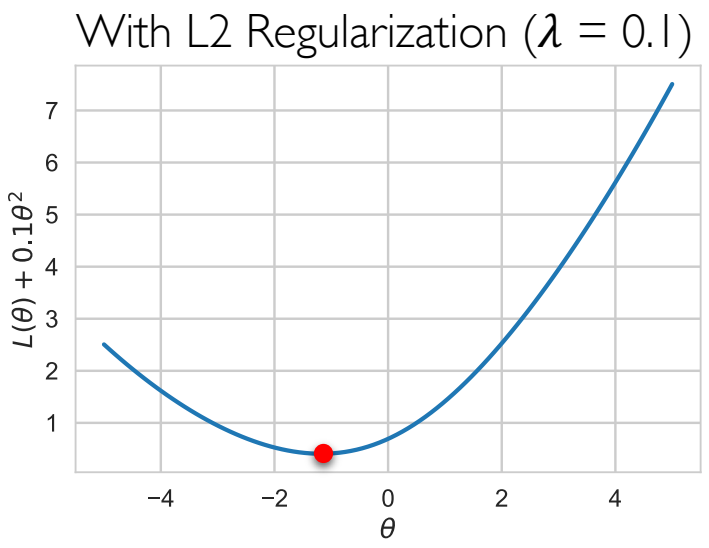
$$\hat{\epsilon}(w) = -\sum_{i=1}^{n}\{y_i \log \sigma(h_{\boldsymbol{w}}(\boldsymbol{x})) + (1-y_i)\log[1-\sigma(h_{\boldsymbol{w}}(\boldsymbol{x}))] + \lambda \sum_{j=1}^{d} w_j^2$$

- Similar with regression: L1, L2…

Loss Surface

$\rightarrow \infty$



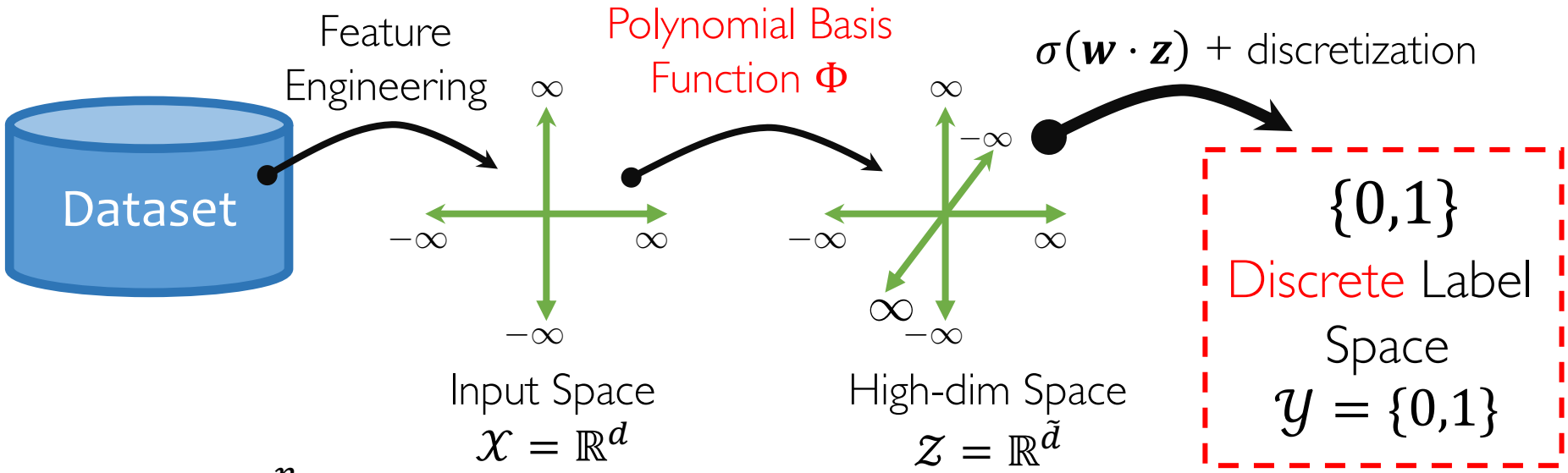


Logistic Regression

2-class Classification:

Hypothesis Space

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x})$$



$$\hat{\epsilon}(w) = -\sum_{i=1}^{n} \{y_i \log \sigma(h_{\boldsymbol{w}}(\boldsymbol{x})) + (1 - y_i) \log[1 - \sigma(h_{\boldsymbol{w}}(\boldsymbol{x}))]\} + \lambda \Omega(\boldsymbol{w})$$

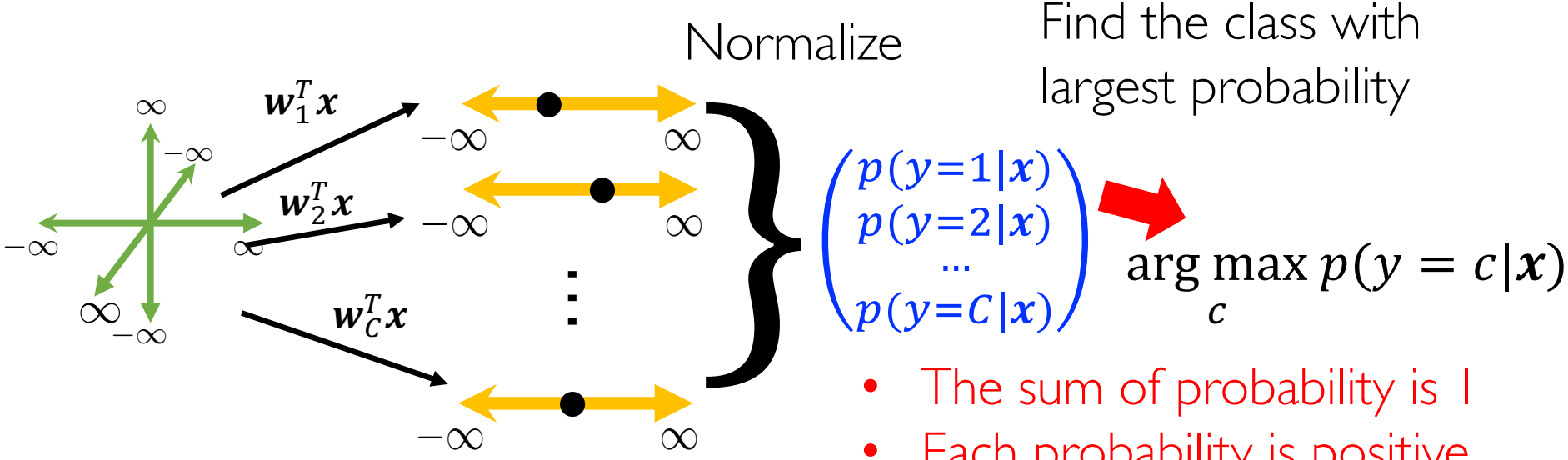
- How to solve this problem?
  - No analytic solution. Using GD, SGD, Newton method…
- How to deal with multiclass problems?

Outline

- **Linear Classification**
  - Logistic Regression
  - **Softmax Regression**
- Support Vector Machine (SVM)
  - Soft-SVM
  - Kernel Soft-SVM
- Decision Tree
- Random Forest

Binary-class to Multi-class

- We keep on mapping the output of linear hypothesis into probability.
- There are multiple classes, so we use multiple linear predictors to predict the probability: $\boldsymbol{w_1, w_2, \ldots, w_C}$.
- How to change the outputs of multiple predictors into probability?



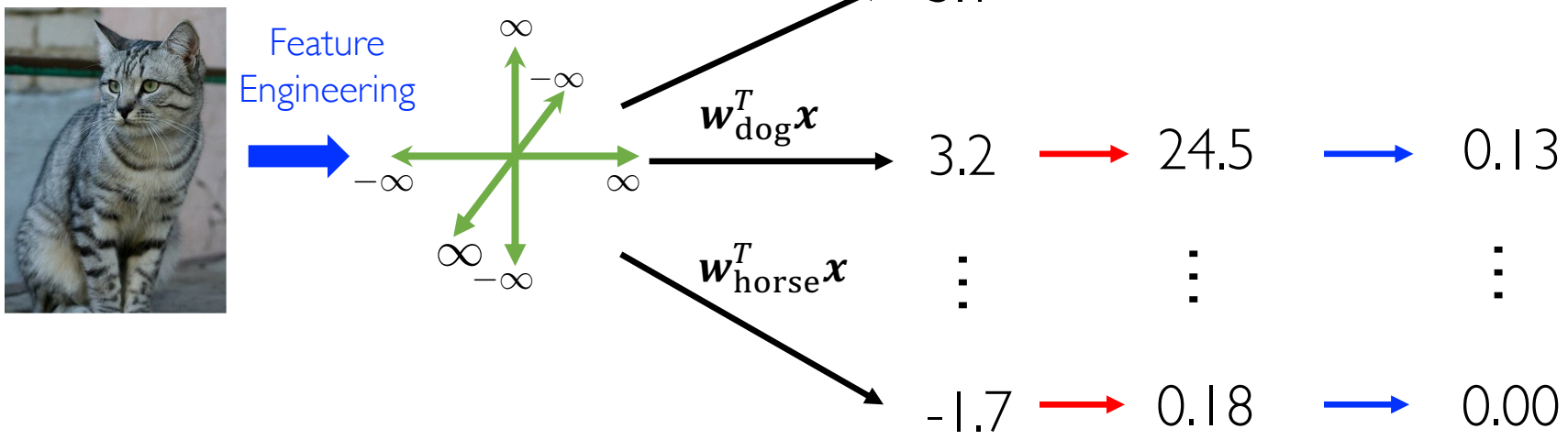
- The sum of probability is 1
- Each probability is positive

Softmax Regression: Softmax Function

- Softmax function normalizes multiple outputs in a probability vector:

$$p(y = i|\boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_i^T \boldsymbol{x})}{\sum_{r=1}^{C} \exp(\boldsymbol{w}_r^T \boldsymbol{x})}$$

Sum over all classes.



Softmax Regression: Cross-Entropy Loss

Loss for each data point (x, y):

$$\ell(h(\boldsymbol{x}_i), y_i) = \begin{cases} -\log\left[\dfrac{\exp(\boldsymbol{w}_1^T \boldsymbol{x})}{\sum_{r=1}^{C} \exp(\boldsymbol{w}_r^T \boldsymbol{x})}\right] & y_i = 1 \\ -\log\left[\dfrac{\exp(\boldsymbol{w}_2^T \boldsymbol{x})}{\sum_{r=1}^{C} \exp(\boldsymbol{w}_r^T \boldsymbol{x})}\right] & y_i = 2 \\ \vdots & \\ -\log\left[\dfrac{\exp(\boldsymbol{w}_C^T \boldsymbol{x})}{\sum_{r=1}^{C} \exp(\boldsymbol{w}_r^T \boldsymbol{x})}\right] & y_i = C \end{cases}$$

Equivalent to maximum log-likelihood estimation

These two hypotheses have same outputs

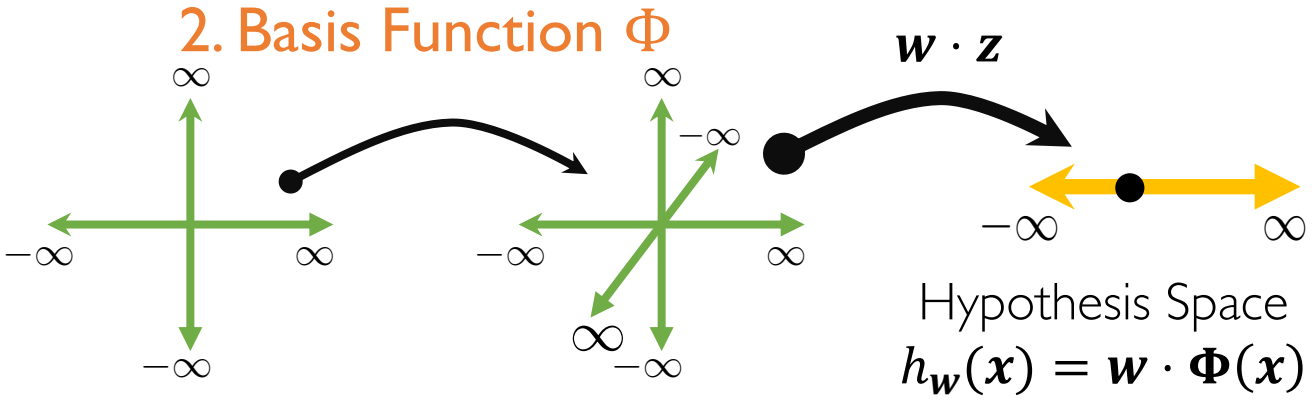
$$\begin{pmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \\ \cdots \\ \boldsymbol{w}_C \end{pmatrix} \quad \begin{pmatrix} \boldsymbol{w}_1 + \boldsymbol{V} \\ \boldsymbol{w}_2 + \boldsymbol{V} \\ \cdots \\ \boldsymbol{w}_C + \boldsymbol{V} \end{pmatrix}$$

This loss is still convex. ✅

Add regularization. Or the parameter will go infinity.

Minimize this loss: GD, SGD…

Linear Models: Summary



Objective function may not be linear to the parameters.

$$\min_{w} \sum_{i=1}^{n} \ell(h_w(x_i), y_i) + \lambda\Omega(w)$$

4. Optimization

- High-dimension
  - GD
  - Subgradient
- High-dimension big data
  - SGD

1. Loss Function

- Regression
  - Squared Loss
- Classification
  - Cross-Entropy

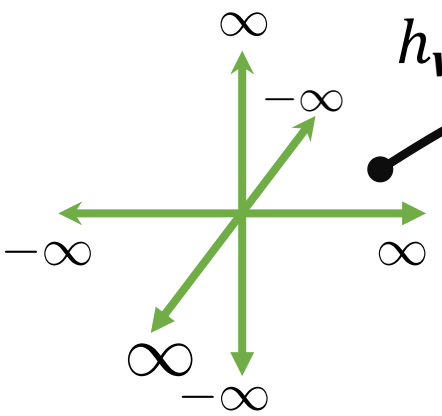
3. Regularization

- General
  - L2 Regularizer
- Sparse Solution
  - L1 Regularizer

Outline

- Linear Classification
  - Logistic Regression
  - Softmax Regression
- **Support Vector Machine (SVM)**
  - Soft-SVM
  - Kernel Soft-SVM
- Decision Tree
- Random Forest

Linear Classification



Instance Space
$\mathcal{X} = \mathbb{R}^d$

$h_{\boldsymbol{w}} = \boldsymbol{w} \cdot \boldsymbol{x} + b$

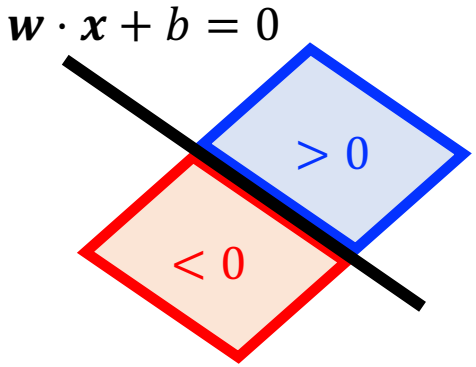
> 0 Class $+1$

< 0 Class -1

Discrete Label Space
$\mathcal{Y} = \{-1, 1\}$

New convention: we explicitly write the intercept b

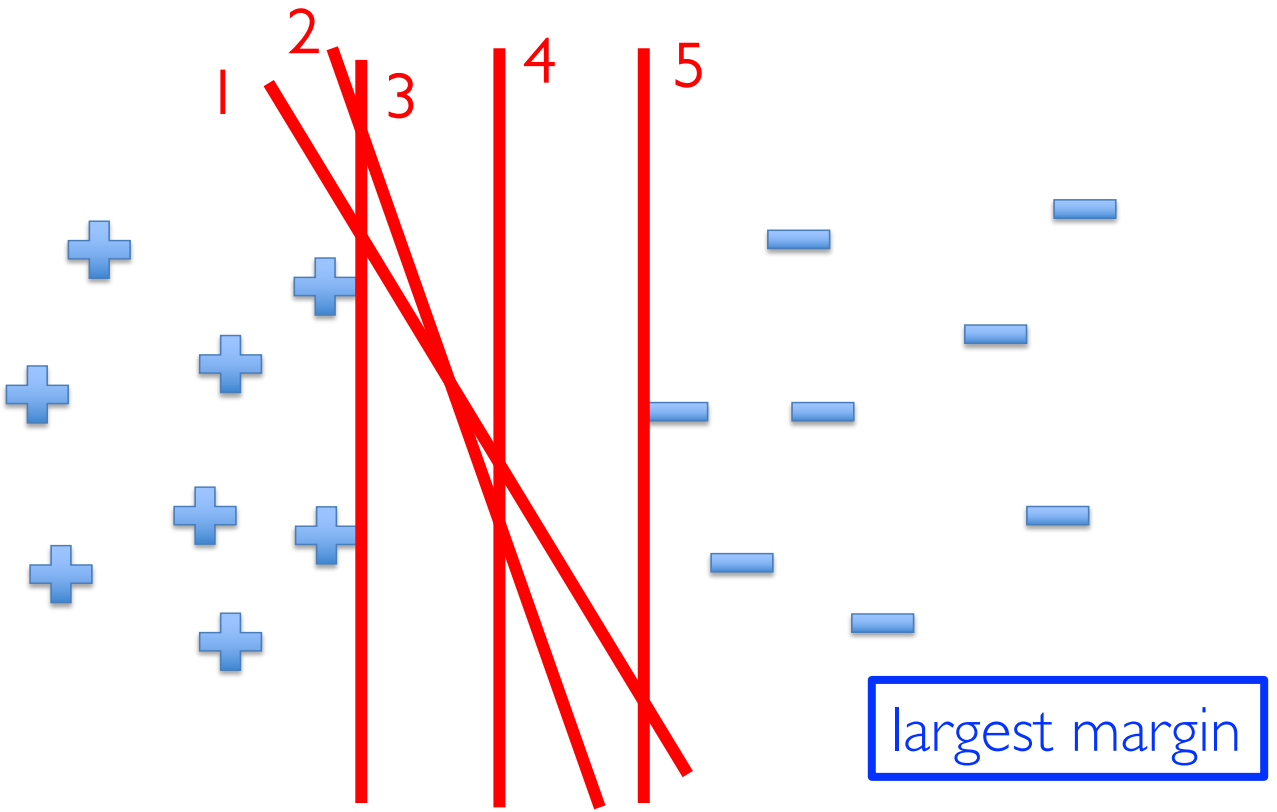
New convention: we change $\{0,1\}$ to $\{-1, +1\}$.



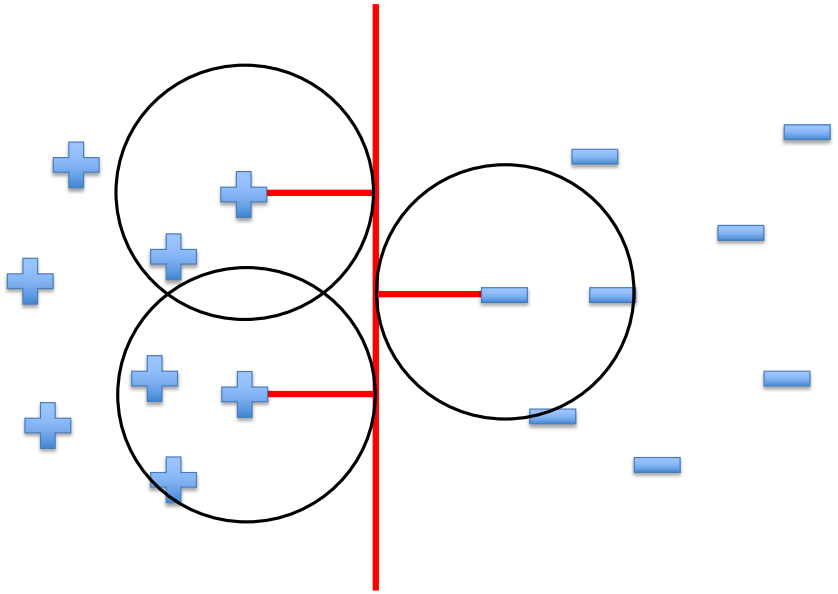
- How to evaluate classifier? 0-1 loss: $\mathbf{1}[h(\boldsymbol{x}) \neq y]$
  - Correct: $\mathbf{1}[h(\boldsymbol{x}) \neq y] = 0$ or $y(\boldsymbol{w} \cdot \boldsymbol{x} + b) > 0$
  - Wrong: $\mathbf{1}[h(\boldsymbol{x}) \neq y] = 1$ or $y(\boldsymbol{w} \cdot \boldsymbol{x} + b) < 0$
- Problem: How to choose the best linear classifier (hyperplane)?

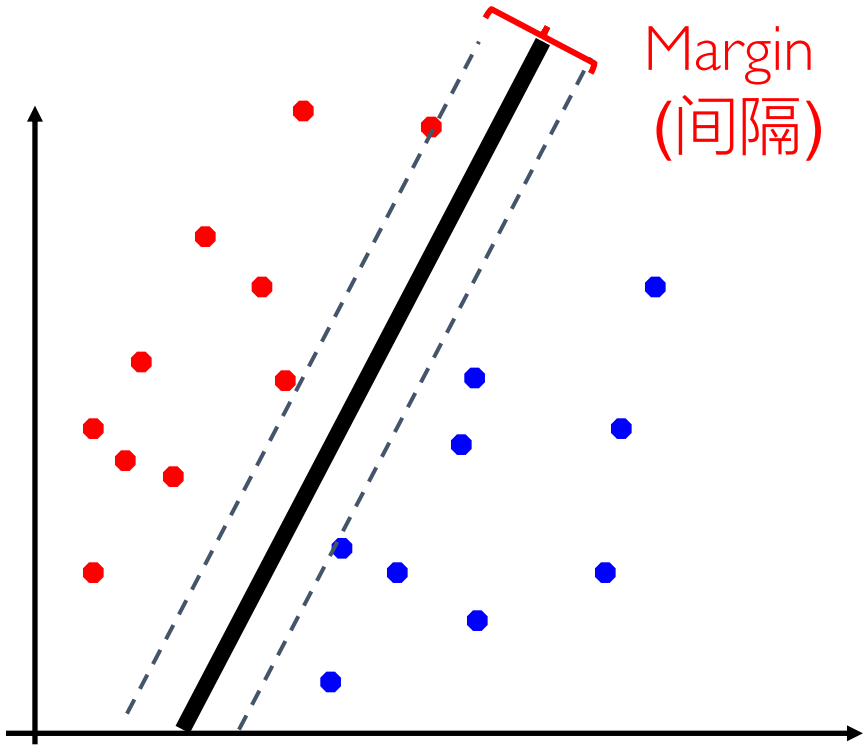
Which is the Best Classifier

- Consider the simplest linear separable setting.
- Which classifier is the best classifier, i.e. will yield the lowest test error?



Support Vector Machine: Margin





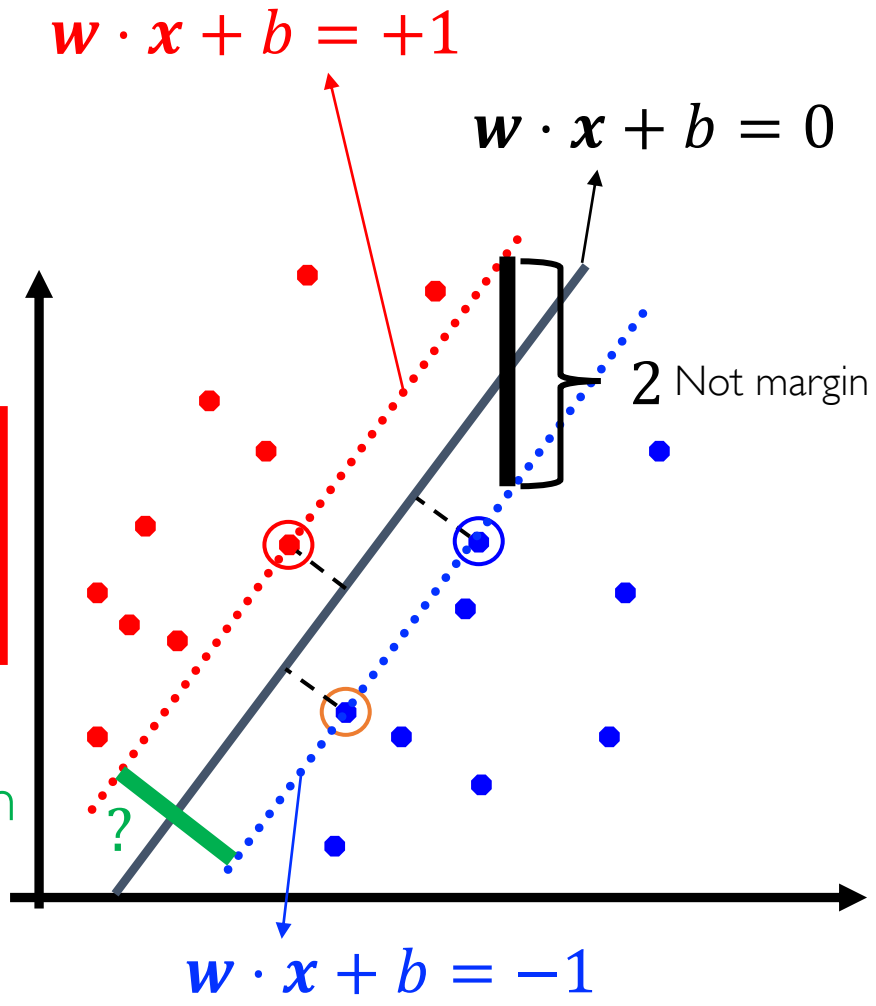
- Margin: Twice of the distance to the closest points of either class.
  - Twice of the largest noise that can be tolerated by the classifier.
- Problem: How to find the linear classifier with the largest margin?

Support Vector Machine: Margin

- Requirements:
  - The margin is the largest.
  - Classify all data points correctly.
- Constrained optimization problem:

$$\max_{\boldsymbol{w},b} \ \text{margin}(\boldsymbol{w}, b)$$
$$\text{s.t. } y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1, 1 \leq i \leq n$$

- $1 > 0$ is not margin.
- How to quantify the margin?

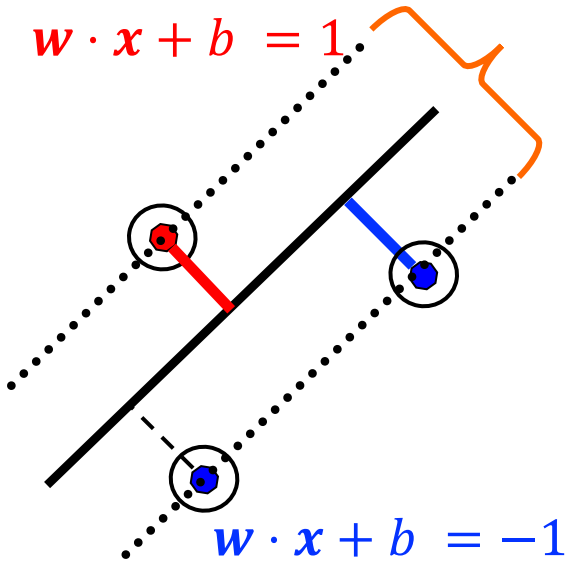


Hard-Margin Support Vector Machine

- Recall: The distance between point $\boldsymbol{x}$ and hyperplane $(\boldsymbol{w}, b)$ is:

$$r = \frac{|\boldsymbol{w} \cdot \boldsymbol{x} + b|}{\|\boldsymbol{w}\|_2}$$

- The points closest to the classifier lies on the lines $\boldsymbol{w} \cdot \boldsymbol{x} + b = \pm 1$.



$$\gamma = \frac{1}{\|\boldsymbol{w}\|_2} + \frac{|-1|}{\|\boldsymbol{w}\|_2} = \frac{2}{\|\boldsymbol{w}\|_2}$$

margin

- Hard-margin Support Vector Machine (SVM):

$$\max_{\boldsymbol{w},b} \frac{2}{\|\boldsymbol{w}\|_2}$$

Linearly Separable

$$\text{s.t. } y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1, 1 \leq i \leq n$$

Hard-Margin Support Vector Machine

$$\max_{\boldsymbol{w},b} \frac{2}{\|\boldsymbol{w}\|_2}$$
$$\text{s.t. } y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1, 1 \leq i \leq n$$

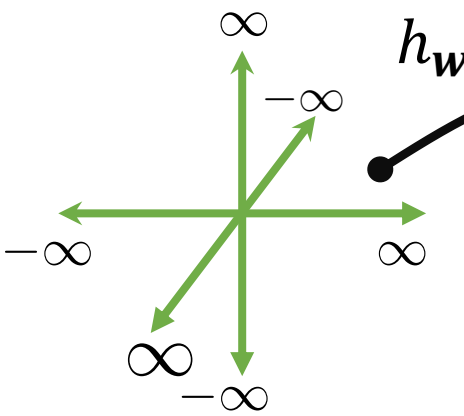
- Non-convex problem. But it is equivalent to:

Hard-margin Support Vector Machine (SVM)

$$\min_{\boldsymbol{w},b} \frac{1}{2} \|\boldsymbol{w}\|_2^2$$
$$\text{s.t. } y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1, 1 \leq i \leq n$$

- The factor $\frac{1}{2}$ is added for computational convenience.
- $\|\boldsymbol{w}\|_2$ is non-convex but $\|\boldsymbol{w}\|_2^2$ is a convex function of $\boldsymbol{w}$.

Support Vector Machine (SVM)



Instance Space

$\mathcal{X} = \mathbb{R}^d$

$h_{\boldsymbol{w}} = \boldsymbol{w} \cdot \boldsymbol{x} + b$

> 0 Class $+1$

< 0 Class -1

Discrete Label Space

$\mathcal{Y} = \{-1, 1\}$

Linearly Separable

New convention: we explicitly write the intercept b

New convention: we change $\{0,1\}$ to $\{-1, +1\}$.

Hard-margin Support Vector Machine (SVM)

$$\min_{\boldsymbol{w},b} \frac{1}{2} \|\boldsymbol{w}\|_2^2$$

$$\text{s.t. } y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1, 1 \leq i \leq n$$

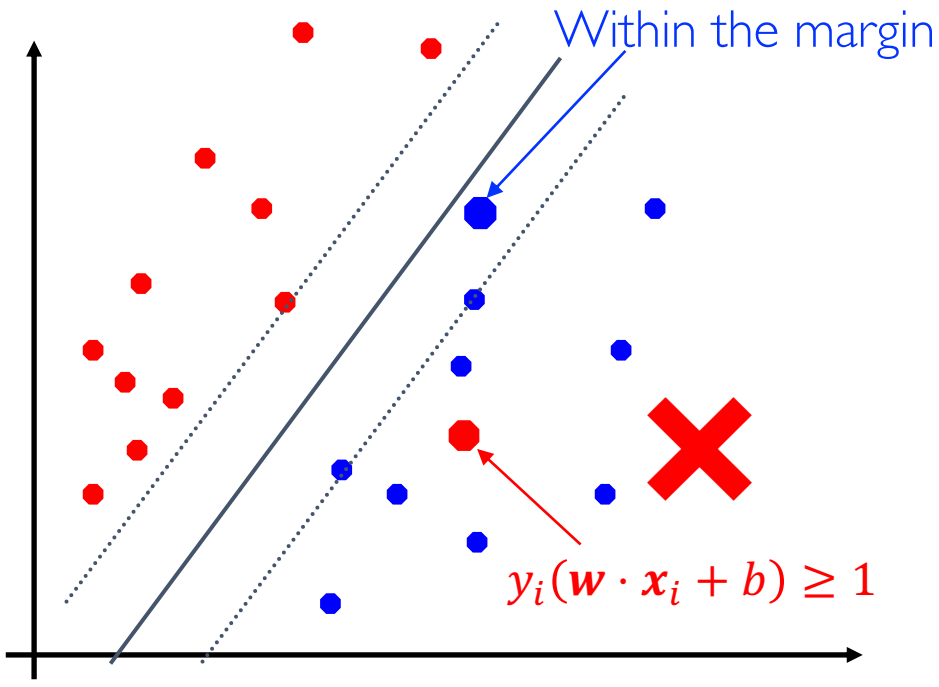
This is only for the linearly separable case. Hardly used in practice!

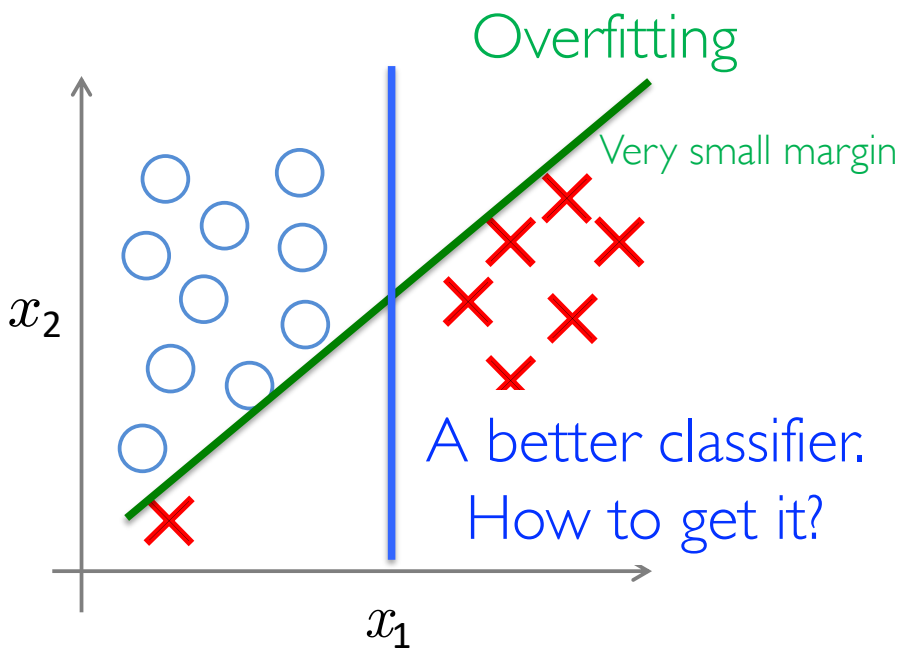
Outline

- Linear Classification
  - Logistic Regression
  - Softmax Regression
- **Support Vector Machine (SVM)**
  - **Soft-SVM**
  - Kernel Soft-SVM
- Decision Tree
- Random Forest

Linearly Non-Separable Classification

- In the linearly non-separable case, we cannot find a solution to the hard-margin support vector machine (left).
- Even we can find a solution, small margin may cause overfitting (right).





Soft-Margin Support Vector Machine

- Instead of constraining all data points to be correctly classified:
  - Allow some points on the wrong side of the margin.
  - Their number should be small.

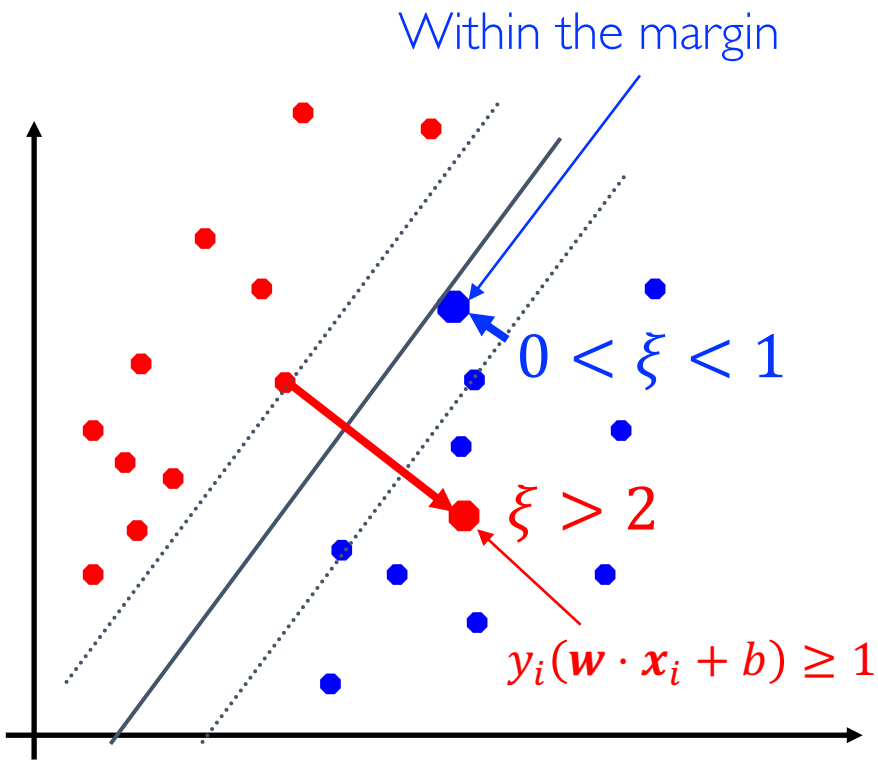
$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|_2^2$$

$$\text{s.t. } y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0, \sum_{i=1}^{n} \xi_i \leq n'$$

$$1 \leq i \leq n$$

From hard-margin to soft-margin



Slack variables (松弛变量): $\xi_i, i \in [n]$

Soft-Margin Support Vector Machine

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|_2^2$$

$$\text{s.t. } y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0, \sum_{i=1}^{n} \xi_i \leq n', 1 \leq i \leq n$$

- Computationally, we re-express in the (Lagrangian) equivalent form:

Soft-margin Support Vector Machine (SVM)

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n} \xi_i$$

$$\text{s.t. } y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0, 1 \leq i \leq n$$

Soft-SVM: Hinge Loss

Soft-margin Support Vector Machine (SVM)

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

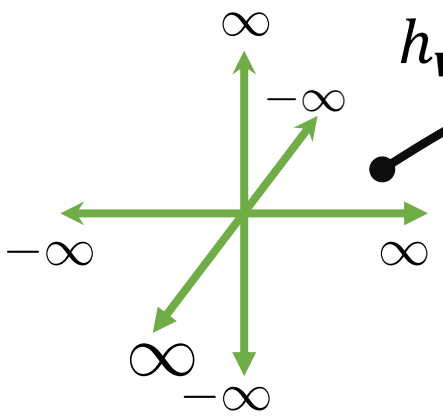
$$\text{s.t. } y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0, 1 \leq i \leq n$$

- Define Hinge loss $\ell(f(\boldsymbol{x}), y) = \max\{0, 1 - yf(\boldsymbol{x})\}$
  - For the linear hypothesis: $\ell(f(\boldsymbol{x}), y) = \max\{0, 1 - y(\boldsymbol{w}\cdot\boldsymbol{x} + b)\}$
- Theorem: Soft-SVM is equivalent to a Regularized Risk Minimization:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \frac{C}{n}\sum_{i=1}^{n}\max\{0, 1 - y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b)\}$$

Support Vector Machine (SVM)



Instance Space
$\mathcal{X} = \mathbb{R}^d$

$h_{\boldsymbol{w}} = \boldsymbol{w} \cdot \boldsymbol{x} + b$

> 0 → Class $+1$

< 0 → Class -1

Discrete Label Space
$\mathcal{Y} = \{-1, 1\}$

Linearly Non-separable

New convention: we explicitly write the intercept b

New convention: we change $\{0,1\}$ to $\{-1, +1\}$.

Soft-margin Support Vector Machine (SVM)

$$\min_{\boldsymbol{w},b} \frac{1}{2} \|\boldsymbol{w}\|_2^2 + \frac{C}{n} \sum_{i=1}^{n} \max\{0, 1 - y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b)\}$$

We can solve SVM like generalized linear models☺ How to deal with nonlinearity?

Outline

- Linear Classification
  - Logistic Regression
  - Softmax Regression
- **Support Vector Machine (SVM)**
  - Soft-SVM
  - **Kernel Soft-SVM**
- Decision Tree
- Random Forest

Recap: Basis Function

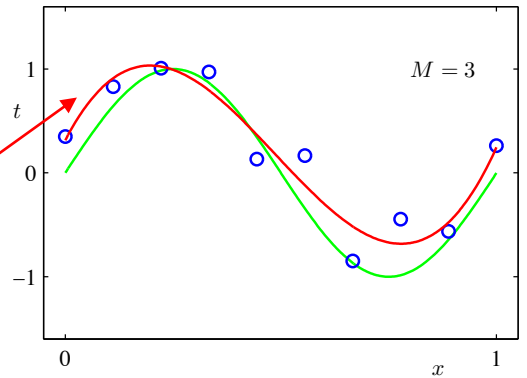
- Feature map: $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathcal{X} \xrightarrow{\Phi} \boldsymbol{z} = (z_1, \ldots, z_{\tilde{d}}) \in \mathcal{Z}$

$$\boldsymbol{z} = \Phi(\boldsymbol{x})$$

- Each $z_j = \phi_j(\boldsymbol{x})$ depends on some nonlinear transform ϕ_j.
- $\{\phi_j\}_{0 \le j \le \tilde{d}}$ is called basis functions.
- Polynomial basis functions

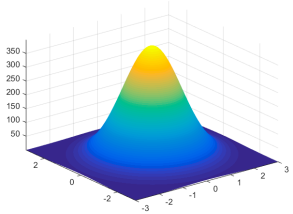
  1-D vector: $\boldsymbol{z}' = (1, x_1, x_1^2, x_1^3)$

  2-D vector: $\boldsymbol{z}' = (1, x_2, x_1 x_2, x_1^2)$

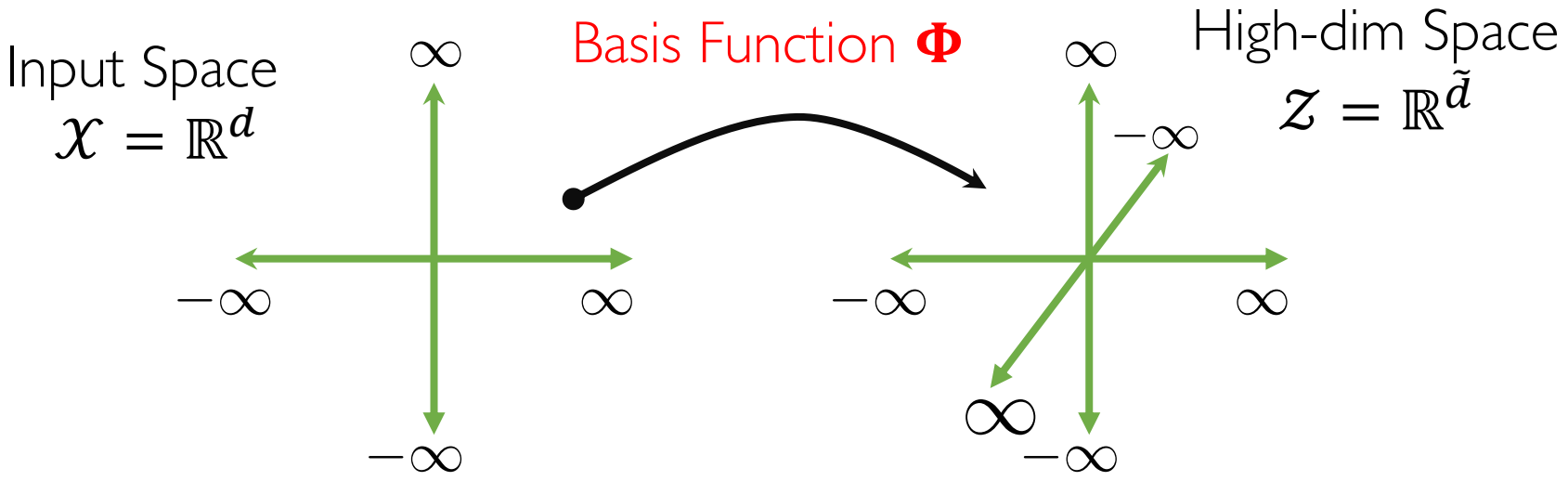


- Radial basis functions (RBF)

$$\phi_j(\boldsymbol{x}) = \left\{ \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{\mu}_j||_2^2}{2\sigma^2} \right) : j = 1, \ldots, \tilde{d} \right\}$$



Kernel Function



- $k(\cdot,\cdot): \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called kernel function induced by basis function $\boldsymbol{\Phi}$ if it can be represented as:

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{\Phi}(\boldsymbol{x}_1) \cdot \boldsymbol{\Phi}(\boldsymbol{x}_2)$$

- $k(\boldsymbol{x}_1, \boldsymbol{x}_2)$ usually enjoys much more efficient computation (if $n \ll d$) than explicitly computing basis functions $\boldsymbol{\Phi}(\boldsymbol{x}_1)$ and $\boldsymbol{\Phi}(\boldsymbol{x}_2)$.

Polynomial Kernel

- For $p = 2, d = 2$:

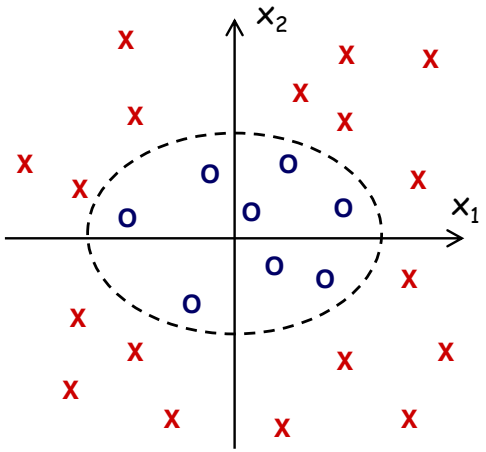
$$\boldsymbol{\Phi}: \mathbb{R}^2 \to \mathbb{R}^3, \qquad \boldsymbol{x} = (x_1, x_2) \to \boldsymbol{\Phi}(\boldsymbol{x}) = \left(x_1^2, x_2^2, \sqrt{2}x_1x_2\right)$$

- Kernel function induced by these polynomial basis functions:

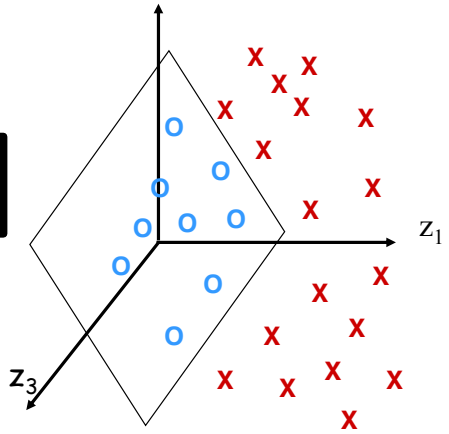
$$\boldsymbol{\Phi}(\boldsymbol{x}_1) \cdot \boldsymbol{\Phi}(\boldsymbol{x}_2) = \left(x_{11}^2, x_{12}^2, \sqrt{2}x_{11}x_{12}\right) \cdot \left(x_{21}^2, x_{22}^2, \sqrt{2}x_{21}x_{22}\right)$$
$$= (x_{11}x_{21} + x_{12}x_{22})^2 = (\boldsymbol{x}_1 \cdot \boldsymbol{x}_2)^2 = k(\boldsymbol{x}_1, \boldsymbol{x}_2)$$

Efficient Computation: $O(d)$

Original Space $\mathcal{X}$



$\boldsymbol{\Phi}: \mathcal{X} \to \mathcal{Z}$



Feature Space $\mathcal{Z}$

Kernel Functions on Vector Data

- Commonly used kernel functions for vector data:

Linear	$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{x}_1 \cdot \boldsymbol{x}_2$
Polynomial	$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1 \cdot \boldsymbol{x}_2)^d$
Polynomial (with constant terms)	$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1 \cdot \boldsymbol{x}_2 + 1)^d$
RBF (Gaussian kernel)	$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp\left(-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}{2\sigma^2}\right)$ sigma增加，学到的函数越光滑
Hyperbolic Tangent	$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \tanh(\beta \boldsymbol{x}_1 \cdot \boldsymbol{x}_2 + c)$

- From kernel functions k_1, k_2, we can construct new kernel functions:
  - $k'(\boldsymbol{x}_1, \boldsymbol{x}_2) = k_1 \otimes k_2(\boldsymbol{x}_1, \boldsymbol{x}_2) = k_1(\boldsymbol{x}_1, \boldsymbol{x}_2) k_2(\boldsymbol{x}_1, \boldsymbol{x}_2)$
  - For any function $g: \mathcal{X} \to \mathbb{R}$, $k'(\boldsymbol{x}_1, \boldsymbol{x}_2) = g(\boldsymbol{x}_1) k_1(\boldsymbol{x}_1, \boldsymbol{x}_2) g(\boldsymbol{x}_2)$

Kernel Soft-SVM

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \frac{C}{n}\sum_{i=1}^{n} \max\{0, 1 - y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b)\}$$

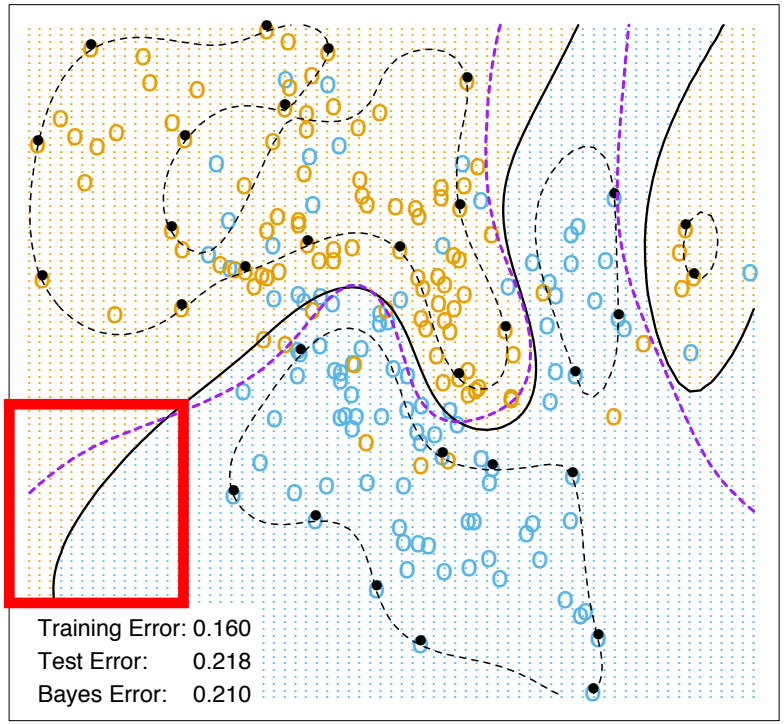
- Apply Representer theorem, we can kernelize Soft-SVM as follows:

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{\alpha} + \frac{C}{n}\sum_{i=1}^{n} \max\left\{0, 1 - y_i\sum_{j=1}^{n}\alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)\right\}$$

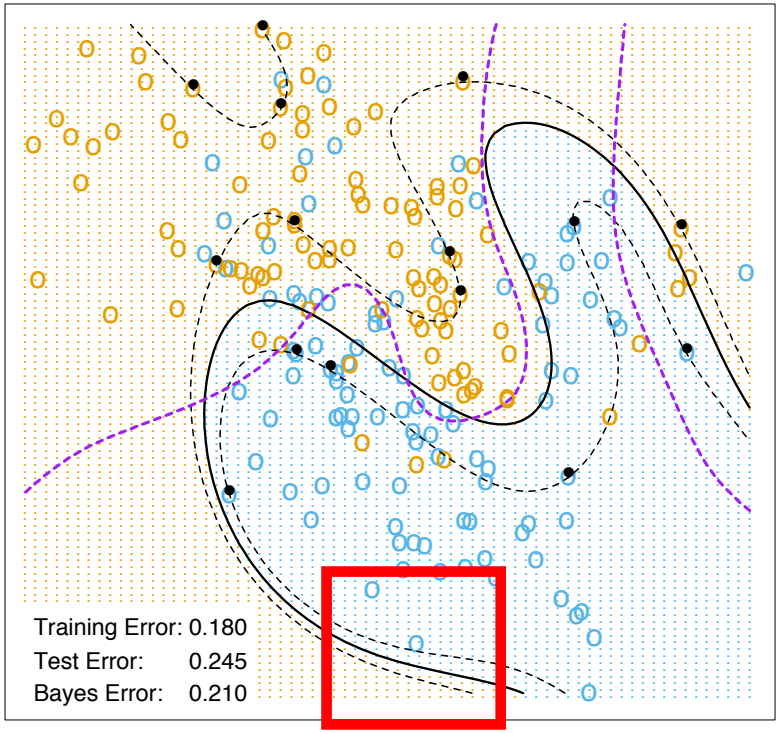
  - α_j is the weight of each reference point $\boldsymbol{x}_j$ to the prediction of $\boldsymbol{x}_i$
  - It is actually a Primal Form with kernel functions.
- We can also kernelize all learning algorithms:
  - Linear regression, logistic regression, regularized linear regression…

Kernel Soft-SVM

SVM - Radial Kernel in Feature Space



SVM - Degree-4 Polynomial in Feature Space



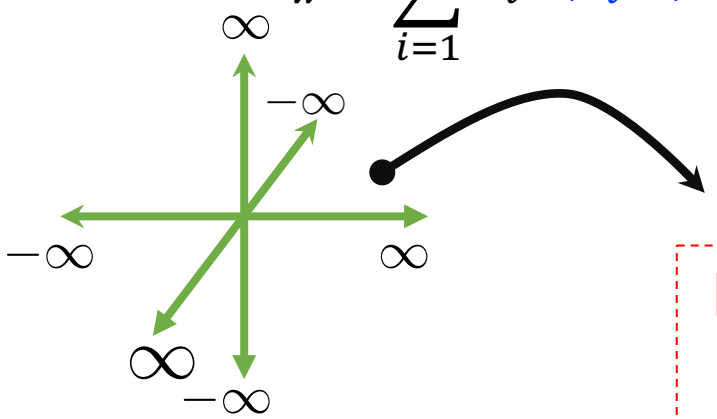
- Select kernels and its parameters (e.g. σ) is crucial: cross-validation.
- Solving the dual form requires big computation complexity: $O(n^{2.65})$!

Support Vector Machine (SVM)

$$h_{\boldsymbol{w}} = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

> 0 → Class $+1$

< 0 → Class -1



Instance Space

$\mathcal{X} = \mathbb{R}^d$

Discrete Label Space

$\mathcal{Y} = \{-1, 1\}$

Non-linearly Non-separable

New convention: we explicitly write the intercept b

New convention: we change $\{0,1\}$ to $\{-1, +1\}$.

Kernel Soft-SVM (most powerful variant)

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha} + \frac{C}{n} \sum_{i=1}^{n} \max\left\{0, 1 - y_i \sum_{j=1}^{n} \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)\right\}$$

We have gone through SVMs with What, How, and Why.

How to choose between LM or SVM?

Outline

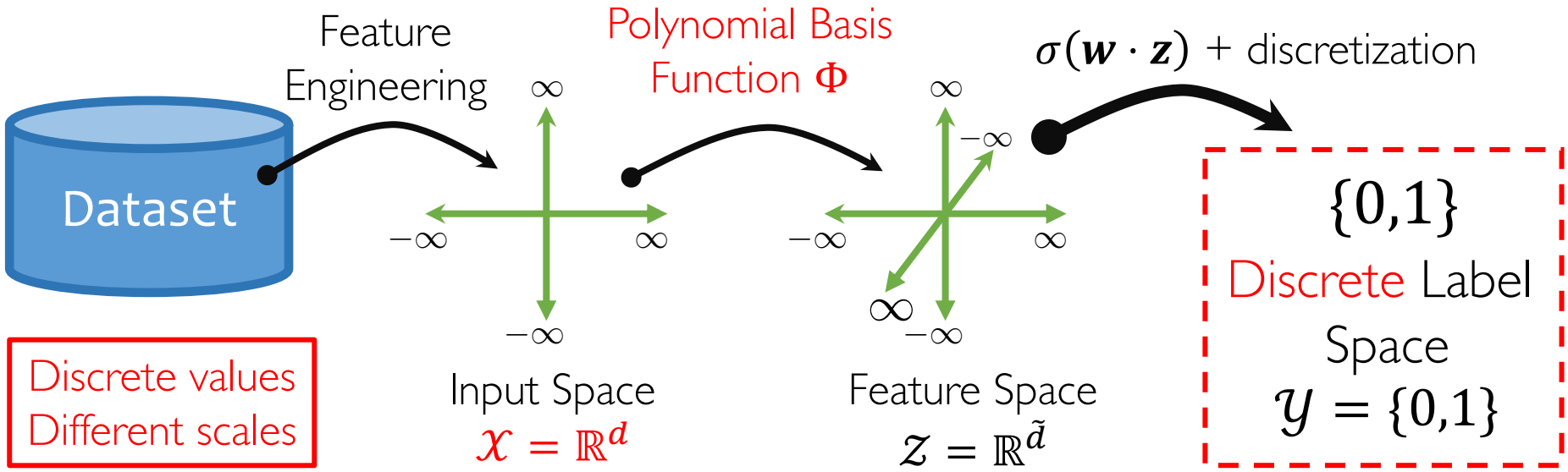
- Linear Classification
  - Logistic Regression
  - Softmax Regression
- Support Vector Machine (SVM)
  - Soft-SVM
  - Kernel Soft-SVM
- **Decision Tree**
- Random Forest

Linear Model

2-class Classification:

Hypothesis Space

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{\Phi}(\boldsymbol{x})$$



Day	Outlook	Temperature	Humidity	Wind	EnjoyTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes

How to do with feature heterogeneity and achieve model interpretability?

Human Classifier

- How do people solve this problem?

Do we combine features as in LM?

green, round, no leaf, sweet, … — Apple

red, round, leaf, sweet, … — Apple

yellow, round, leaf, sour, … — Lemon

yellow, curved, no leaf, sweet, … — Banana

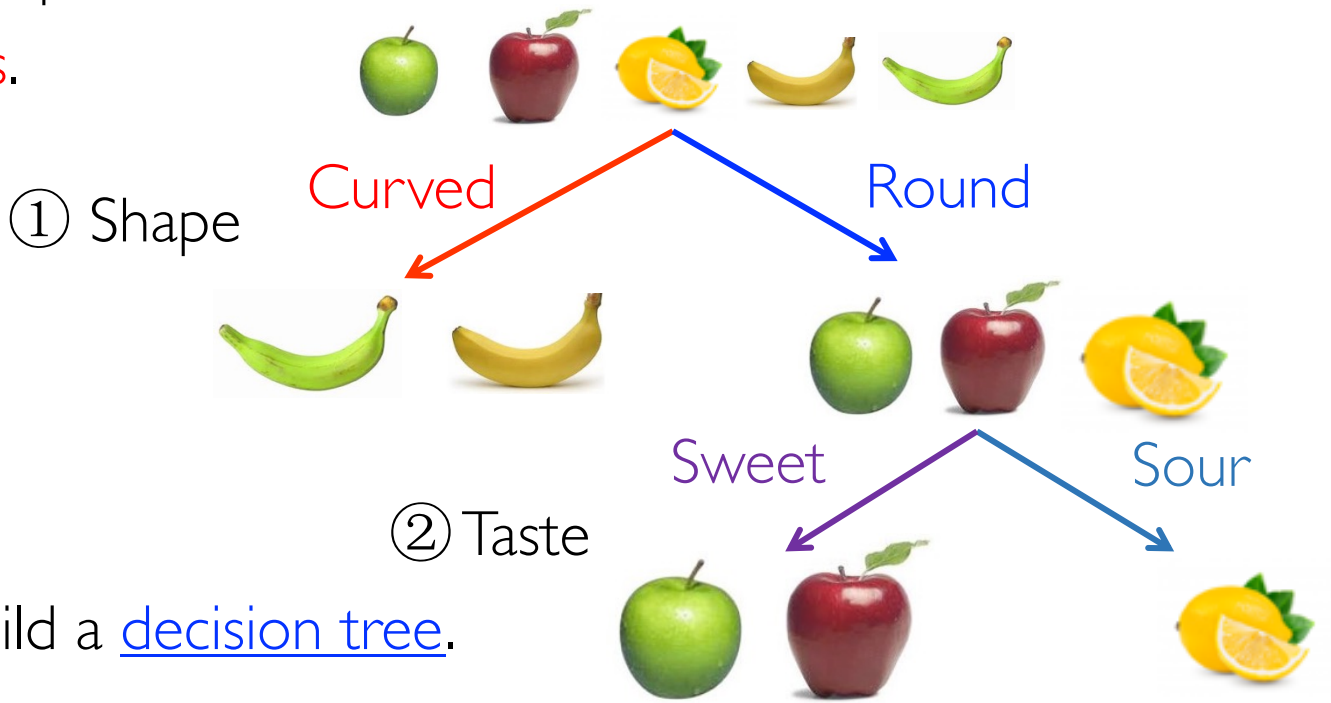
green, curved, no leaf, sweet, … — Banana

Feature

Label

Human Classifier

- We find a most useful feature, such as *shape*: curved or round.
  - Split the dataset: If the fruit is curved, then it is a banana.
- After the first split, select next best feature until each node contains only one class.



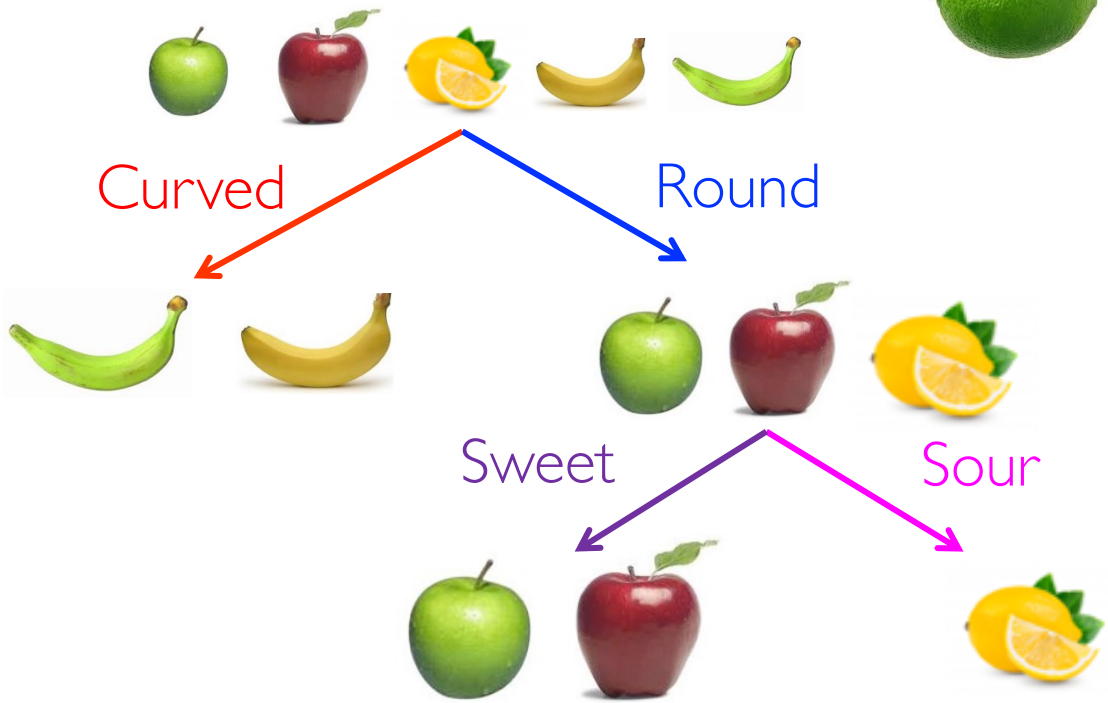
- At last, we build a decision tree.

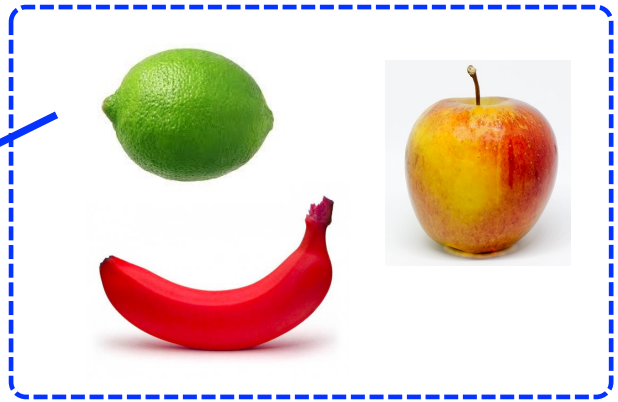
Decision Tree

Interpretability ✓

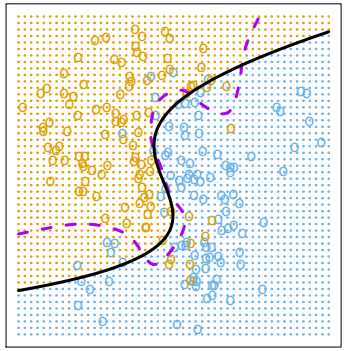
Generalize to unseen data ✓

- During test:
  - Why do you think this fruit is lemon?
  - Because it is round and sour.





Compared with LM:

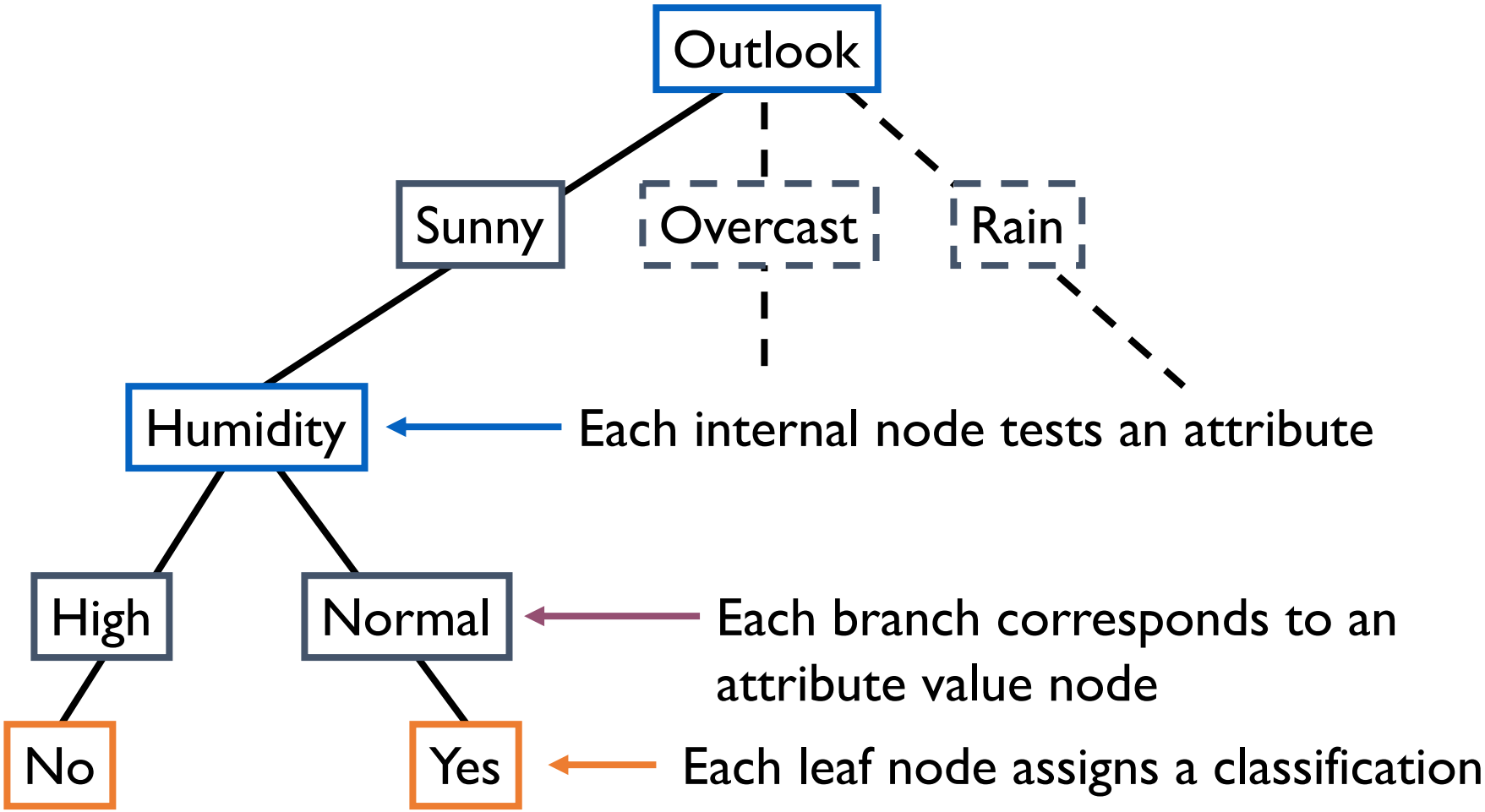


Decision Tree

- When we face many features, it is hard for us to build tree by hand.
- Can we teach machine to find it? What are the difficulties?
  - How to find the most useful feature on each node?
  - When should we stop growing the tree?
  - What if some features are missing or continuous-valued?

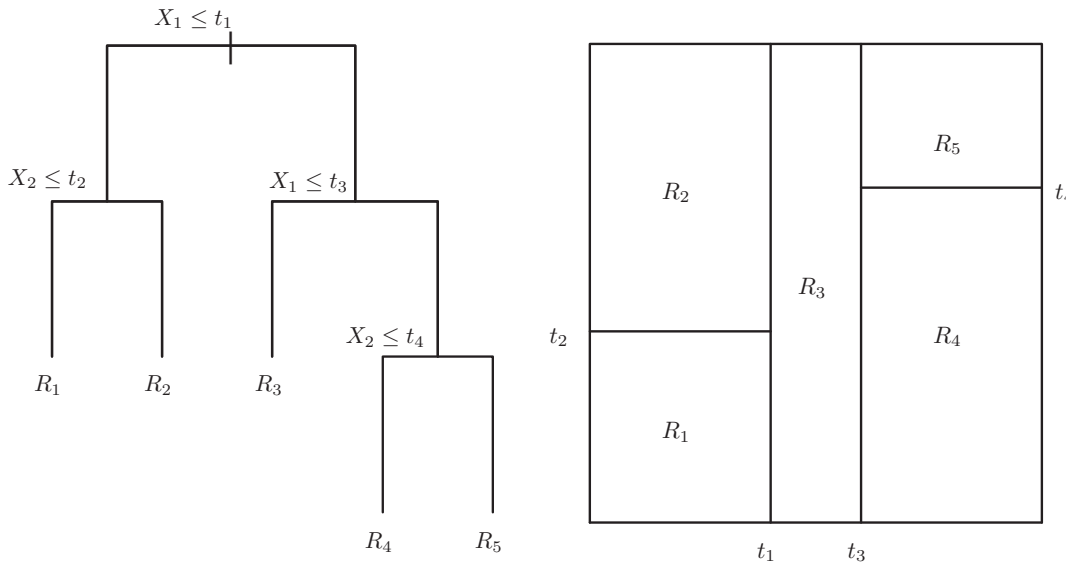
Day	Outlook	Temperature	Humidity	Wind	EnjoyTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes

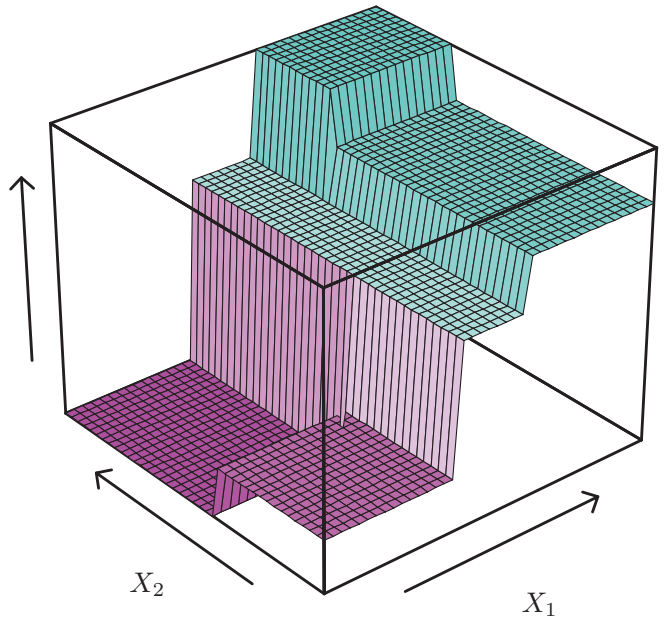
Decision Tree



Decision Tree

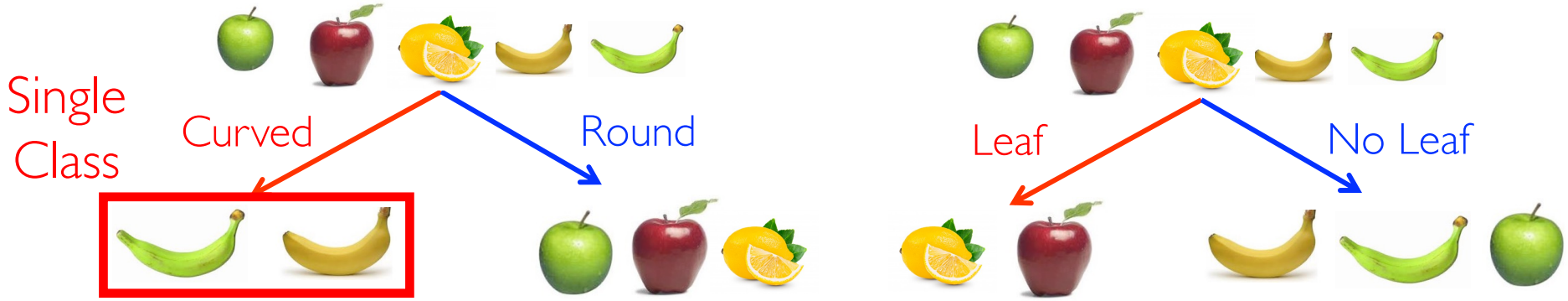
- **Hypothesis space** of decision tree:
  - Decision trees divide the feature space into axis-parallel rectangles.
  - Each rectangular region is labeled with a specific label.
  - What structure is hard for decision tree?



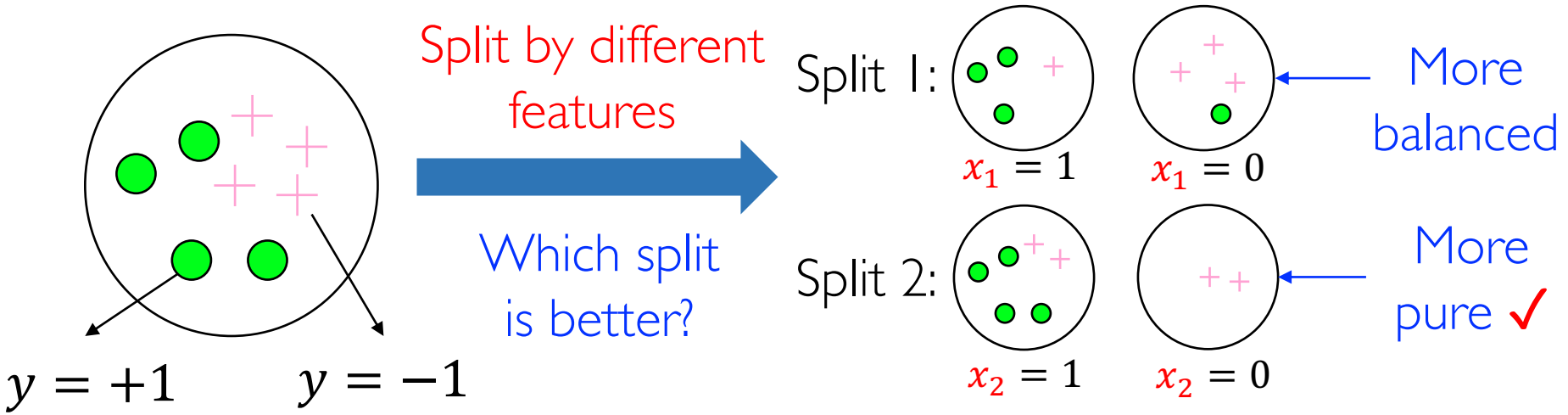


Node Splitting

- Which split is better?



- As example, we visualize splits in a binary classification problem:

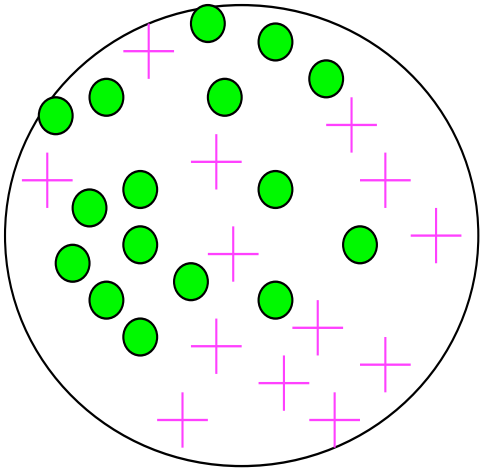


How to Measure a Node

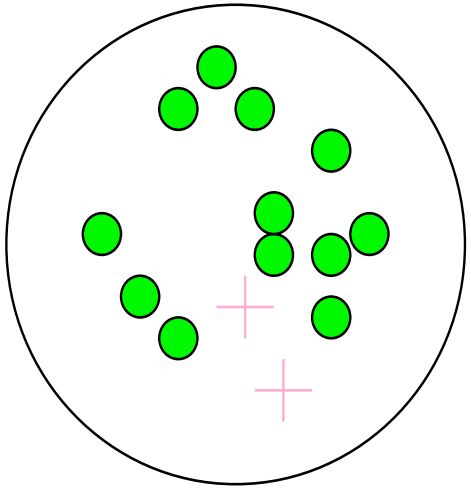
- We want pure leaf nodes, as close to a single class as possible:
  - Lead to a lower classification error.
  - The classification error is $\min(\frac{|\mathcal{C}_1|}{|\mathcal{D}|}, \frac{|\mathcal{C}_2|}{|\mathcal{D}|})$.

Number of samples in each class

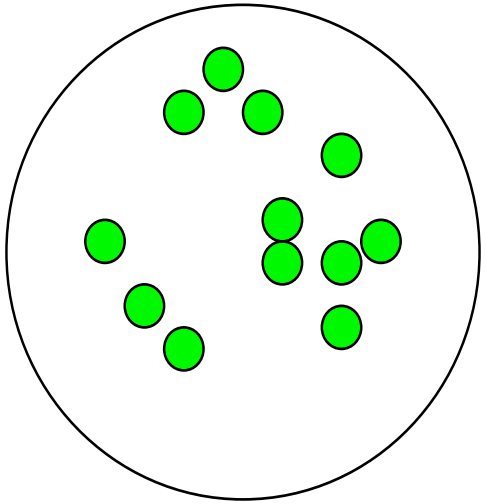
- We'll choose the splitting feature that minimizes impurity measure.



Very impure group
Error: $13/29$



Less impure
Error: $2/14$



Minimum impurity
Error: 0

Node Impurity Measures

- Three standard node impurity measures:
  - Misclassification error:

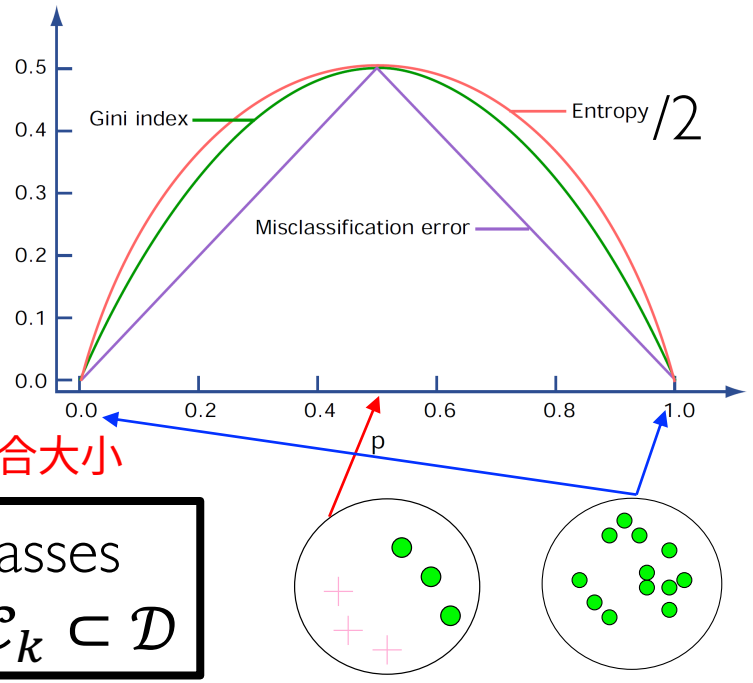
$$\mathrm{Err}(\mathcal{D}) = 1 - \max_{1 \le k \le K} \left(\frac{|\mathcal{C}_k|}{|\mathcal{D}|} \right)$$

  - Entropy (used in ID3 and C4.5):

$$H(\mathcal{D}) = -\sum_{k=1}^{K} \frac{|\mathcal{C}_k|}{|\mathcal{D}|} \log \frac{|\mathcal{C}_k|}{|\mathcal{D}|}$$

  - Gini index (used in CART):

$$\mathrm{Gini}(\mathcal{D}) = 1 - \sum_{k=1}^{K} \left(\frac{|\mathcal{C}_k|}{|\mathcal{D}|} \right)^2$$



D：总集合大小

K #classes
Each is $\mathcal{C}_k \subset \mathcal{D}$

Max: $\frac{|\mathcal{C}_k|}{|\mathcal{D}|} = 0.5$

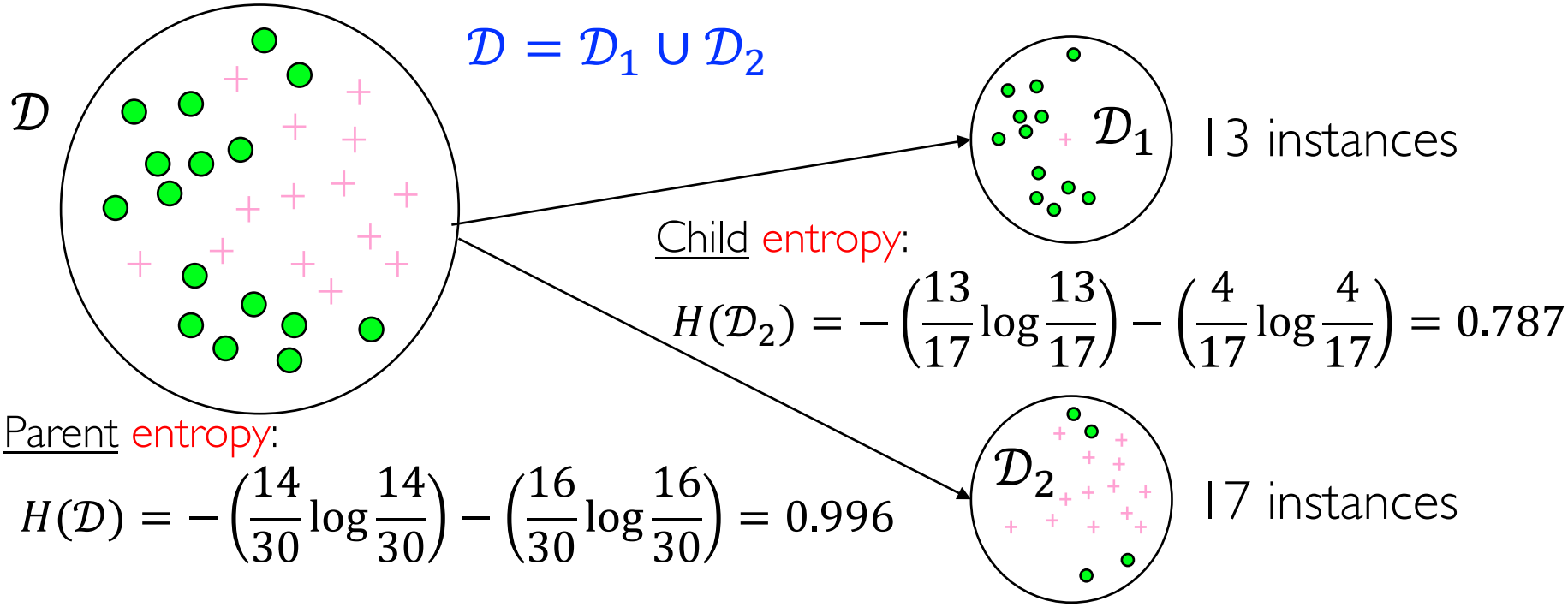
Min: $\frac{|\mathcal{C}_k|}{|\mathcal{D}|} = 0$

$(0 \log 0 = 0)$

How to Measure a Split

Entire dataset (30 instances)

$$H(\mathcal{D}_1) = -\left(\frac{1}{13}\log\frac{1}{13}\right) - \left(\frac{12}{13}\log\frac{12}{13}\right) = 0.391$$



- We want to measure this split – how this split minimizes entropy.
  - How about $H(\mathcal{D}) - (H(\mathcal{D}_1) + H(\mathcal{D}_2))/2$?
  - But the instance number on each node is different → Weighting.

Information Gain (IG)

- A good split gives minimal weighted average of node impurities:

$$\frac{|\mathcal{D}_1|}{|\mathcal{D}|} H(\mathcal{D}_1) + \frac{|\mathcal{D}_2|}{|\mathcal{D}|} H(\mathcal{D}_2)$$

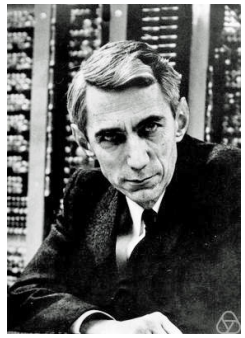
- Equivalent to maximizing the Information Gain (IG):

Child nodes

Parent node

$$H(\mathcal{D}_1 \cup \mathcal{D}_2) - \frac{|\mathcal{D}_1|}{|\mathcal{D}|} H(\mathcal{D}_1) - \frac{|\mathcal{D}_2|}{|\mathcal{D}|} H(\mathcal{D}_2)$$

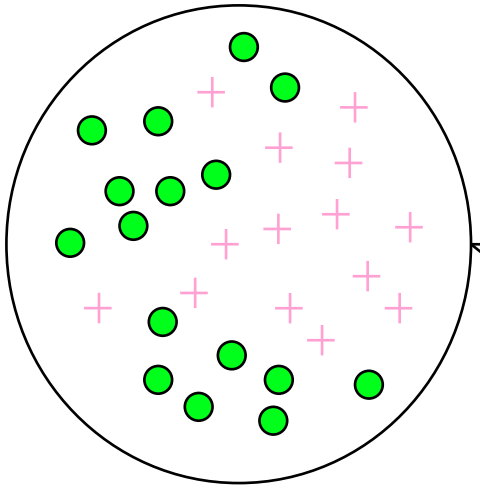
- In information theory, entropy is the expected number of bits needed to encode a randomly drawn value of X.
- Larger entropy, less information. That's why we call it IG. https://en.wikipedia.org/wiki/Entropy_(information_theory)



Claude Shannon

Calculating Information Gain (IG)

Entire population (30 instances)

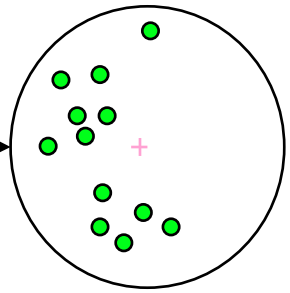


Parent entropy:

$$-\left(\frac{14}{30}\log\frac{14}{30}\right)-\left(\frac{16}{30}\log\frac{16}{30}\right)=0.996$$

Child entropy:

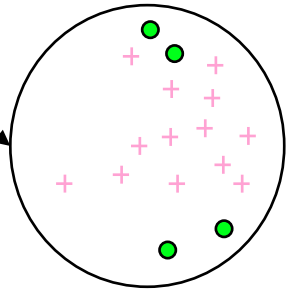
$$-\left(\frac{1}{13}\log\frac{1}{13}\right)-\left(\frac{12}{13}\log\frac{12}{13}\right)=0.391$$



13 instances

Child entropy:

$$-\left(\frac{13}{17}\log\frac{13}{17}\right)-\left(\frac{4}{17}\log\frac{4}{17}\right)=0.787$$

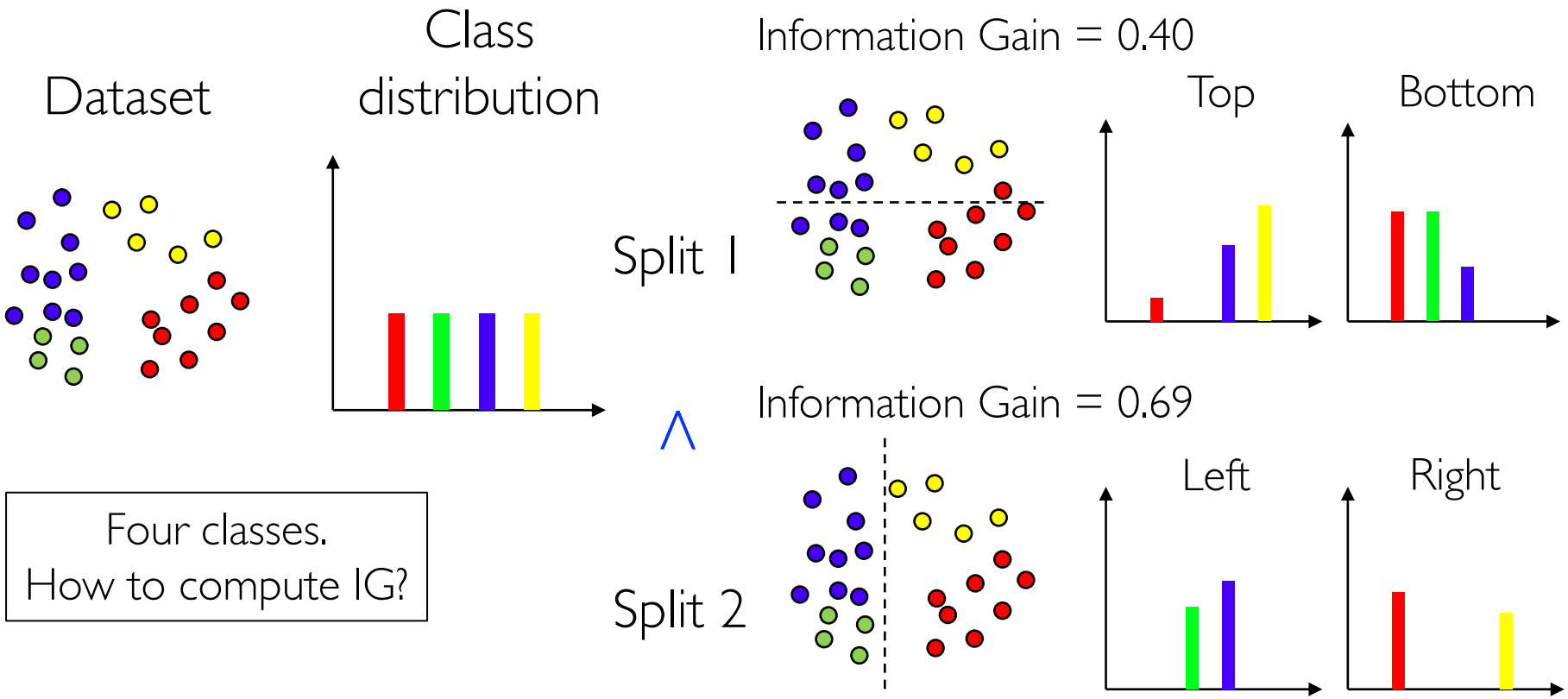


17 instances

Information Gain (IG):

$$0.996-\frac{17}{30}\cdot 0.787-\frac{13}{30}\cdot 0.391=0.38$$

Split Search



- Compute IG for all splits induced by every feature.
  - If IGs of all features are small (e.g. $< \epsilon$): stop.
  - Else: find the feature that maximizes the Information Gain (IG).

ID3 Algorithm

Class label

ID3(*Examples*, *Target_attribute*, *Attributes*)

- create a *Root* node for the tree; assign all *Examples* to *Root*;

Stop Criteria

- if all *Examples* are positive, return the single-node tree *Root*, with label=+;
- if all *Examples* are negative, return the single-node tree *Root*, with label=−;
- if *Attributes* is empty, return the single-node tree *Root*, with label = the most common value of *Target_attribute* in *Examples*;
- otherwise // **Main loop:**

  $A \leftarrow$ the attribute from *Attributes* that best* classifies *Examples*;
  the decision attribute for *Root* $\leftarrow A$;
  for each possible value v_i of A

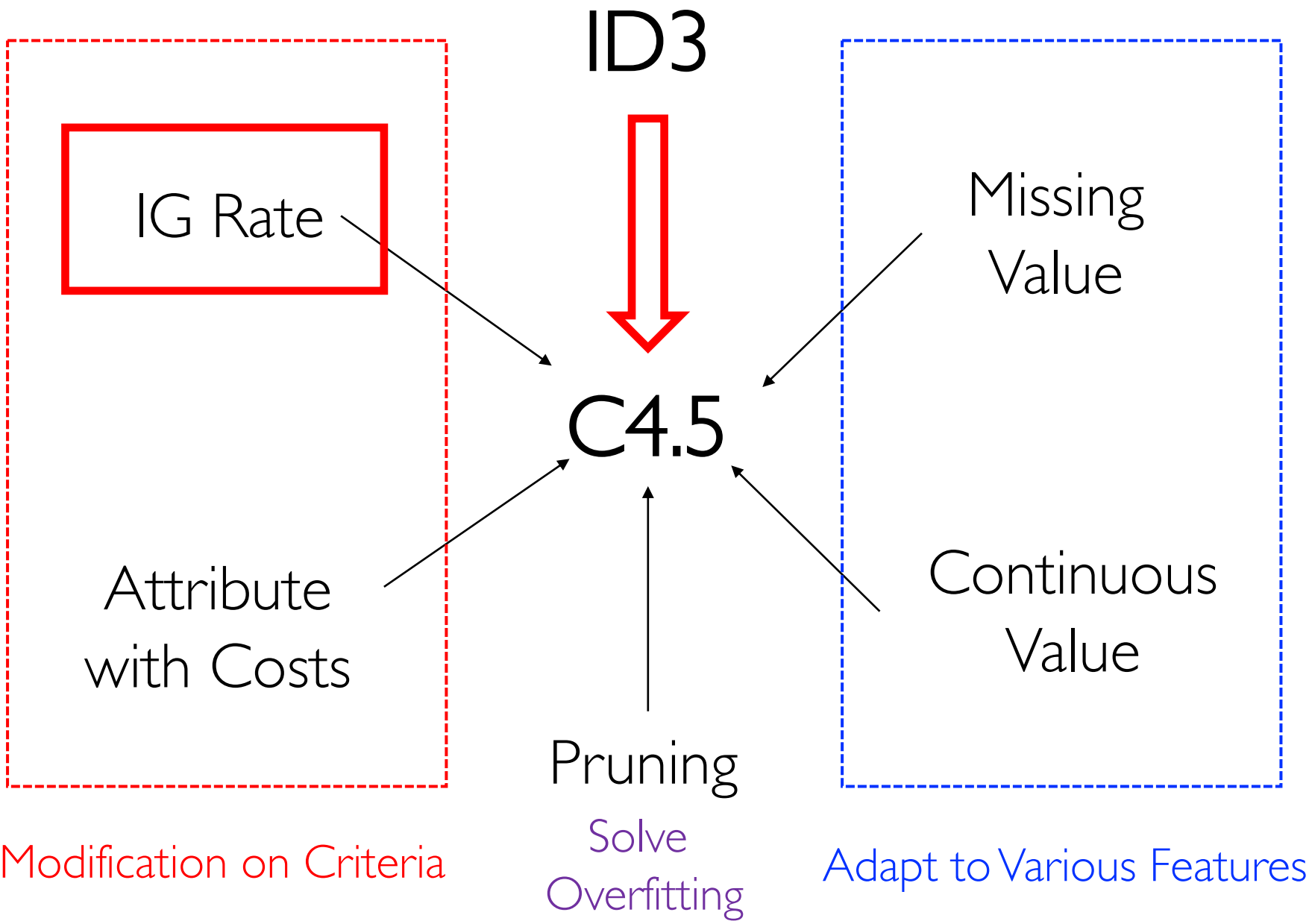
  $O(dn)$ in each layer

  add a new tree branch below *Root*, corresponding to the test $A = v_i$;
  let $Examples_{v_i}$ be the subset of *Examples* that have the value v_i for A;
  if $Examples_{v_i}$ is empty
  below this new branch add a leaf node with label = the most common value of *Target_attribute* in *Examples*;
  else
  below this new branch add the subtree
  ID3($Examples_{v_i}$, *Target_attribute*, *Attributes*$\backslash\{A\}$);

  depth $\min(d, \log n)$

- return *Root*;

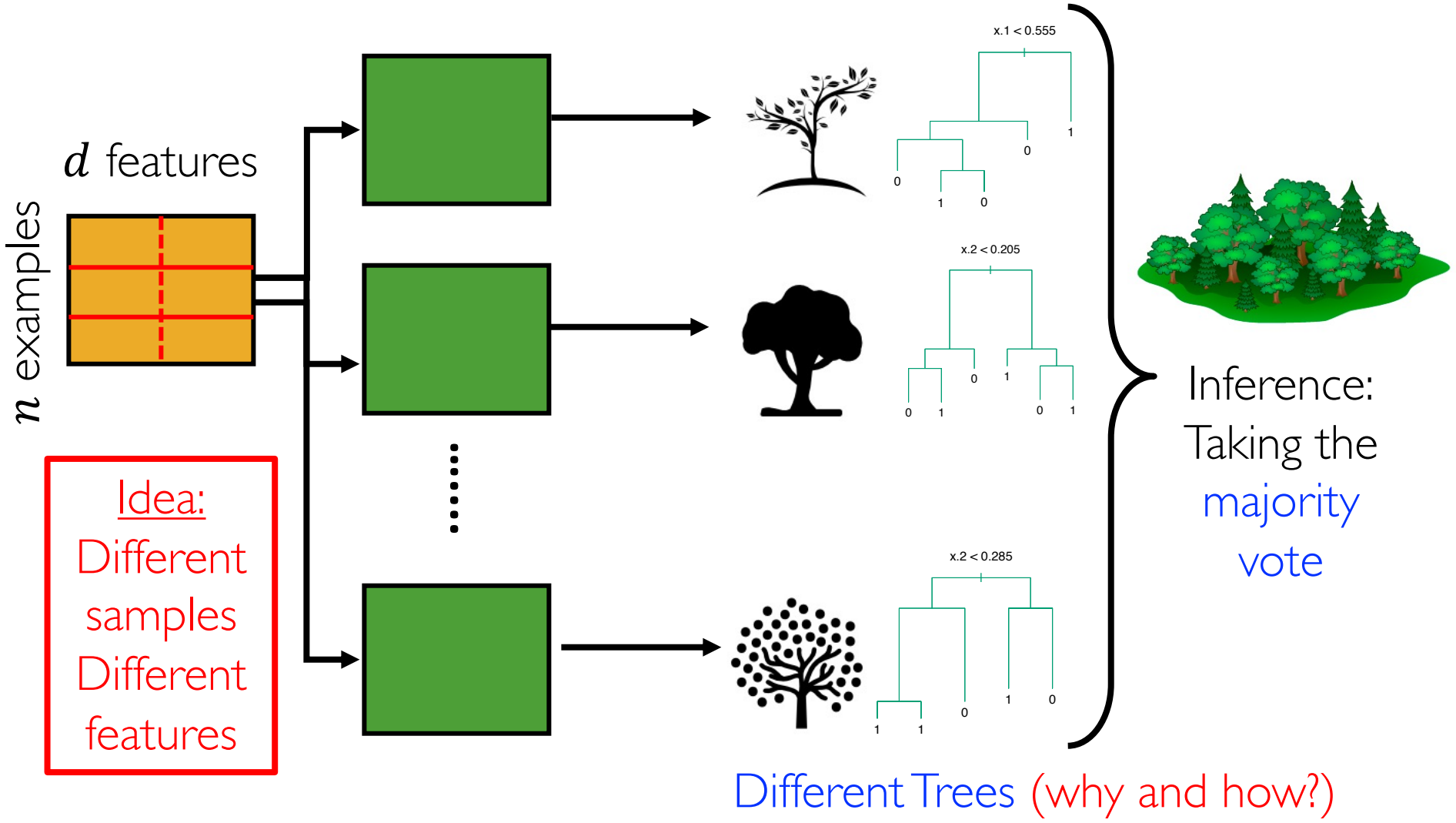
* The *best* attribute is the one with the highest *information gain*.


ID3
IG Rate
Missing Value
C4.5
Attribute with Costs
Continuous Value
Pruning
Solve Overfitting
Modification on Criteria
Adapt to Various Features

Outline

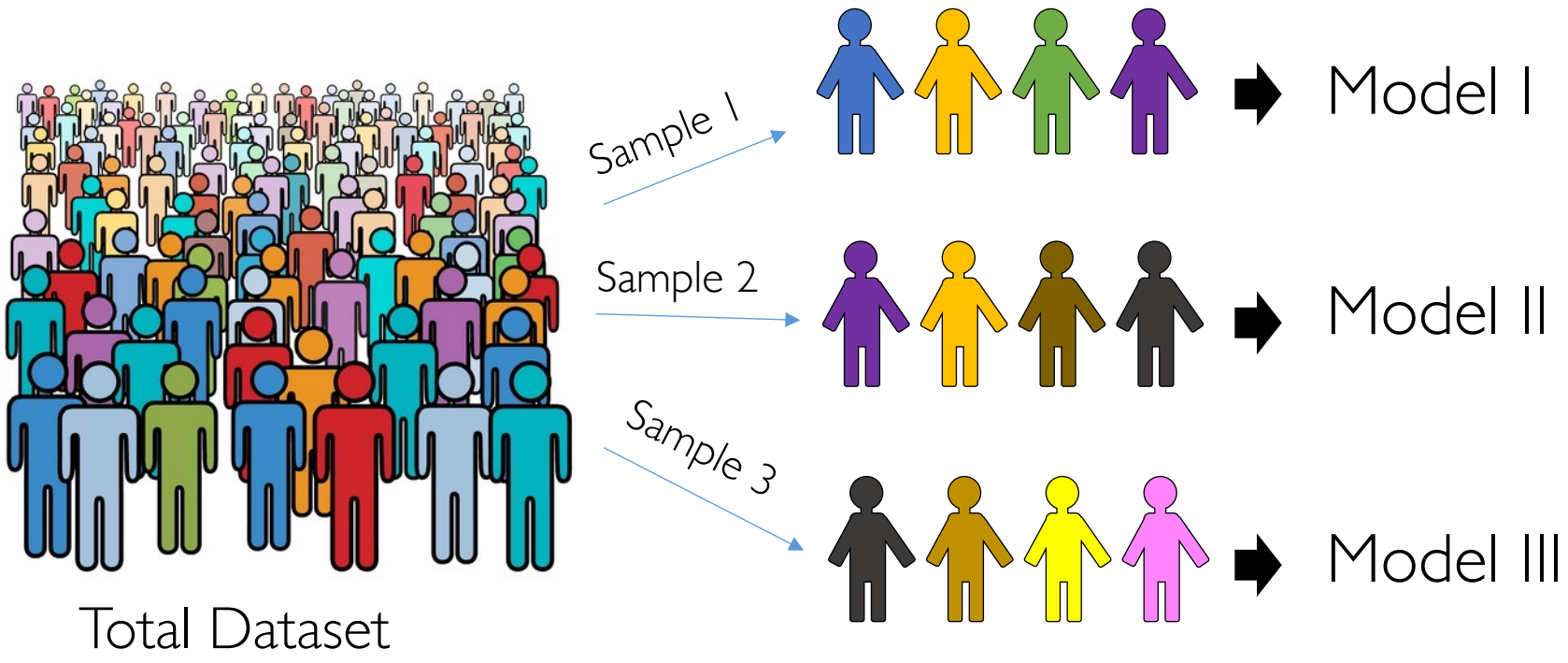
- Linear Classification
  - Logistic Regression
  - Softmax Regression
- Support Vector Machine (SVM)
  - Soft-SVM
  - Kernel Soft-SVM
- Decision Tree
- **Random Forest**

Random Forest



Bootstrap Sample

- Bootstrap (自助法): randomly draw datasets with replacement from the training data, with the same sample size as the original training set



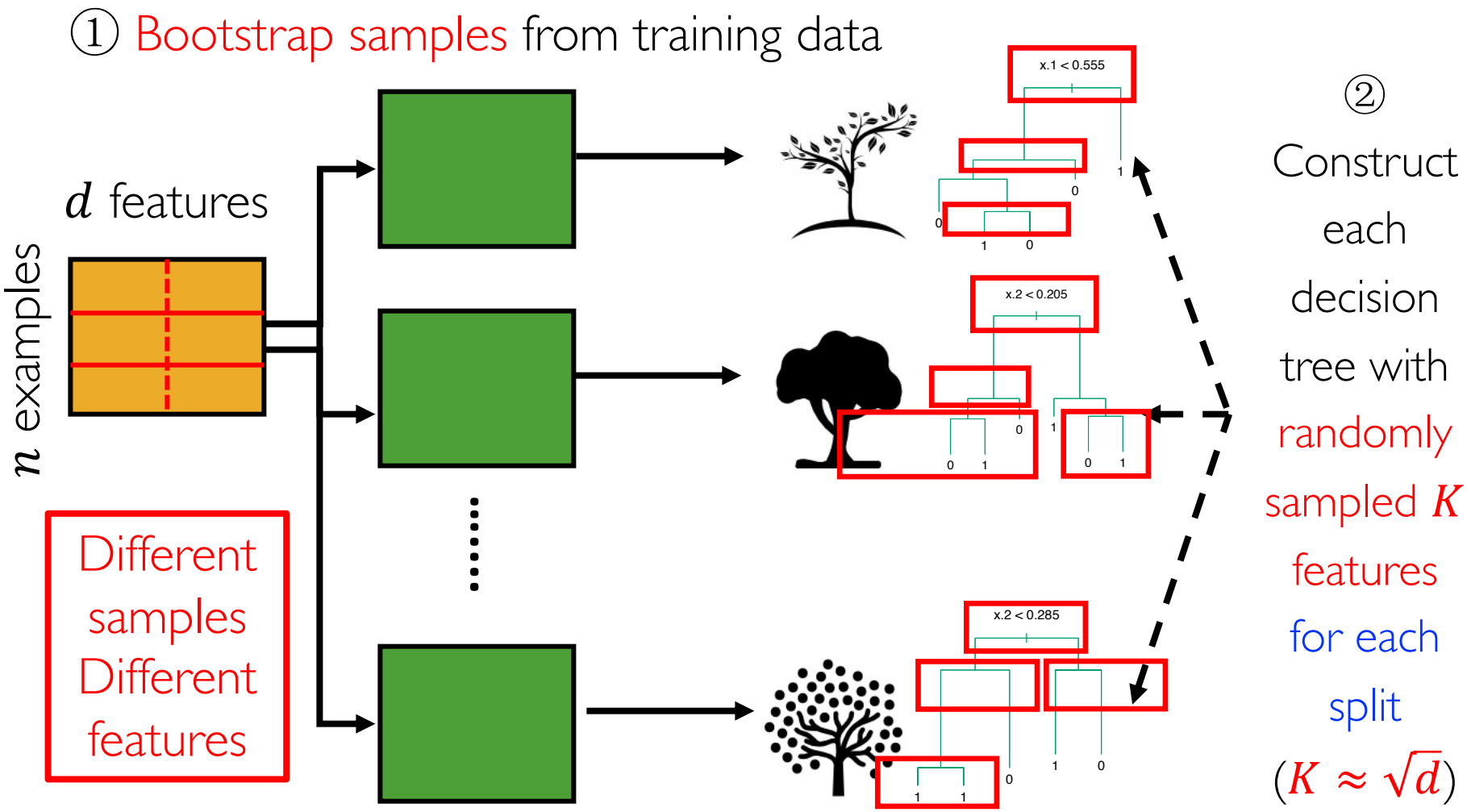
Bootstrap Sample

- **Definition**: A bootstrap sample from $\mathcal{D}_n$ is a sample of size n drawn with replacement from $\mathcal{D}_n$.
- In a bootstrap sample, some elements of $\mathcal{D}_n$
  - will show up multiple times.
  - some will not show up at all. Will this happen?
- Each X_i has a probability $\left(1 - \frac{1}{n}\right)^n$ of being not selected.
- Recall from calculus that for large n,

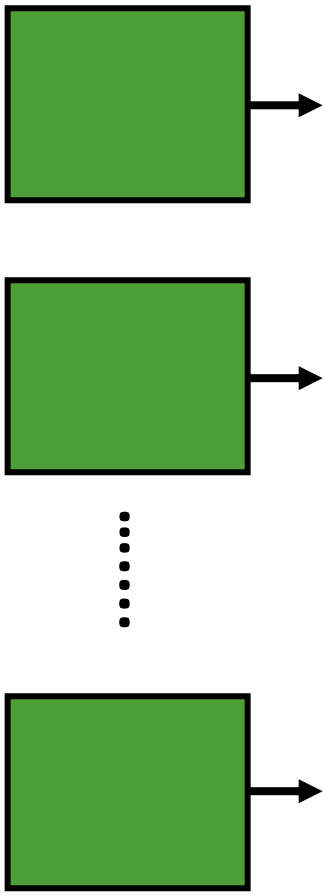
$$\left(1 - \frac{1}{n}\right)^n \approx \frac{1}{e} \approx 0.368.$$

- So we expect $\sim 63.2\%$ of elements of $\mathcal{D}$ will show up at least once.

Random Forest



Breiman Algorithm



Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Feature subset size K.

Process:

1. $N \leftarrow$ create a tree node based on D;
2. **if** *all instances in the same class* **then return** N
3. $\mathcal{F} \leftarrow$ the set of features that can be split further;
4. **if** $\mathcal{F}$ *is empty* **then return** N
5. $\tilde{\mathcal{F}} \leftarrow$ select K features from $\mathcal{F}$ randomly;
6. $N.f \leftarrow$ the feature which has the best split point in $\tilde{\mathcal{F}}$;
7. $N.p \leftarrow$ the best split point on $N.f$;
8. $D_l \leftarrow$ subset of D with values on $N.f$ smaller than $N.p$;
9. $D_r \leftarrow$ subset of D with values on $N.f$ no smaller than $N.p$;
10. $N_l \leftarrow$ call the process with parameters (D_l, K);
11. $N_r \leftarrow$ call the process with parameters (D_r, K);
12. **return** N

Output: A random decision tree

$K \approx \sqrt{d}$

Randomized features

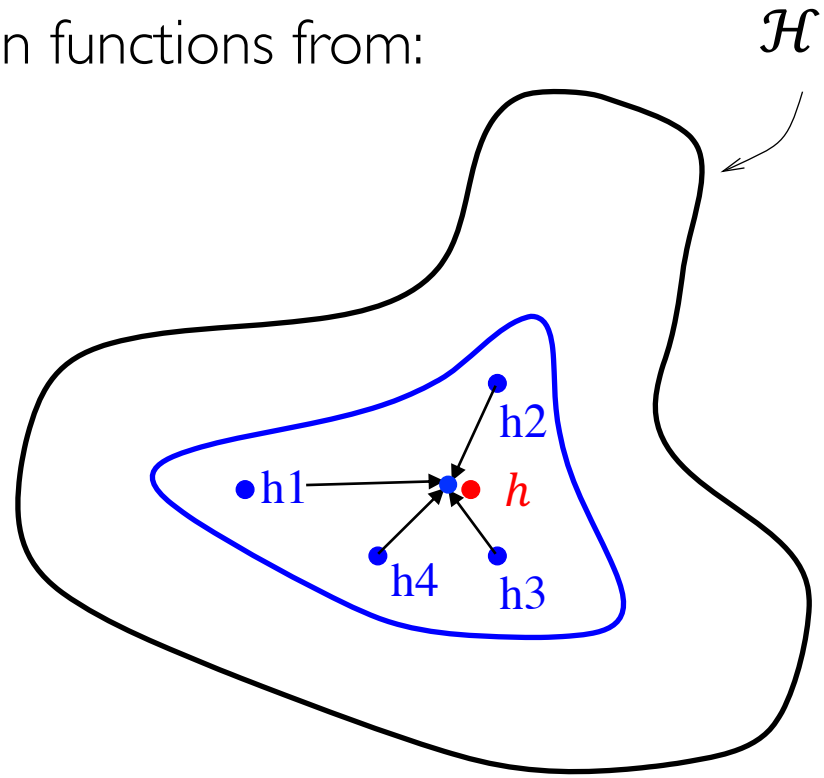
Random forests
L Breiman
Machine learning 45 (1), 5-32

By Leo Breiman

50227 2001

Random Forest

- A usual approach is to build very deep trees
  - Low training error, high generalization error.
- Diversity in individual tree prediction functions from:
  - Bootstrap samples
  - Randomized tree building
- Bagging works better when we are combining a diverse set of prediction functions.
- RF find a way to achieve them.



Thank You

Questions?

Mingsheng Long
mingsheng@tsinghua.edu.cn
http://ise.thss.tsinghua.edu.cn/~mlong
答疑：东主楼11区413室