香港浸會大學
HONG KONG BAPTIST UNIVERSITY

DEPARTMENT OF
COMPUTER SCIENCE
計算機科學系

COMP7015 Artificial Intelligence (S1, 2024-25)

Lecture 5: Basics of Statistical Machine Learning

Instructor: Dr. Kejing Yin (cskjyin@hkbu.edu.hk)

Department of Computer Science
Hong Kong Baptist University

October 4, 2024

Announcements

- Lab 1 on Oct. 5: Solving Problems Using Search
- PA1 is released. *Deadline: 11:59am, Oct. 18*
- No class on Oct. 11 (Chung Yeung Festival)

- In-class Quiz on Oct. 18: you can use a *non-programmable calculator*
  - 6:30 — 7:05 pm Explanation & clarification about the course project
  - 7:15 — 9:15 pm Quiz

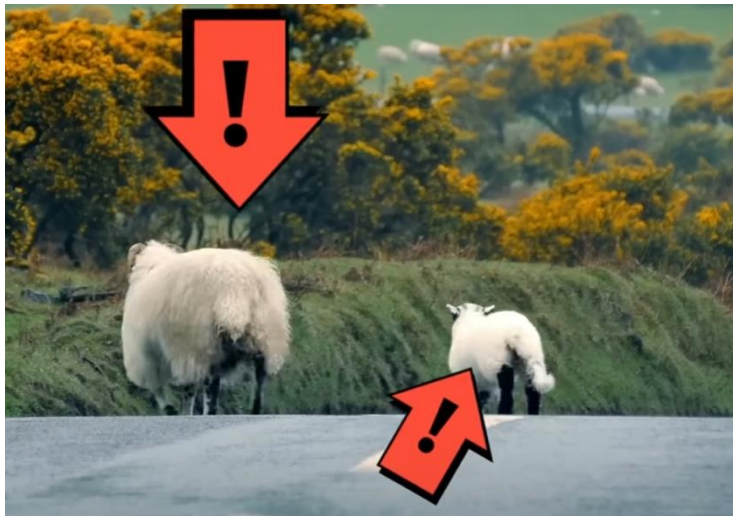
Outline for Today

1. Introduction to Machine Learning
2. Linear Model
3. Generalization and Model Selection
4. Performance Measures
5. Multiclass Classification: Methods and Evalaution
6. Feature Engineering

Introduction to Machine Learning

- Why Machine Learning?
- Machine Learning: Learning from Samples

Example: Detecting Animals for Autonomous Vehicles



List out all rules that specific the physical characteristics of animals?

Not feasible:

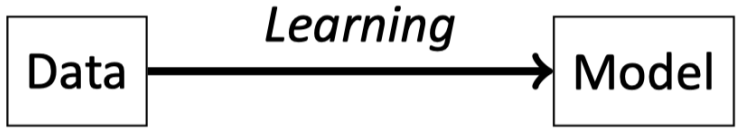
- Not possible to enumerate all rules.
- Cannot map image pixels to the rules.

How Did We Learned to Recognize A Cat?

- We did not start from learning a set of rules that define a cat.
- We started by seeing cats and non-cats, in real life, in cartoon, etc.

vs

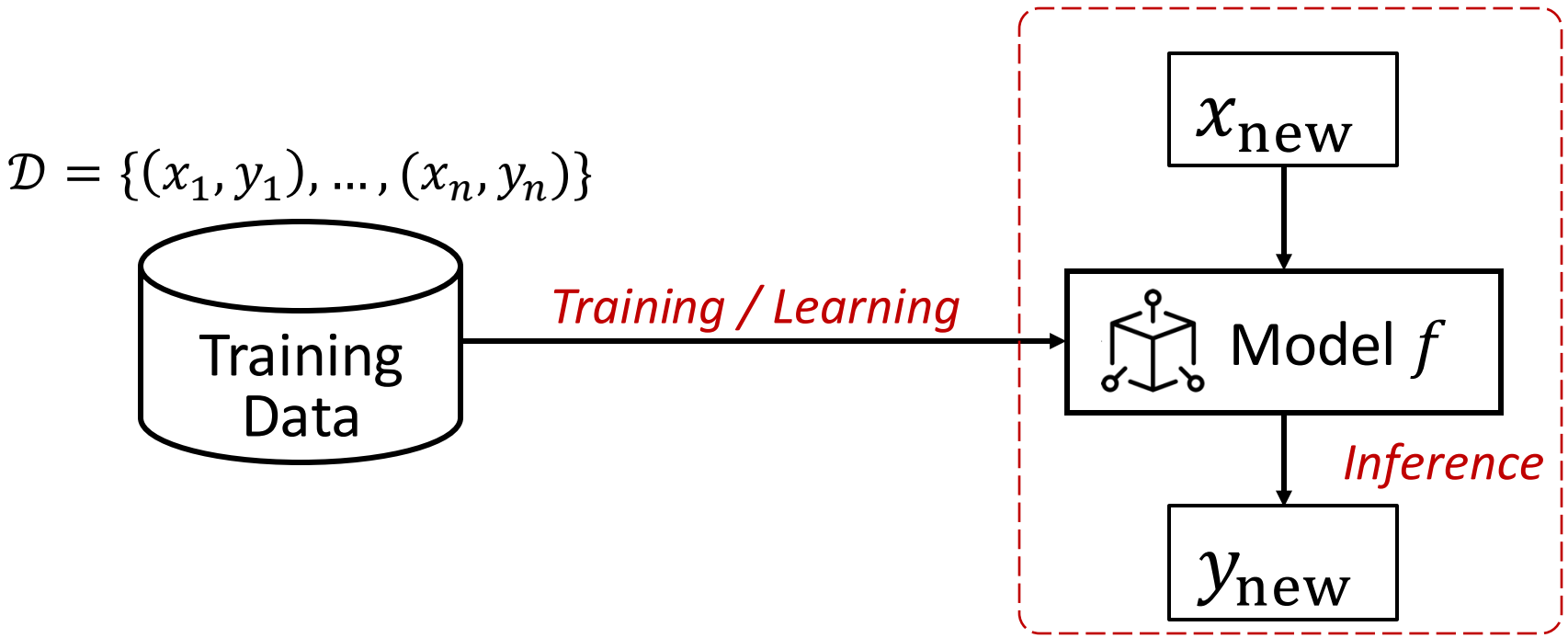
- Machine learning: the same intuitive idea



Types of Machine Learning

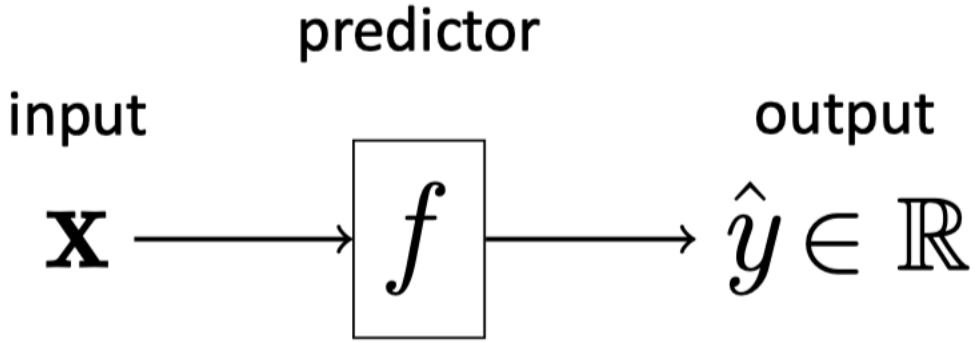
- **Supervised Learning**
  - Training with *labeled* data.
  - Dataset: $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$, where x is the *feature* and y is the *label*.
  - Example: training a cat detector with images of cats and other animals.
- Unsupervised Learning
  - Training with unlabeled data.
  - Dataset: $\mathcal{D} = \{x_1, \dots, x_n\}$, where x is the feature.
  - Example: given a set of images of animals, group images of the same species.
- Semi-supervised Learning
  - A combination of supervised and unsupervised learning.
  - Training with a small amount of labeled data and a large amount of unlabeled data.
  - Dataset: $\mathcal{D} = \{x_1, \dots, x_n\} \cup \{(x_1, y_1), \dots, (x_m, y_m)\}$.
- Reinforcement Learning (will be covered in later lectures)

Abstraction of Supervised Learning Models



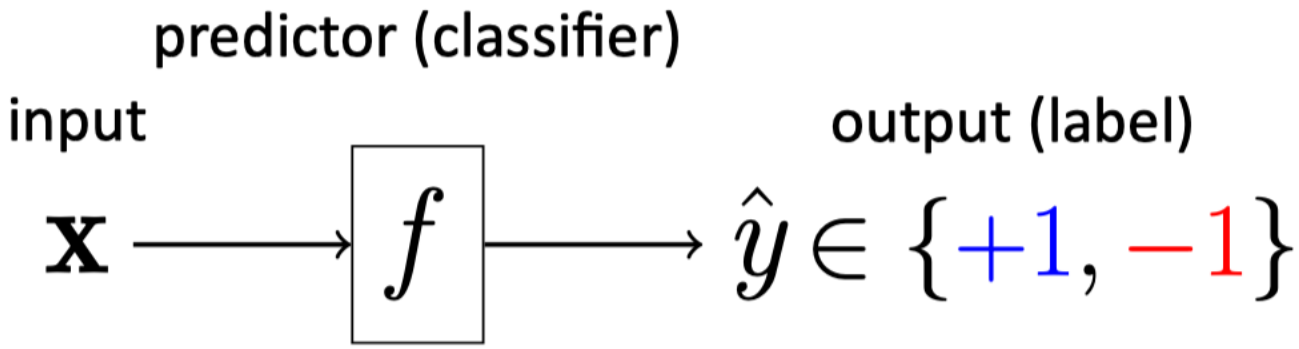
- Training: we want to learn a function (model; predictor) from a collection of training data.
- Testing: with the model, we can perform inference to make predictions for unseen samples.
- The model (predictor) takes some input x and generates some output y .

Supervised Learning: Regression



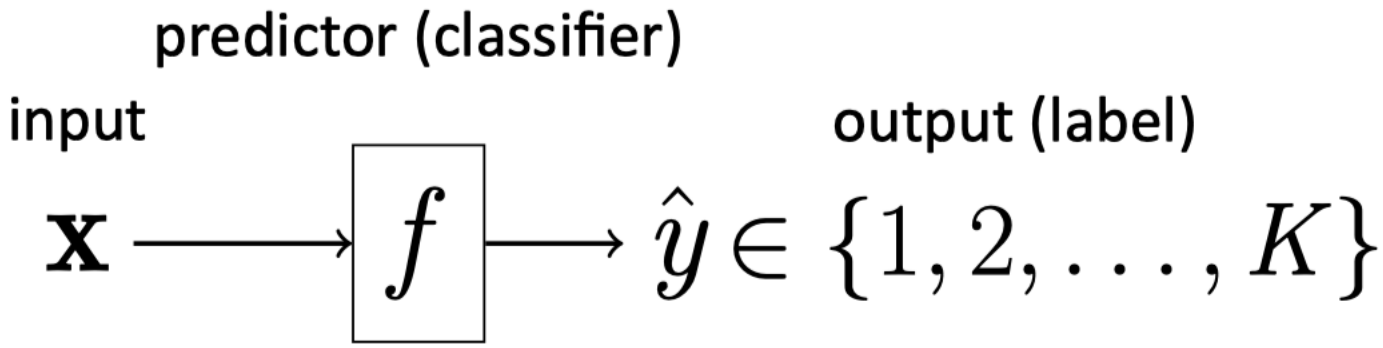
- The output is a real-valued scalar.
- Example: information about houses (location, area, etc.) → Price

Supervised Learning: Binary Classification



- The output is a binary label of positive (+1) or negative (−1 or 0).
- Example:
  - Credit card application: information abut applicants → Approval or Rejection
  - Cat classification: an image → Is a cat or Not a cat
  - COVID-19 diagnosis: CT image → Has COVID-19 or Does not have COVID-19

Supervised Learning: Multiclass Classification



- The output is one of *K* classes (labels).
- Example:
  - Digit recognition: image of handwritten digit → one of {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
  - Animal classification: an image → one of {Cat, Dog, Wolf, …}

Linear Models

- Linear Regression
- Loss Minimization Framework
- Linear Classification

Univariate Linear Regression

- Suppose you have a house and want to sell it.
- How should you price it?
- Intuition: price of a house should be proportional to its area.

- Collect some transaction records in the neighborhood:

$$\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\} \quad \textit{(Dataset)}$$

x_i: area of the i^{th} house, y_i: price of the i^{th} house, n: number of data samples

- Fitting a linear regression: $f(x_i) = wx_i + b$, such that $f(x_i)$ is close to y_i.

coefficient/weight *bias*

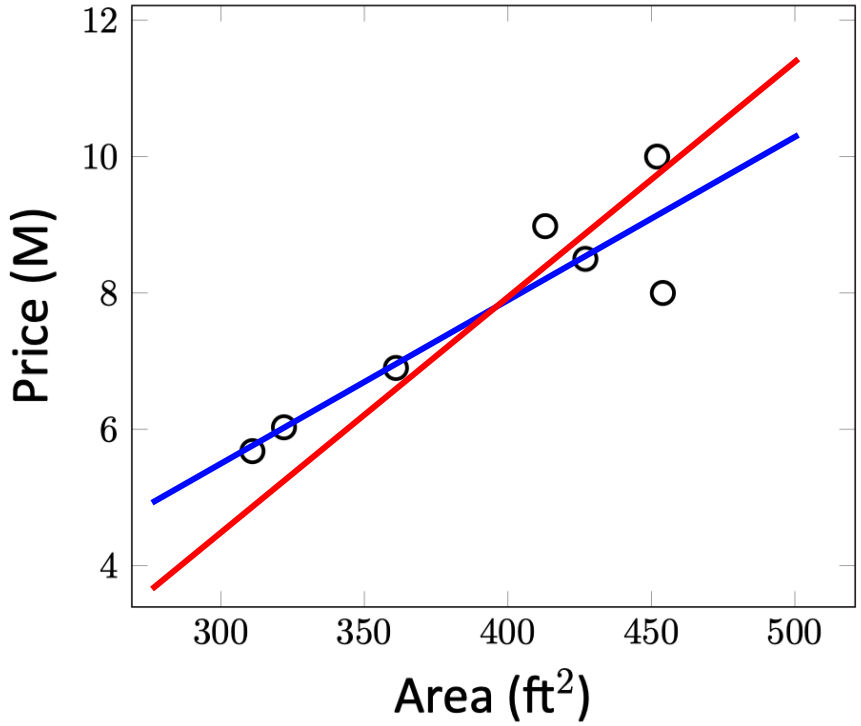
Loss Function: How Good Is A Predictor?

Fitting a linear regression: $f(x_i) = wx_i + b$, such that $f(x_i)$ is close to y_i.

- Hypothesis space (a set of all possible hypotheses):

$$\mathcal{F} = \{f_{w,b} : w \in \mathbb{R}, b \in \mathbb{R}\}$$

  - Hypothesis 1: $f(x) = 0.024x - 1.67$
  - Hypothesis 2: $f(x) = 0.035x - 6.02$



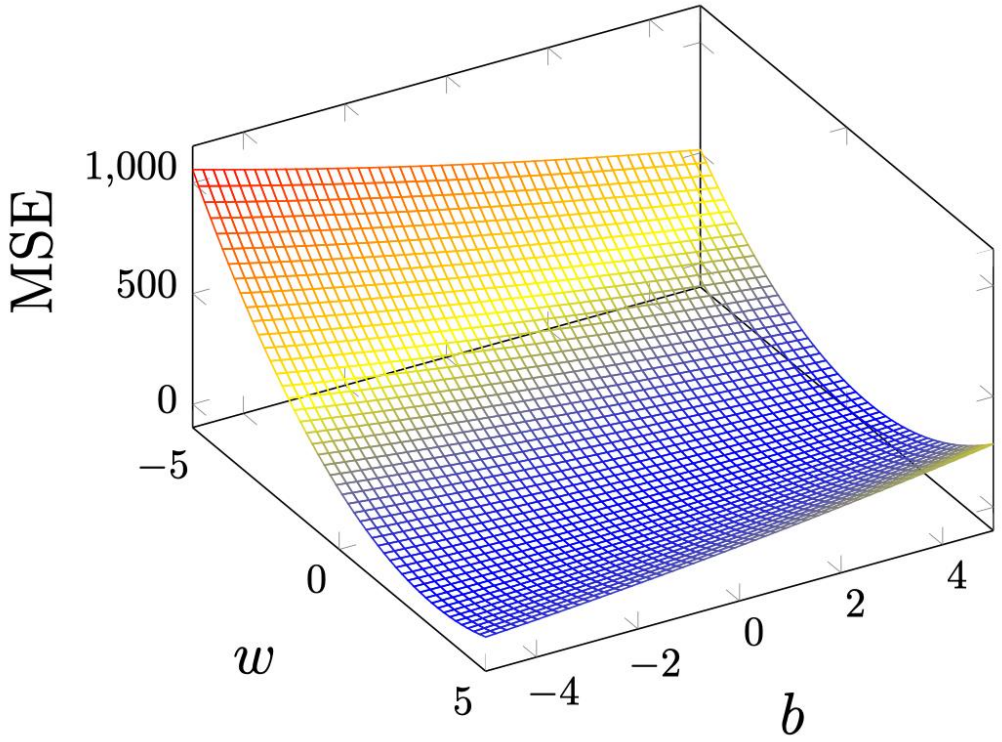
- Which one should we choose?

residual

- The square loss: $\ell(x_i, y_i, w, b) = \left(f_{w,b}(x_i) - y_i\right)^2$

- The **mean square loss (error)**: $MSE(w, b) = \frac{1}{n}\sum_{(x_i,y_i)\in\mathcal{D}}\left(f_{w,b}(x_i) - y_i\right)^2$

Loss Function: A Visualization of A 1-Dimensional MSE Loss



$$\text{MSE}(w, b) = \frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}} (f_{w,b}(x_i) - y_i)^2 = \frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}} (wx_i + b - y_i)^2$$

Loss Minimization Framework: How to Find The Best Parameters?

- Fitting a linear regression: $f(x_i) = wx_i + b$.
- Our objective: to make the mean square loss (MSE) as small as possible.

- **Convert it to the optimization problem:**

Optimal parameters

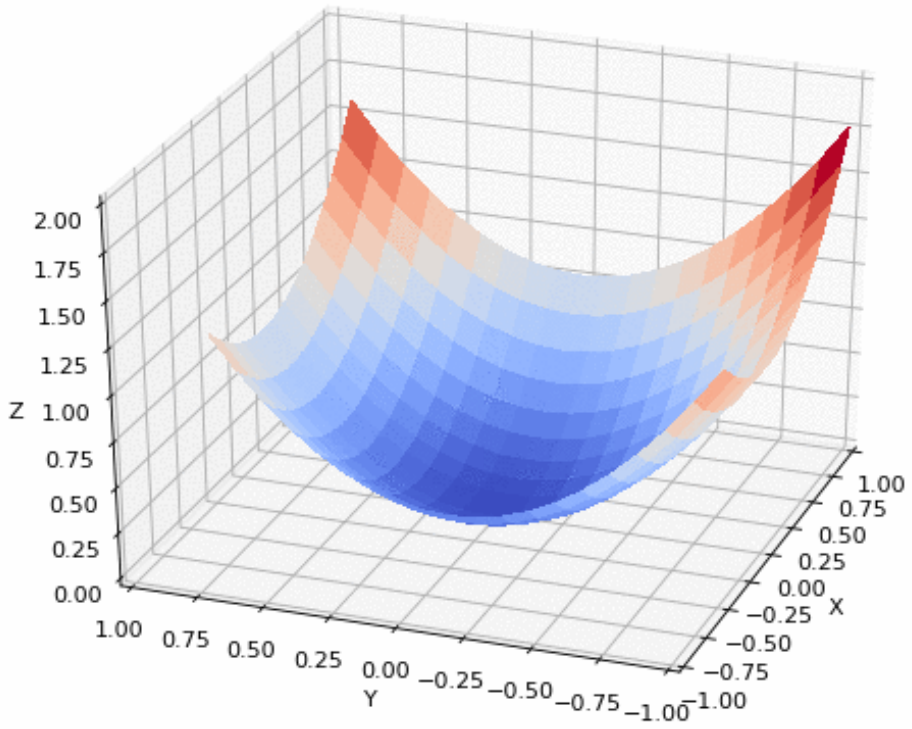
$$\boxed{(w^*, b^*)} = \arg\min_{w,b} \text{MSE}(w, b)$$

$$= \arg\min_{w,b} \frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}} (wx_i + b - y_i)^2$$

- Algorithm to use: Gradient descent *(More about optimization is covered in COMP7180)*.

Gradient Descent Algorithm

- **Gradient**: the gradient $\nabla_w f(w)$ is the direction of the greatest increase of $f(w)$.
- Start from an initial location, repeatedly move a small step opposite the gradient direction.



"Rolling a ball down the hill"

Gradient Descent Algorithm for Linear Regression

Initialize $w = 0$ and $b = 0$;

For $t = 1, \ldots, T$, do

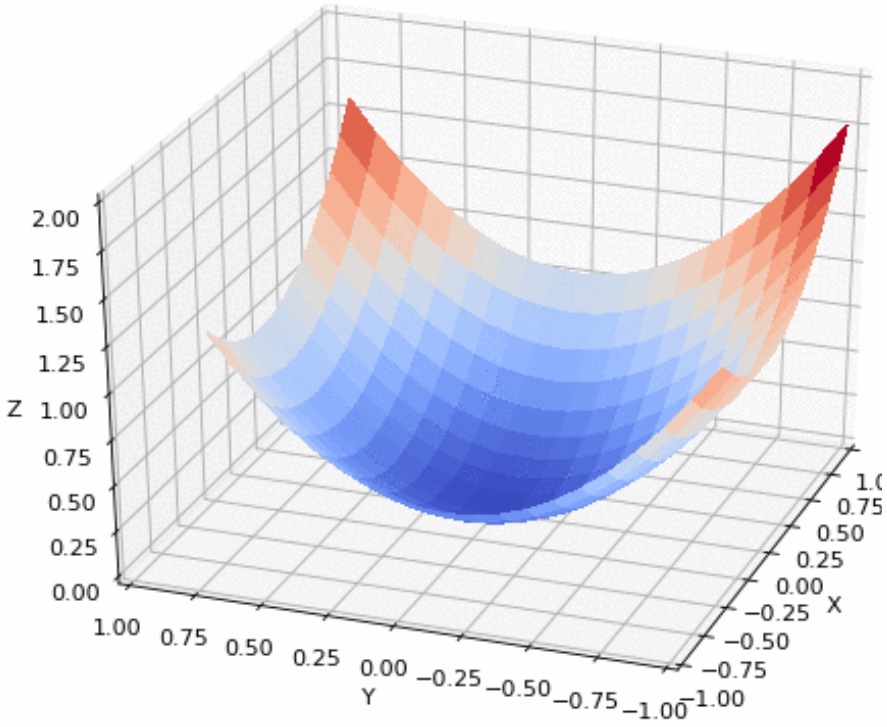
$$w \leftarrow w - \eta \nabla_w \mathrm{MSE}(w, b)$$

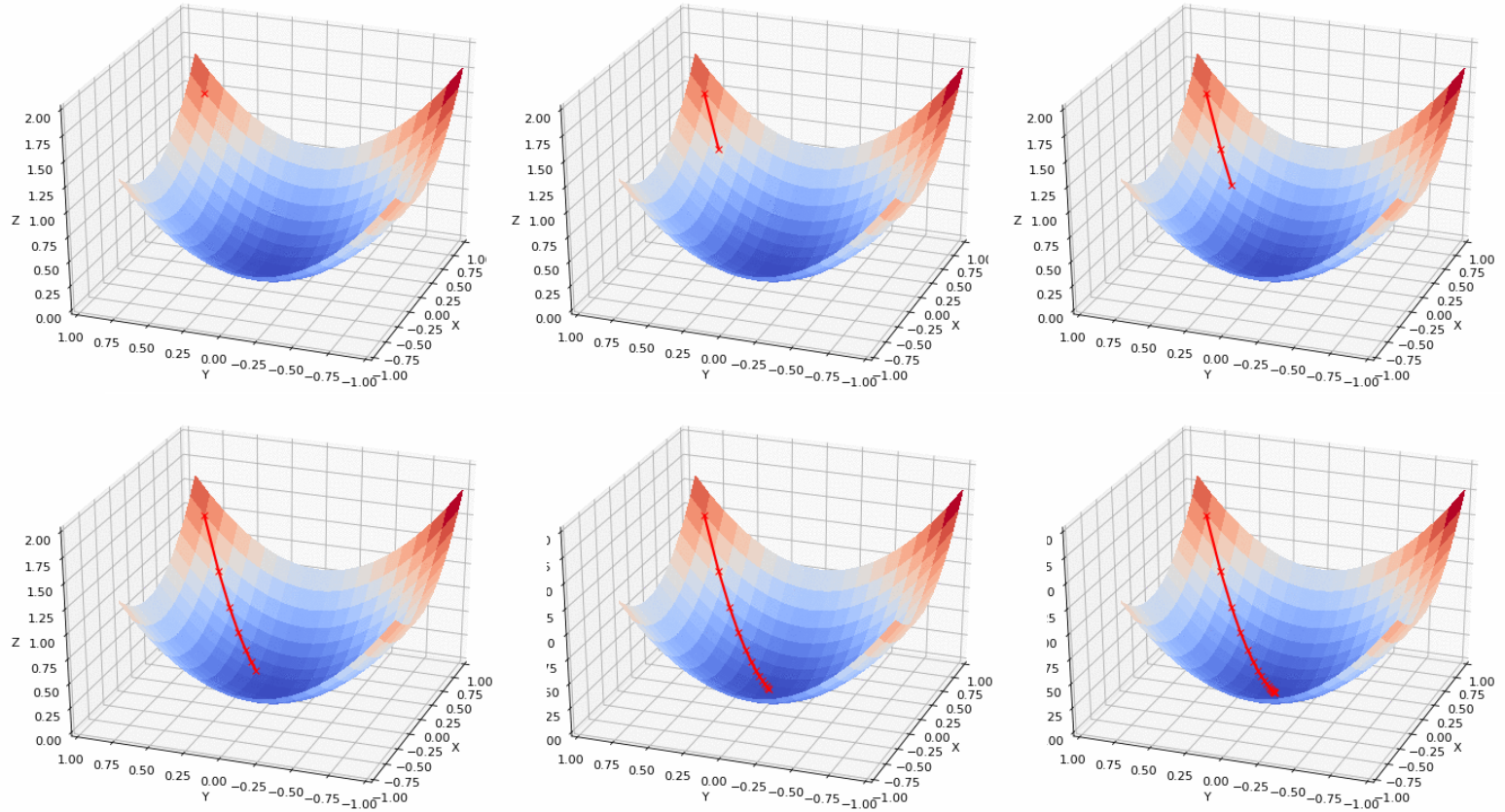
$$b \leftarrow b - \eta \nabla_b \mathrm{MSE}(w, b)$$

- η: step size (how much do we move in each step);
- $\nabla_w MSE(w, b)$ and $\nabla_b MSE(w, b)$: gradient of MSE with respect to w and b, respectively.

Gradient Descent Algorithm

- **Gradient**: the gradient $\nabla_w f(w)$ is the direction of the greatest increase of $f(w)$.
- Start from an initial location, repeatedly move a small step opposite the gradient direction.





"Rolling a ball down the hill"

Gradient Computation

$$\text{MSE}(w, b) = \frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}} (f_{w,b}(x_i) - y_i)^2 = \frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}} (wx_i + b - y_i)^2$$

$$\begin{aligned}
\frac{\partial}{\partial w} \text{MSE}(w, b) &= \frac{\partial}{\partial w} \left[\frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}} (wx_i + b - y_i)^2 \right] \\
&= \frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}} \frac{\partial}{\partial w} (wx_i + b - y_i)^2 \\
\textit{(By chain rule)} \quad &= \frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}} 2(wx_i + b - y_i) \cdot \frac{\partial}{\partial w} (wx_i + b - y_i) \\
&= \frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}} 2(wx_i + b - y_i) \cdot x_i
\end{aligned}$$

Summary So Far

- Dataset: $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$
- Linear regression (1-dimensional): $f_{w,b}(x_i) = wx_i + b$
- Hypothesis space: $\mathcal{F} = \{f_{w,b} : w \in \mathbb{R}, b \in \mathbb{R}\}$
- Mean square loss: $MSE(w, b) = \frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}} \left(f_{w,b}(x_i) - y_i\right)^2$
- Gradient descent algorithm: $w \leftarrow w - \eta \nabla_w MSE(w, b)$

Linear Regression for d-Dimensional Data

- House pricing example: price is also related to the location, building age, facing, etc.
- We represent each data sample by a d dimensional column vector: $x \in \mathbb{R}^d$.
- We say that the data samples have d *features*.
- The dataset: $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$

- d-dimensional linear regression: $f_{w,b}(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i + b$ or $f_{\hat{\mathbf{w}}}(\mathbf{x}_i) = \hat{\mathbf{w}}^T\mathbf{x}_i$

  where the bias term is absorbed into the weight vector:

$$\underbrace{\begin{bmatrix} w_1 & w_2 & w_3 & b \end{bmatrix}}_{\hat{\mathbf{w}}=[\mathbf{w};b]\in\mathbb{R}^{d+1}} \cdot \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix}}_{\hat{\mathbf{x}}\in\mathbb{R}^{d+1}} = w_1x_1 + w_2x_2 + w_3x_3 + b$$

Linear Regression for d-Dimensional Data

- Dataset: $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$
- Linear regression (1-dimensional): $f_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i$

  (from now on, we directly use $\boldsymbol{w} \in \mathbb{R}^{d+1}$ instead of $\hat{\boldsymbol{w}}$ for simplicity)
- Hypothesis space: $\mathcal{F} = \{f_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^{d+1}\}$
- Mean square loss:

$$MSE(\mathbf{w}) = \frac{1}{n} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} (\mathbf{w}^T \mathbf{x}_i - y_i)^2$$

- Gradient descent algorithm: $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} MSE(\mathbf{w})$

Linear Regression for d-Dimensional Data

- Rewriting it in the matrix form:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_1}^\top & 1 \\ \mathbf{x_2}^\top & 1 \\ \vdots & \vdots \\ \mathbf{x_n}^\top & 1 \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} & 1 \\ x_{21} & x_{22} & \cdots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nd} & 1 \end{bmatrix} \in \mathbb{R}^{n\times(d+1)}$$

$$\mathbf{y} = \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix}^\top \in \mathbb{R}^n$$

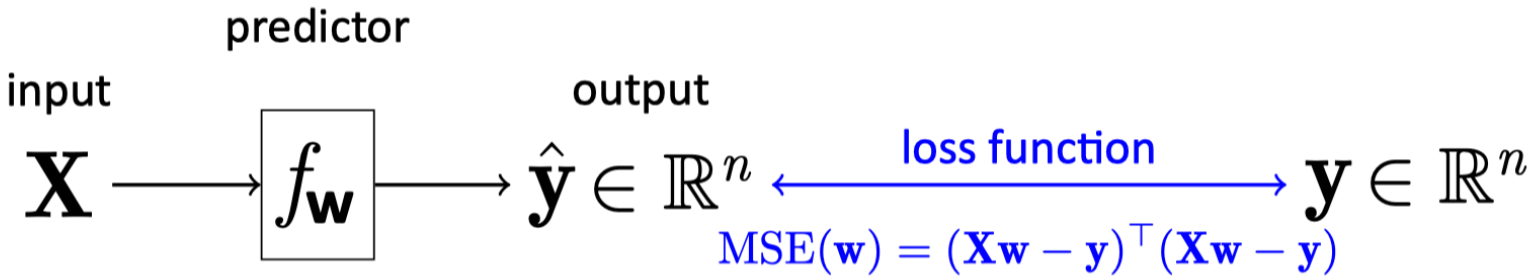
- Each row of $\mathbf{X}$: a *data sample*
- Each column of $\mathbf{X}$: a *feature*
- The mean square loss can be written as: $MSE(\mathbf{w}) = \|\mathbf{Xw} - \mathbf{y}\|_2^2 = (\mathbf{Xw} - \mathbf{y})^T(\mathbf{Xw} - \mathbf{y})$
- Gradient in matrix form: $\nabla_{\mathbf{w}} MSE(\mathbf{w}) = 2\mathbf{X}^T(\mathbf{Xw} - \mathbf{y})$

Gradient of MSE Loss

- The mean square loss can be written as: $MSE(\mathbf{w}) = \|\mathbf{Xw} - \mathbf{y}\|_2^2 = (\mathbf{Xw} - \mathbf{y})^T(\mathbf{Xw} - \mathbf{y})$
- Its gradient can be derived by:

$$
\begin{aligned}
\nabla_{\mathbf{w}} \mathrm{MSE}(\mathbf{w}) &= \nabla_{\mathbf{w}} \left[(\mathbf{Xw} - \mathbf{y})^\top (\mathbf{Xw} - \mathbf{y}) \right] \\
&= \nabla_{\mathbf{w}} \left[(\mathbf{Xw})^\top (\mathbf{Xw}) - (\mathbf{Xw})^\top \mathbf{y} - \mathbf{y}^\top (\mathbf{Xw}) + \mathbf{y}^\top \mathbf{y} \right] \\
&= \nabla_{\mathbf{w}} \left[\mathbf{w}^\top (\mathbf{X}^\top \mathbf{X}) \mathbf{w} - 2 (\mathbf{X}^\top \mathbf{y})^\top \mathbf{w} \right] \\
&= 2\mathbf{X}^\top \mathbf{Xw} - 2\mathbf{X}^\top \mathbf{y} \\
&= 2\mathbf{X}^\top (\mathbf{Xw} - \mathbf{y})
\end{aligned}
$$

Summary of Linear Regression



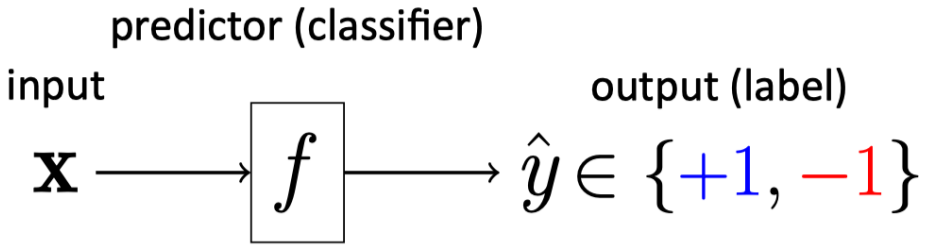
- What is the parameter to be learned? **w**
- How to learn it?

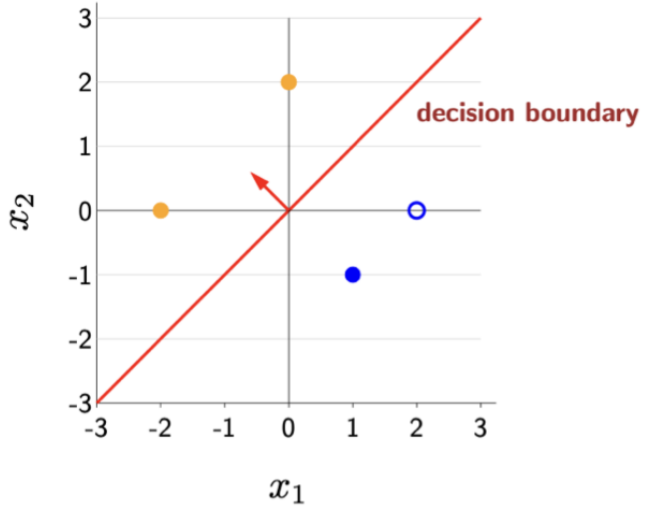
  Minimizing the loss function. *The loss function defines how good a model predicts the targets y.*
- How to minimize the loss function?

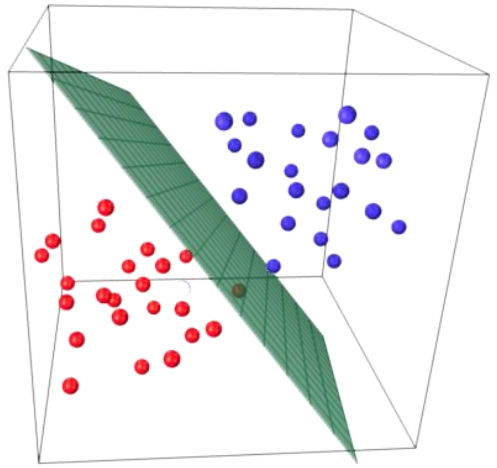
  Gradient descent algorithm.

Linear Classification

Can we use linear regression for classification?







- The **decision boundary** divides the data space into two regions: one would have output of +1 and the other would have output of −1.
- But our output from linear regression is a real number, instead of a binary label.
- We need to transform the real number output $\hat{y} \in \mathbb{R}$ to a binary value $\hat{y} \in \{+1, -1\}$.

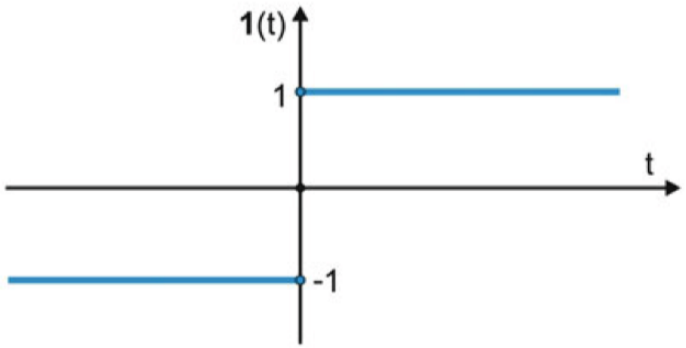
Unit-Step Function

Unit-Step Function:

$$f(\mathbf{x}_i) = sign(\mathbf{w}^T\mathbf{x}_i)$$

score

where $sign(t) = \begin{cases} +1 & if\ t > 0 \\ -1 & if\ t < 0 \\ 0 & if\ t = 0 \end{cases}$



The score on an example $(\mathbf{x}_i, y_i)$ is $\mathbf{w}^T\mathbf{x}_i$: How confident the model is in predicting +1.

Zero-One Loss Function

Predicted label: $f(\mathbf{x}_i) = sign(\mathbf{w}^T\mathbf{x}_i)$

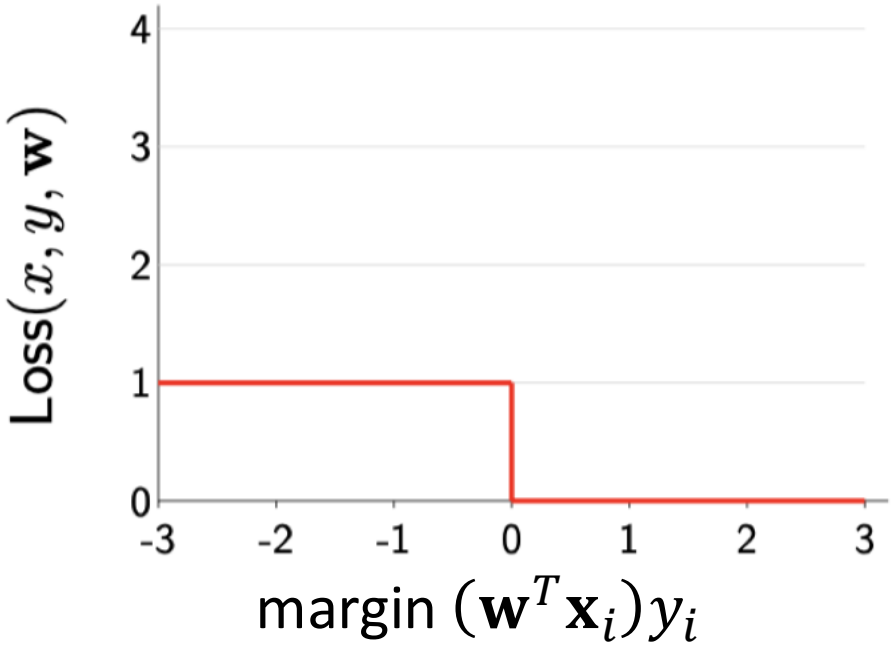
Target label: $y \in \{+1, -1\}$

We define the zero-one loss as:

$$\text{Loss}_{0-1}(\mathbf{x}, y, \mathbf{w}) = \underbrace{\mathbb{1}\left[f_{\mathbf{w}}(\mathbf{x}) \neq y\right]}_{\text{counting how many mistakes we make}}$$

$$= \mathbb{1}\left[\boxed{(\mathbf{w}^\top \mathbf{x})y} \leq 0\right]$$

margin



The margin on an example $(\mathbf{x}_i, y_i)$ is $(\mathbf{w}^T\mathbf{x}_i)y_i$: How correct the model is.

- $(\mathbf{w}^T\mathbf{x}_i)$: score; the larger the higher confidence in predicting +1.
- If $y = +1$, larger score → larger margin → "more correct".
- If $y = -1$, lower score → larger margin → "more correct".

Optimization: Gradient Descent with Zero-One Loss?

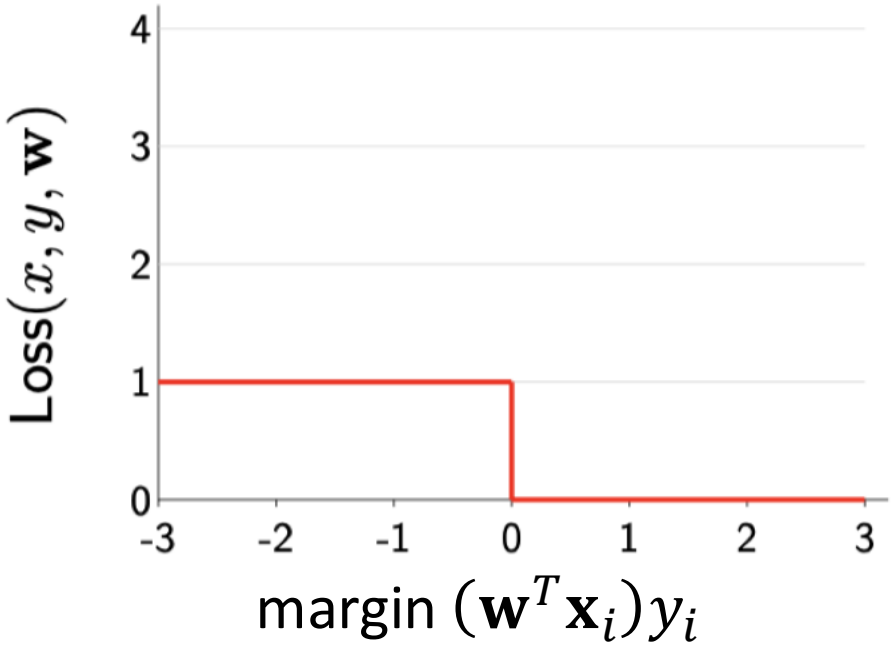
Predicted label: $f(\mathbf{x}_i) = sign(\mathbf{w}^T\mathbf{x}_i)$

Target label: $y \in \{+1, -1\}$

Zero-one loss: $\text{Loss}_{0-1}(\mathbf{x}, y, \mathbf{w}) = \mathbb{1}\left[(\mathbf{w}^\top\mathbf{x})y \leq 0\right]$

What is the gradient w.r.t. $\mathbf{w}$ in the zero-one loss?

Almost zero everywhere!



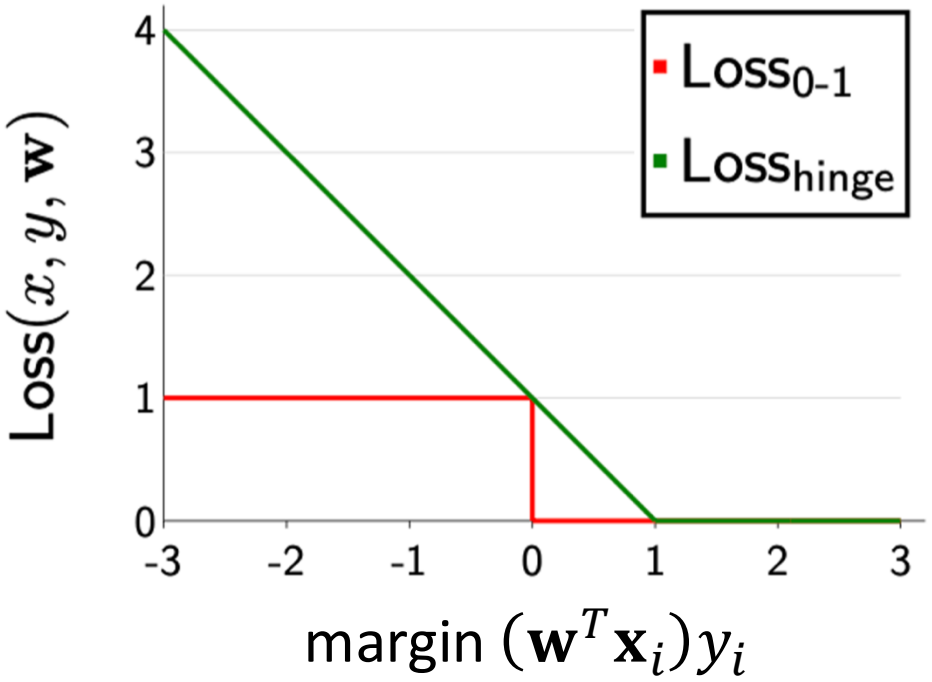
What went wrong?

- In MSE, when we move w a little bit, the residuals will change. The loss of some data points get smaller while some others get larger.
- In zero-one loss, moving w a tiny bit changes very little the loss, until we make a big change that makes an example cross the decision boundary.

Hinge Loss Function (SVM)

The margin on an example $(\mathbf{x}_i, y_i)$ is $(\mathbf{w}^T\mathbf{x}_i)y_i$: How correct the model is.

$$\text{Loss}_{\text{hinge}}(\mathbf{x}, y, \mathbf{w}) = \max\{1 - (\mathbf{w}^\top \mathbf{x})y, 0\}$$



- The hinge loss is zero when the model classifies all data points correctly and confidently (margin ≥ 1).
- If the model makes confident error, it incurs a linearly increasing penalty (margin < 0).
- The hinge loss is non-zero even if the margin is between zero and one.
- Application: Support Vector Machine (SVM)

Optimization: Gradient Descent with Hinge Loss

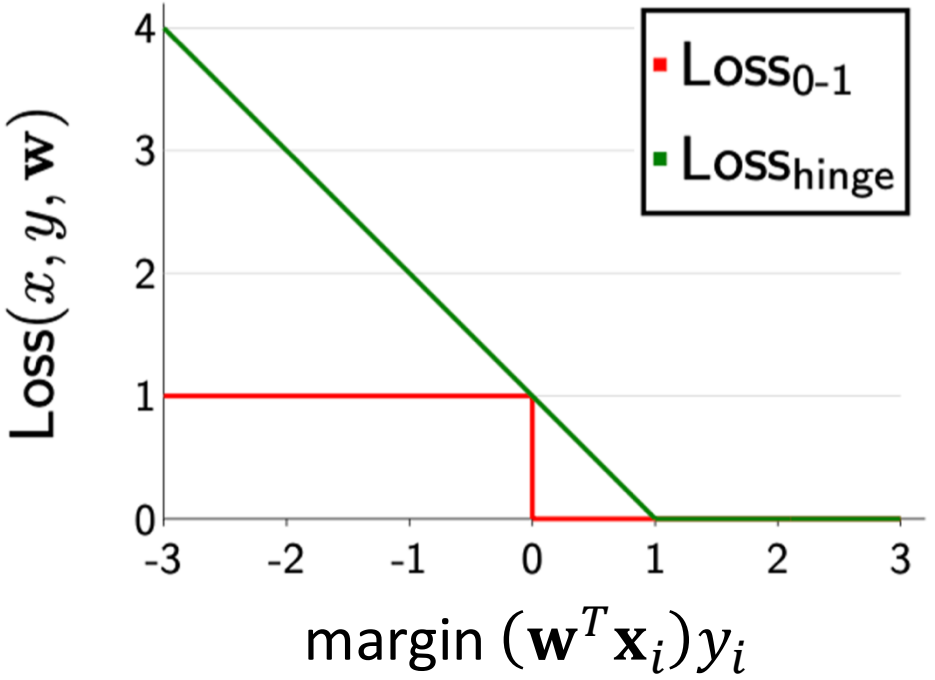
$$\text{Loss}_{\text{hinge}}(\mathbf{x}, y, \mathbf{w}) = \max\{1 - (\mathbf{w}^\top \mathbf{x})y, 0\}$$

What is the gradient for Hinge loss?

$$\nabla \text{Loss}_{\text{hinge}}(\mathbf{x}, y, w) = \begin{cases} -\mathbf{x}y & \text{if} (\mathbf{w}^\top \mathbf{x})y < 1 \\ 0 & \text{otherwise} \end{cases}$$

Gradient descent update rule:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \left[\frac{1}{n} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} \nabla \text{Loss}_{\text{hinge}}(\mathbf{x}, y, w) \right]$$



Logistic Regression: Sigmoid Function

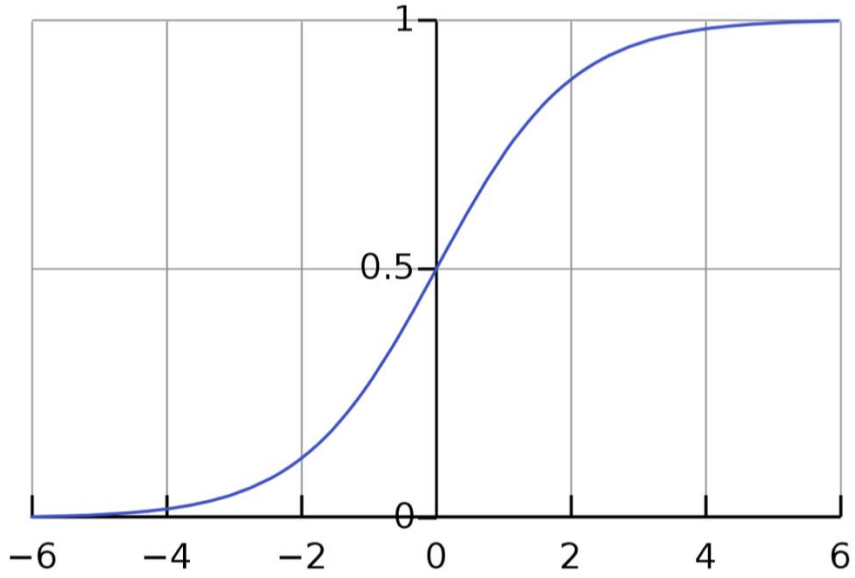
The sigmoid function (logistic function) maps a real value to be between (0, 1):

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

Substitute z with our score $\mathbf{w}^T\mathbf{x}$, we have:

$$f_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x})}$$

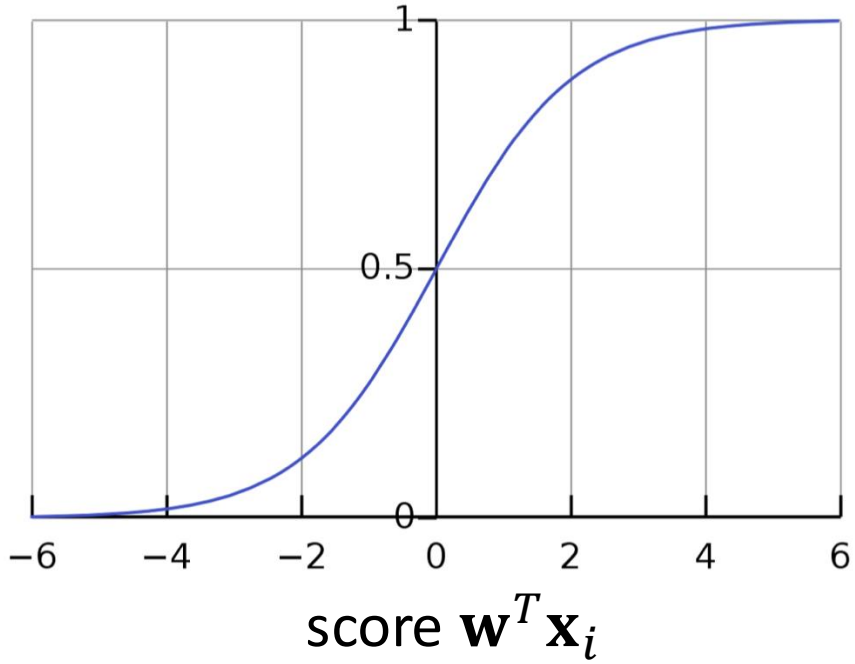
This model is called **logistic regression**.



Logistic Regression: The Probabilistic Interpretation

Consider this transformation returns a "probability":

- $p(y = +1|x) = \sigma(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^\top \mathbf{x})}$
- $p(y = -1|x) = 1 - \sigma(\mathbf{w}^\top \mathbf{x}) = \frac{\exp(-\mathbf{w}^\top \mathbf{x})}{1+\exp(-\mathbf{w}^\top \mathbf{x})} = \frac{1}{1+\exp(+\mathbf{w}^\top \mathbf{x})}$



The likelihood (probability of observing all data samples):

$$\prod_{(\mathbf{x}_i, y_i) \in \mathcal{D}} p(y_i = +1|\mathbf{x})^{\mathbb{1}[y_i=+1]} p(y_i = -1|\mathbf{x})^{\mathbb{1}[y_i=-1]}$$

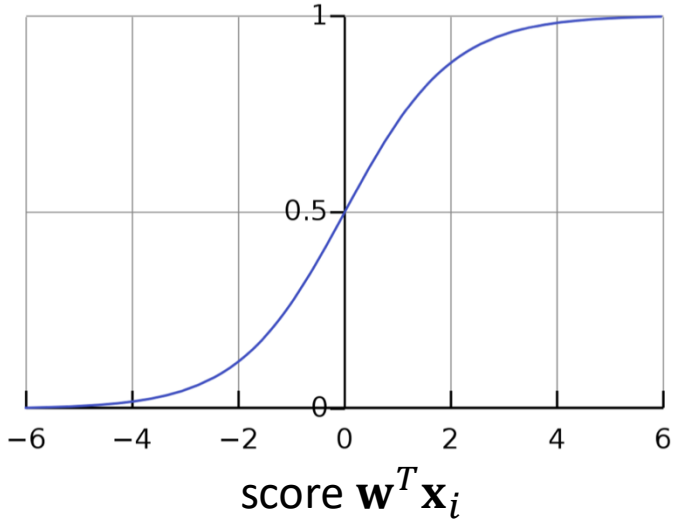
The log likelihood (numerically more stable):

$$\sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} \mathbb{1}[y_i = +1] \log p(y_i = +1|\mathbf{x}) + \mathbb{1}[y_i = -1] \log p(y_i = -1|\mathbf{x})$$

Logistic Regression: The Probabilistic Interpretation

Consider this transformation returns a "probability":

- $p(y=+1|x) = \sigma(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^\top \mathbf{x})}$
- $p(y=-1|x) = 1 - \sigma(\mathbf{w}^\top \mathbf{x}) = \frac{\exp(-\mathbf{w}^\top \mathbf{x})}{1+\exp(-\mathbf{w}^\top \mathbf{x})} = \frac{1}{1+\exp(+\mathbf{w}^\top \mathbf{x})}$



The log likelihood (numerically more stable):

$$\ell(\mathbf{w}) = \sum_{(\mathbf{x}_i, y_i)\in\mathcal{D}} \mathbb{1}[y_i = +1] \log p(y_i = +1|\mathbf{x}) + \mathbb{1}[y_i = -1] \log p(y_i = -1|\mathbf{x})$$

We can maximize the log-likelihood:

For positive samples ($y = +1$): $\ell^+(\mathbf{x}, y, \mathbf{w}) = \underbrace{\mathbb{1}[y_i = +1]}_{1} \log p(y_i = +1|\mathbf{x}) = -\log\left(1 + \exp(-\mathbf{w}^\top \mathbf{x})\right) = -\log\left(1 + \exp(-(\mathbf{w}^\top \mathbf{x})y)\right)$

For negative samples ($y = +1$): $\ell^-(\mathbf{x}, y, \mathbf{w}) = \underbrace{\mathbb{1}[y_i = -1]}_{1} \log p(y_i = -1|\mathbf{x}) = -\log\left(1 + \exp(+\mathbf{w}^\top \mathbf{x})\right) = -\log\left(1 + \exp(-(\mathbf{w}^\top \mathbf{x})y)\right)$

Equivalently, we can minimize the negative log-likelihood:

$$\text{Loss}(\mathbf{x}, y, \mathbf{w}) = \log(1 + \exp(-(\mathbf{w}^\top \mathbf{x})y))$$

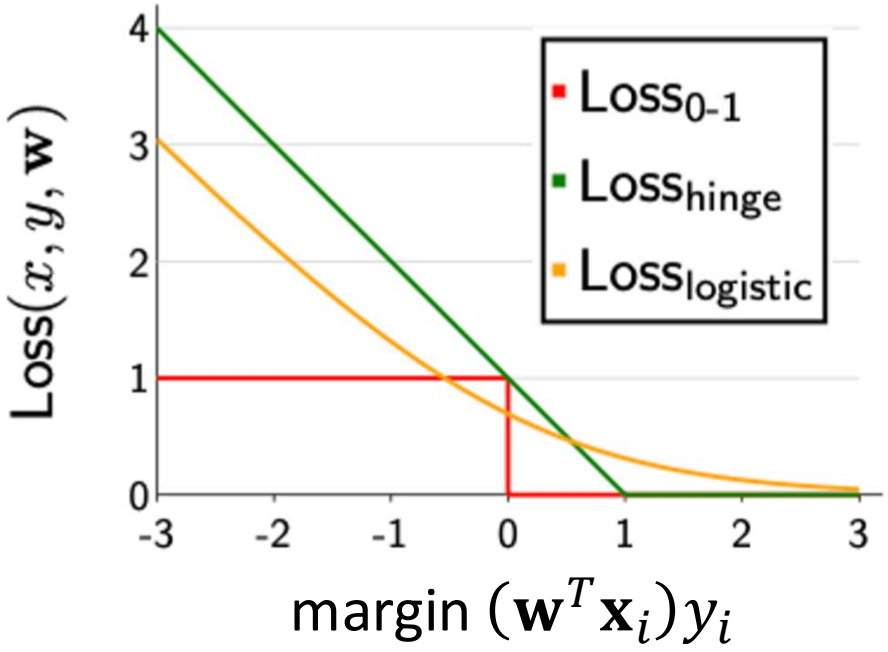
Logistic Regression: The Logistic Loss Function

The logistic loss function:

$$\text{Loss}_{\text{logistic}}(\mathbf{x}, y, \mathbf{w}) = \log(1 + \exp(-(\mathbf{w}^\top \mathbf{x})y))$$

$$\text{TrainLoss}_{\text{logistic}}(\mathbf{w}) = \sum_{(\mathbf{x},y)\in\mathcal{D}_{\text{train}}} \log(1 + \exp(-(\mathbf{w}^\top \mathbf{x})y))$$

- Try to increase margin even when it already exceeds 1.
- Always have non-zero loss.



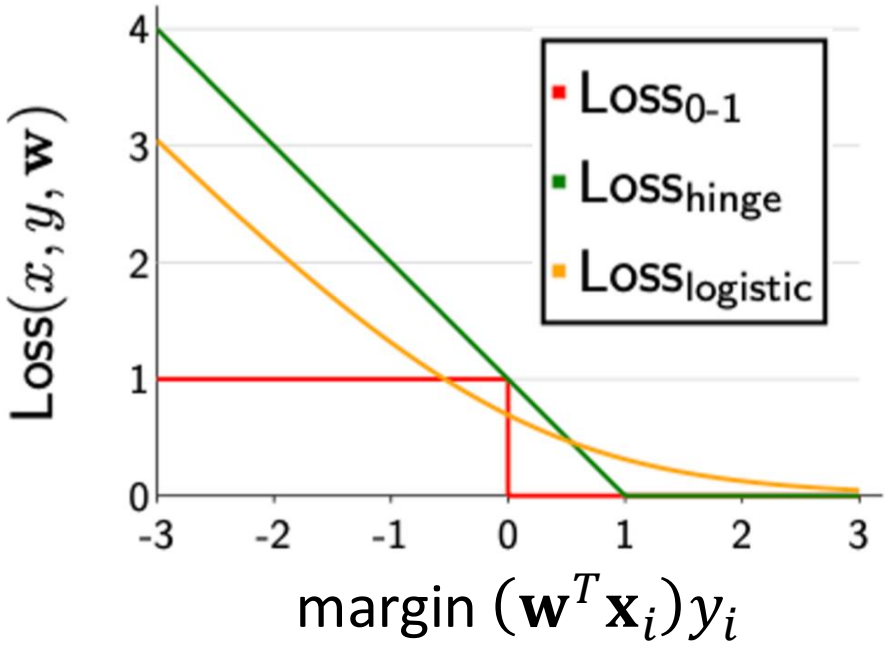
Exercise: computing its gradient.

$$\begin{aligned}
\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w}) &= \sum_{(\mathbf{x},y)\in\mathcal{D}_{\text{train}}} \frac{\partial}{\partial \mathbf{w}} \log(1 + \exp(-(\mathbf{w}^\top \mathbf{x})y)) \\
&= \sum_{(\mathbf{x},y)\in\mathcal{D}_{\text{train}}} \frac{1}{1 + \exp(-(\mathbf{w}^\top \mathbf{x})y)} \frac{\partial}{\partial \mathbf{w}} \exp(-(\mathbf{w}^\top \mathbf{x})y) \\
&= \sum_{(\mathbf{x},y)\in\mathcal{D}_{\text{train}}} \frac{\exp(-(\mathbf{w}^\top \mathbf{x})y)}{1 + \exp(-(\mathbf{w}^\top \mathbf{x})y)} \frac{\partial}{\partial \mathbf{w}} (-(\mathbf{w}^\top \mathbf{x})y) \\
&= -\sum_{(\mathbf{x},y)\in\mathcal{D}_{\text{train}}} y \frac{\exp(-(\mathbf{w}^\top \mathbf{x})y)}{1 + \exp(-(\mathbf{w}^\top \mathbf{x})y)} \mathbf{x}
\end{aligned}$$

Summary So Far (Linear Models)

- Loss Minimization Framework: loss function & gradient descent.
- Regression: Mean Square Error (MSE)
- Classification: Three loss functions defined based on margin.
  - The score on an example $(\mathbf{x}_i, y_i)$ is $\mathbf{w}^T\mathbf{x}_i$: How confident the model is in predicting +1.
  - The margin on an example $(\mathbf{x}_i, y_i)$ is $(\mathbf{w}^T\mathbf{x}_i)y_i$: How correct the model is.

	Regression	Classification
Prediction $f_{\mathbf{w}}(x)$	score	sign(score)
Relate to target y	residual (score $- y$)	margin (score y)
Loss functions	squared absolute deviation	zero-one hinge logistic
Algorithm	gradient descent	gradient descent



Generalization and Model Selection

- Generalization: Training Error vs. Test Error
- Model Selection via Cross Validation
- Preventing Overfitting

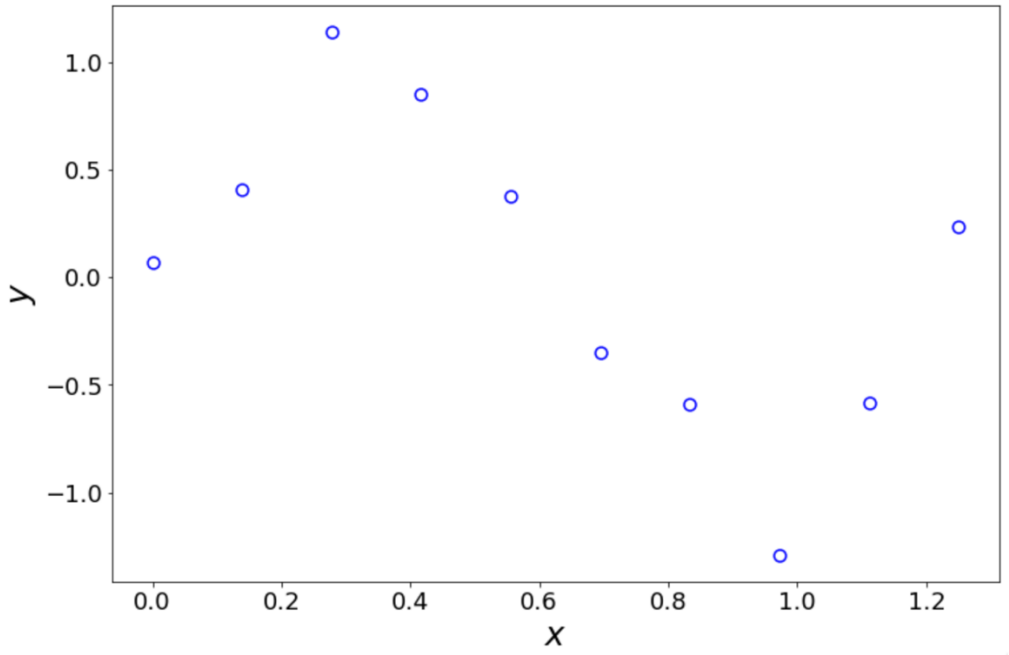
Example: Regression

Let's consider a simple polynomial extension of linear regression:

Linear regression: $y = wx + b$

p-order polynomial regression:

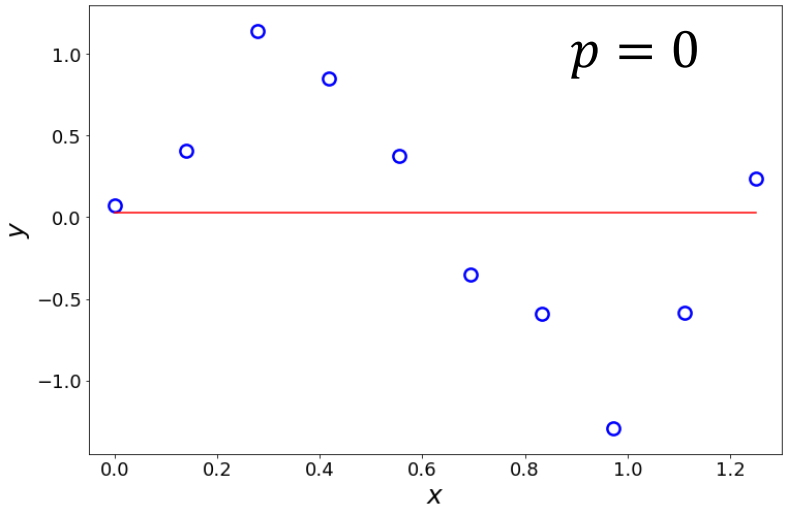
$$y = w_1 x + w_2 x^2 + \cdots + w_p x^p + b$$



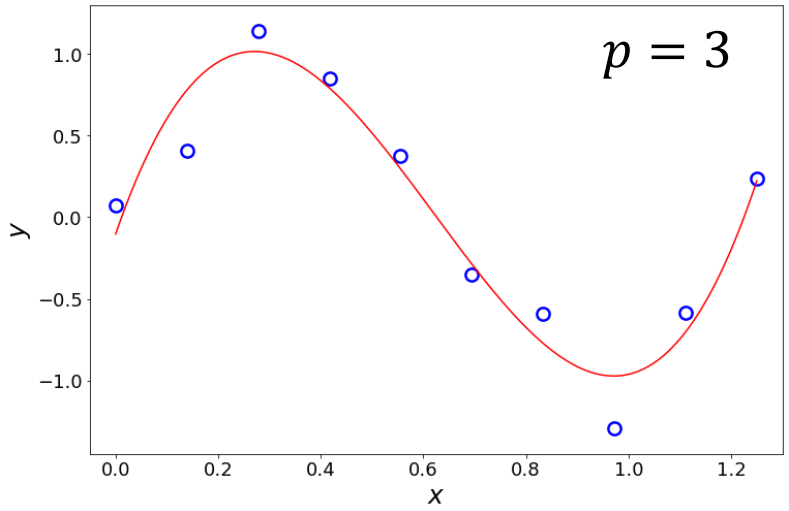
some data points to fit the model

Example: Regression

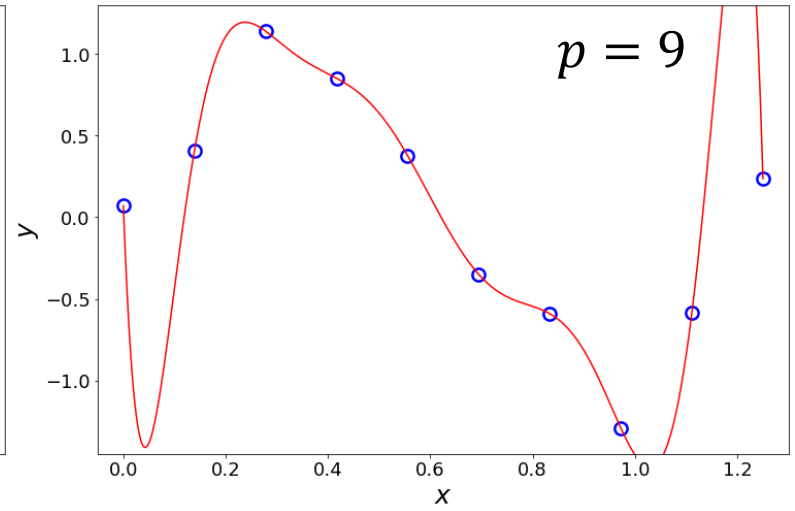
Let's consider a simple polynomial extension of linear regression:



MSE = 0.487



MSE = 0.035



MSE = 0

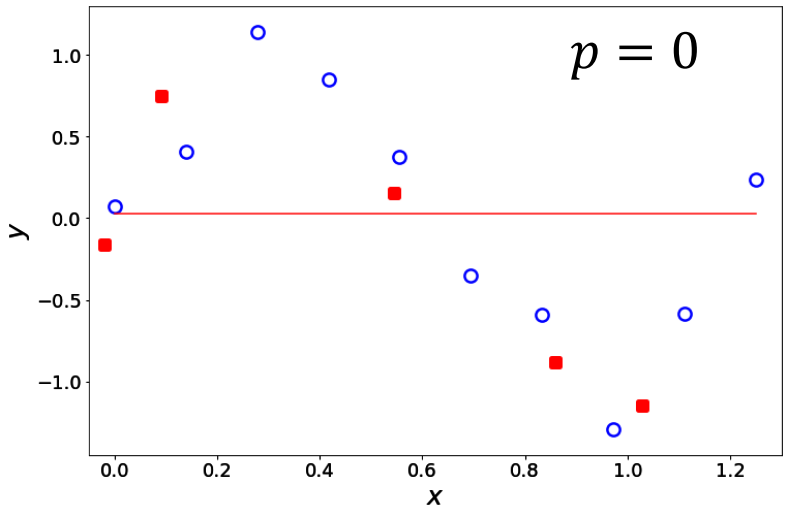
Which is a better model? It seems $p = 9$ is the best: the lowest MSE.

Hold on… What is the goal of doing machine learning?

(Intuition) Making predictions for new data!

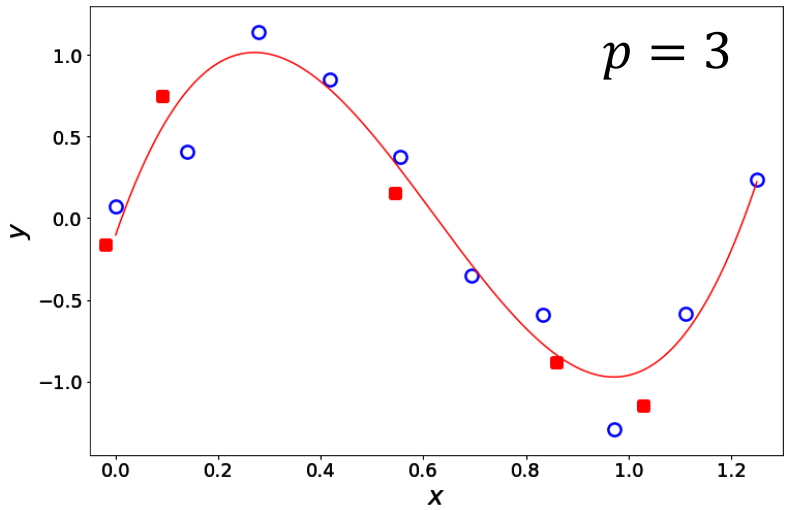
Example: Regression

Let's add in a few new data points (red squares):



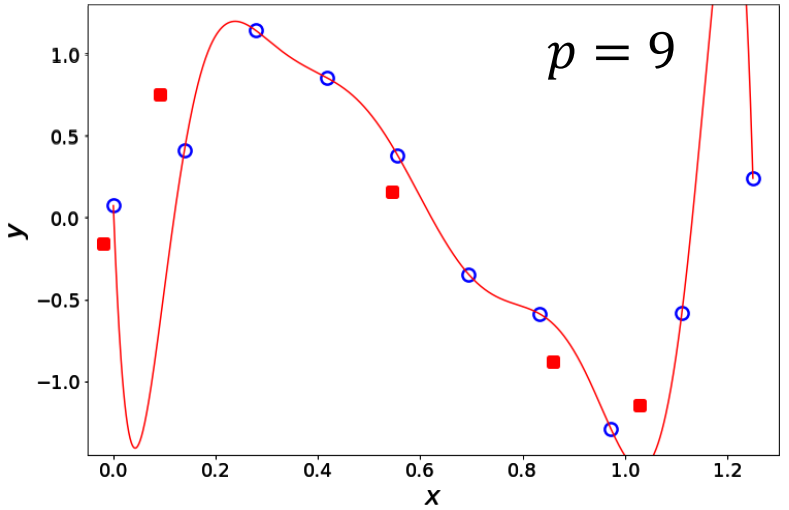
MSE(original data) = 0.487

MSE(new data) = 0.473



MSE (original data) = 0.035

MSE (new data) = 0.135



MSE (original data) = 0

MSE (new data) = 136.76

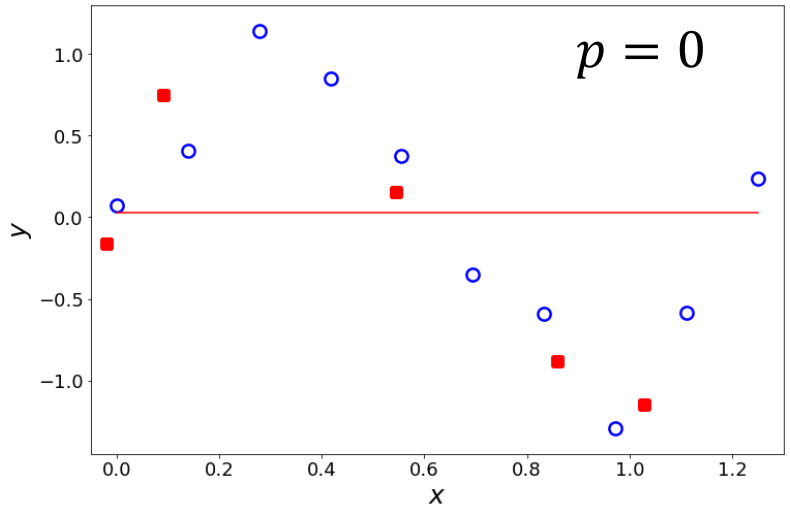
Is $p = 9$ a good model?

Zero loss when fitting the model;

Huge loss when using the model for new data.

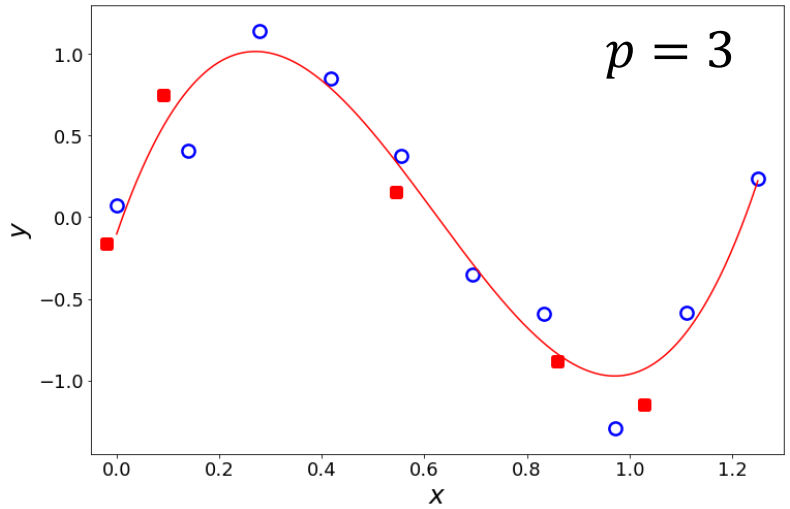
Example: Regression

Terminologies: Underfitting and overfitting



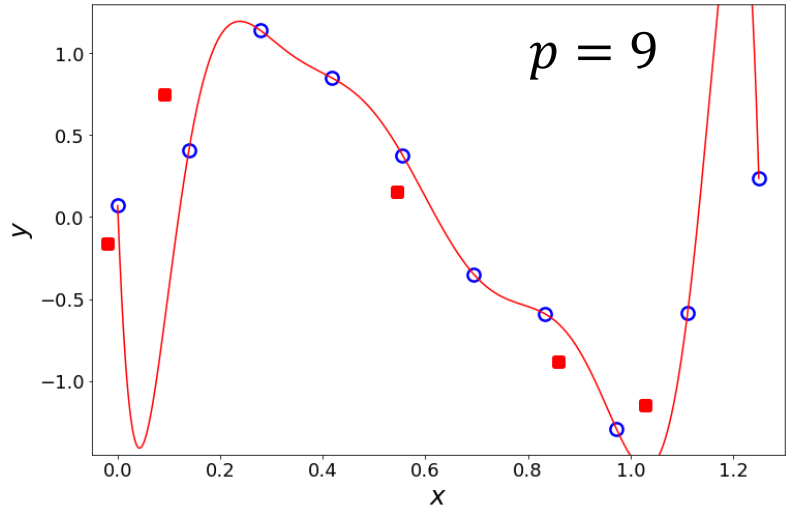
MSE(original data) = 0.487

MSE(new data) = 0.473



MSE (original data) = 0.035

MSE (new data) = 0.135



MSE (original data) = 0

MSE (new data) = 136.76

Underfitting

Large error for training & testing

Overfitting

Small training error, large testing error

Terminologies

- Generalization: the ability of a machine learning model to adapt to new and previously unseen data. (*a central task in ML*)
- We train a ML model on some data (called training data) and want to apply it to some new data (called testing data).
- Underfitting: when a model has large error in both training data and testing data.
- Overfitting: when a model has small error in training data, but large error in testing data.

Terminologies

Why can we expect good generalization?

- Fundamental assumption in machine learning: data are independent and identically distributed (i.i.d. / IID).
- Intuition: it allows the patterns learned to be applied to new data.

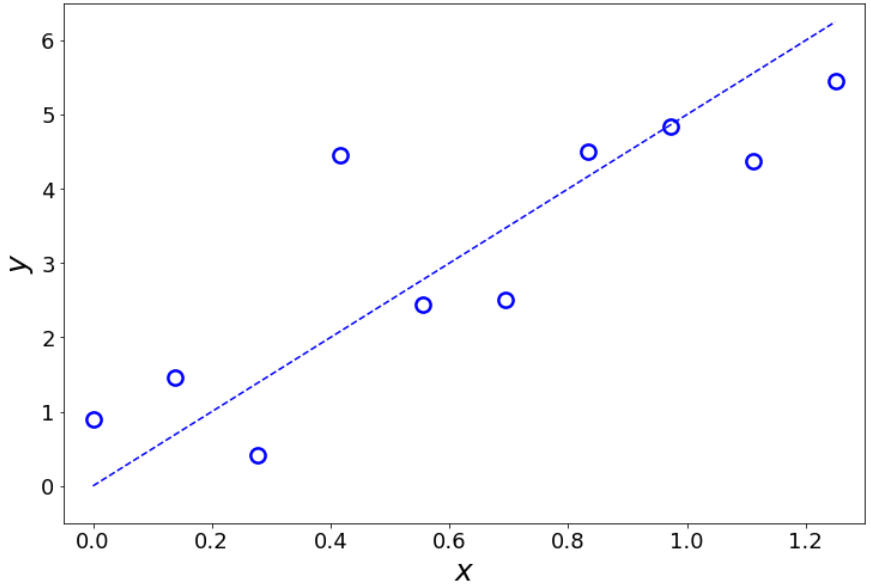
Question: can we train a heart failure prediction model using data collected from elderlies and directly apply that to the young?

No! Elderlies and the young have different distribution in their health conditions.

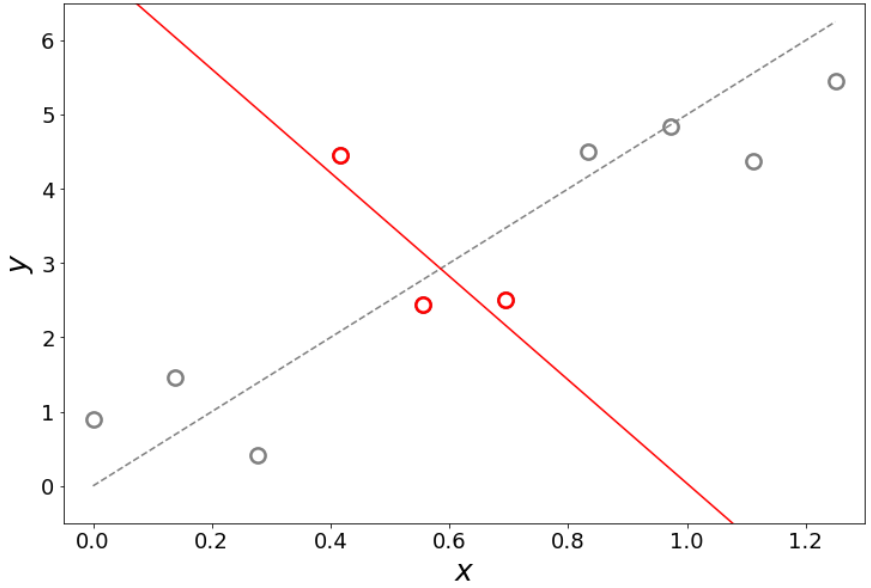
(Advanced methods exist to allow such application,
but direct application could lead to serious outcomes)

Reasons of Poor Generalization

Data is not enough or not representative



Blue line: A well fitted linear model

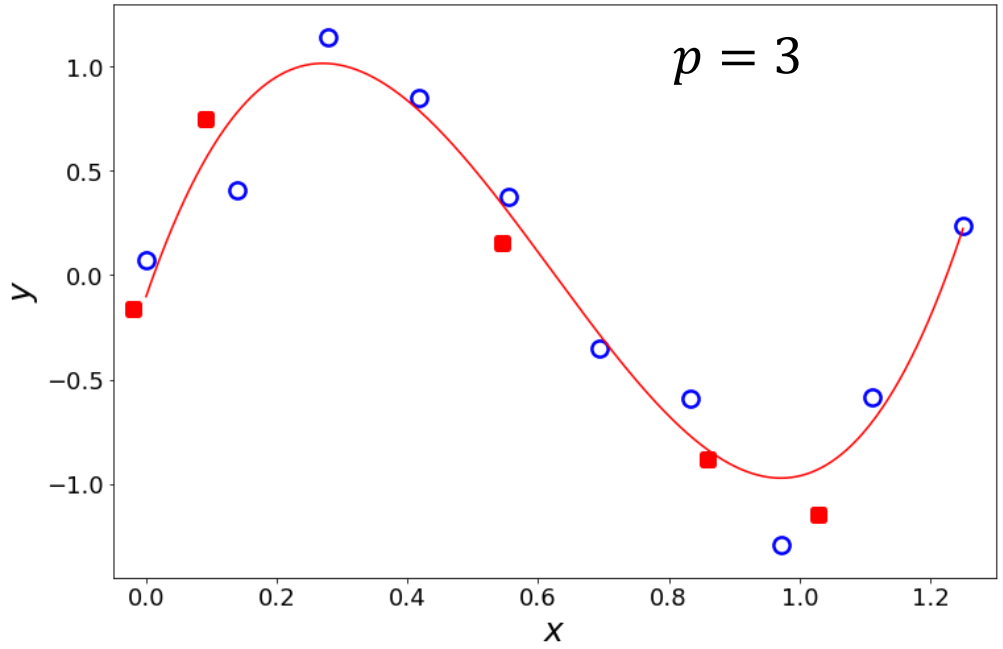


If we only observe three data samples that are not representative

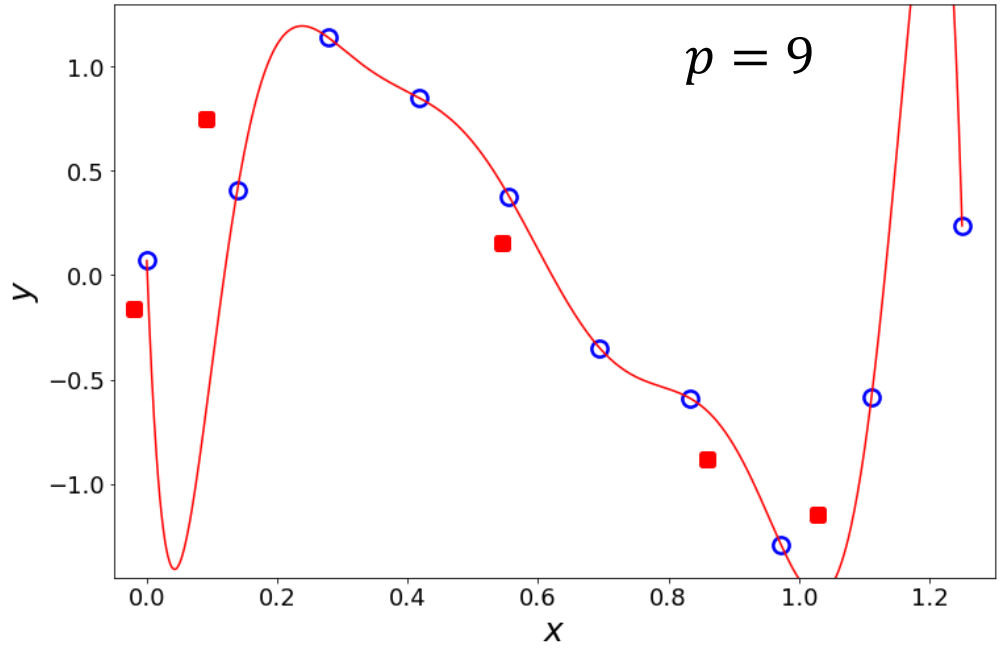
Overfitting to the observed (red) data!

Reasons of Poor Generalization

The model is too complex



Less complex model

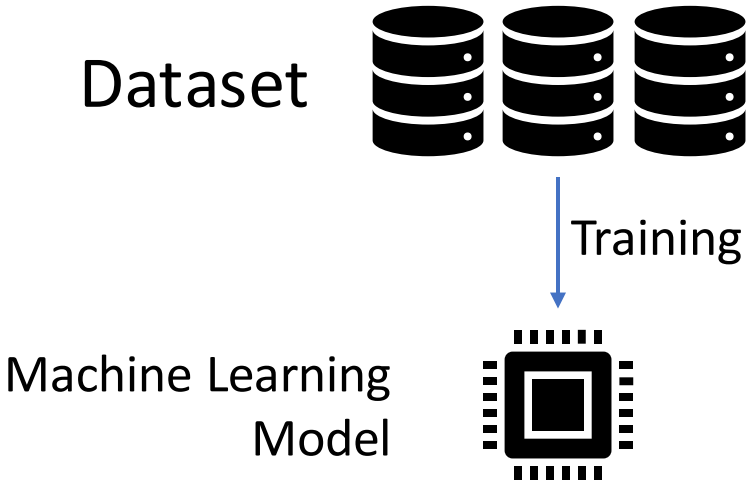


Very complex model

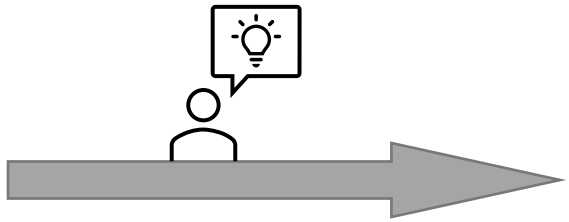
Complex models are more expressive but is more prone to overfitting!

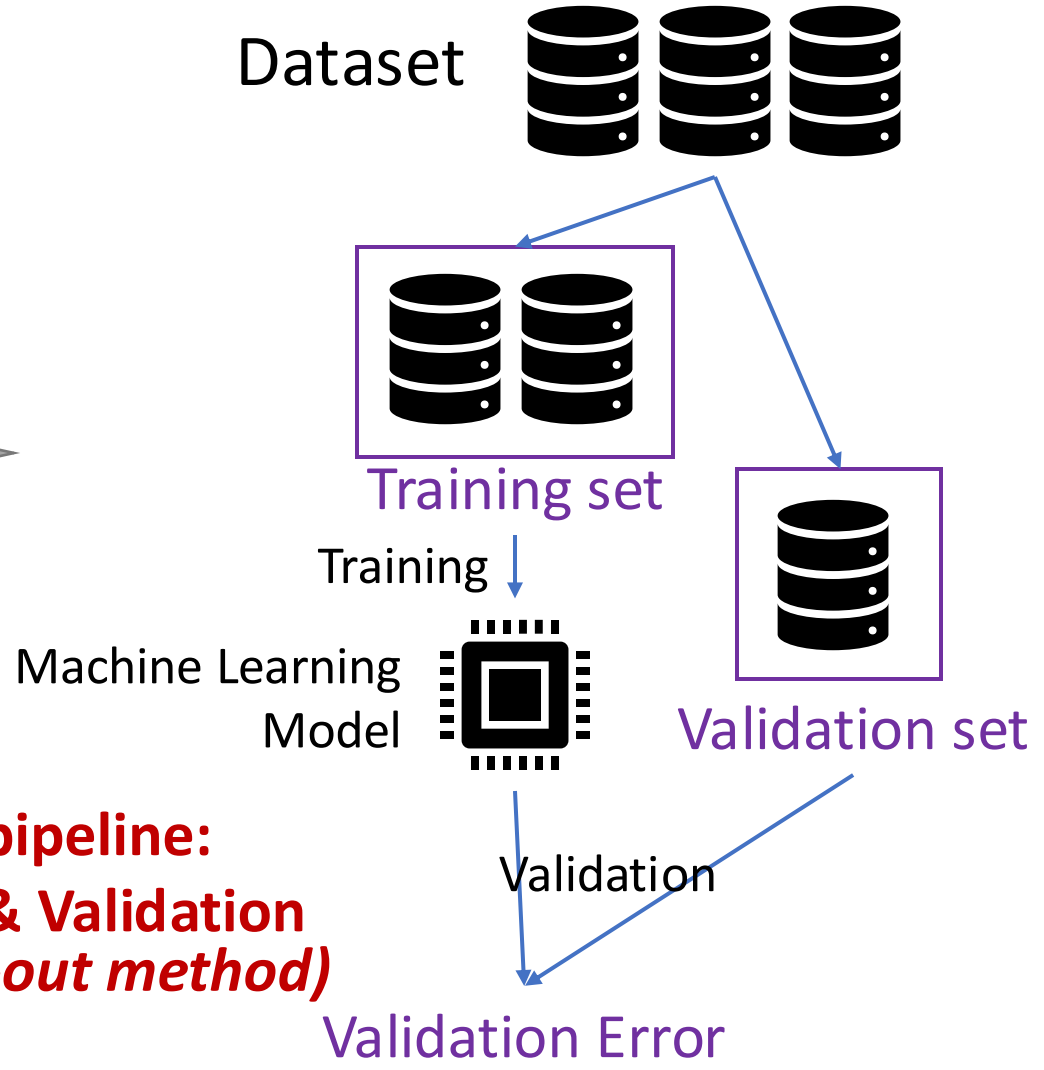
How to Measure Generalization?

(How do we know if the model overfits?)



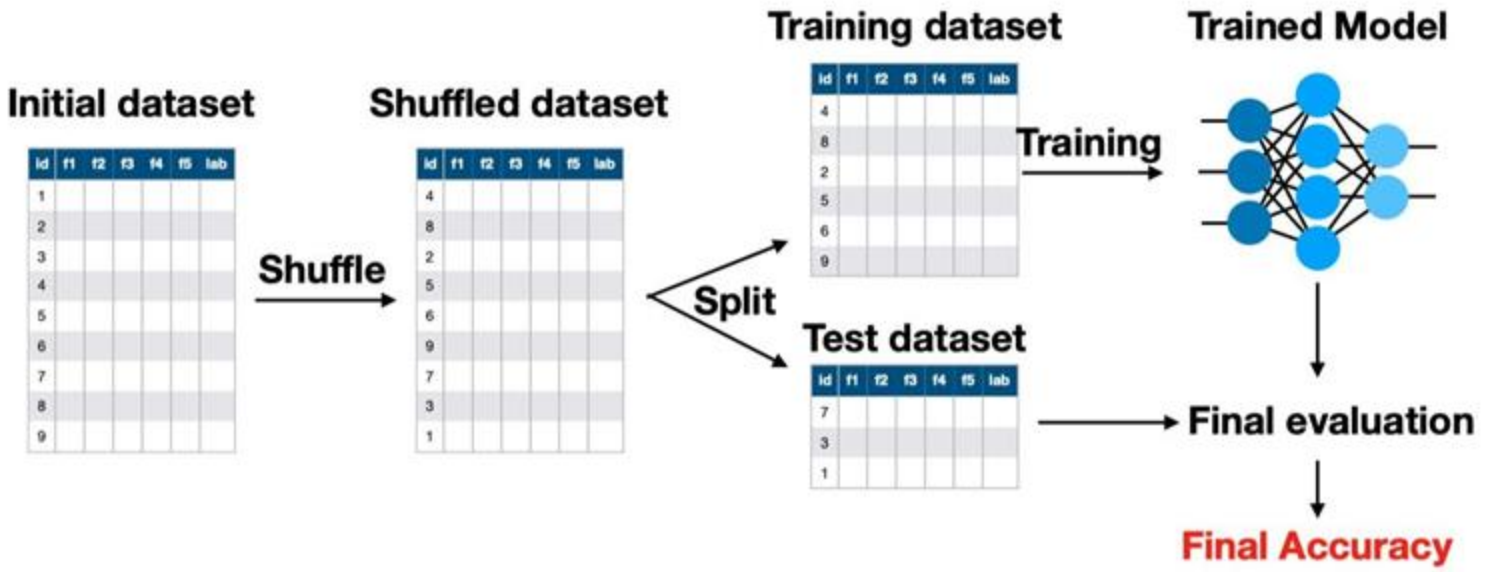
Our current pipeline:
Cannot know how good it generalizes





New pipeline:
Training & Validation
(e.g., hold-out method)

Hold-out Method for Performance Evaluation



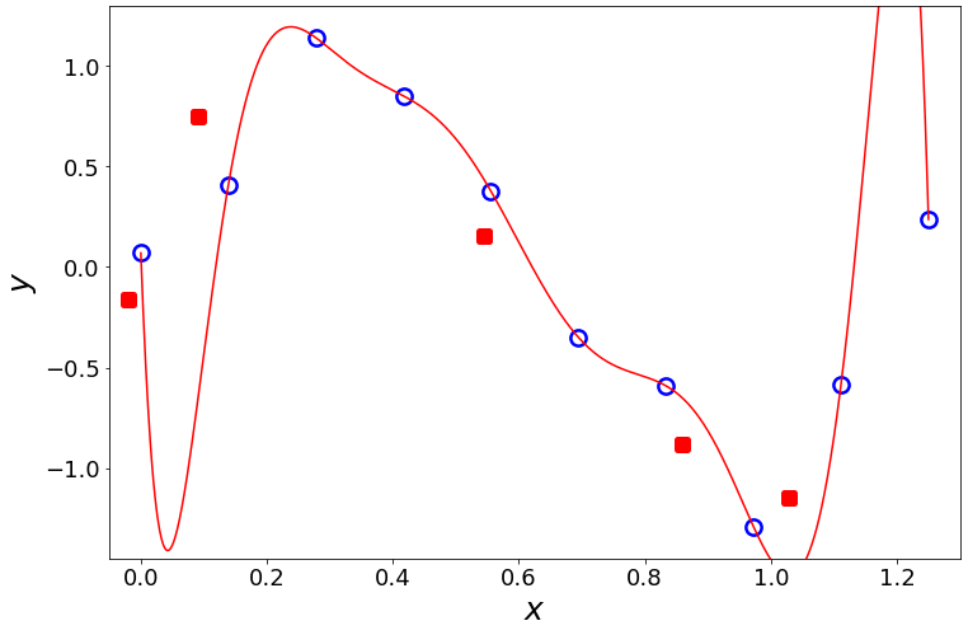
Example of Stratified sampling:

D has 500 positive &
500 negative samples

70% for training:
sample 350 positive &
350 negative for D_{train}

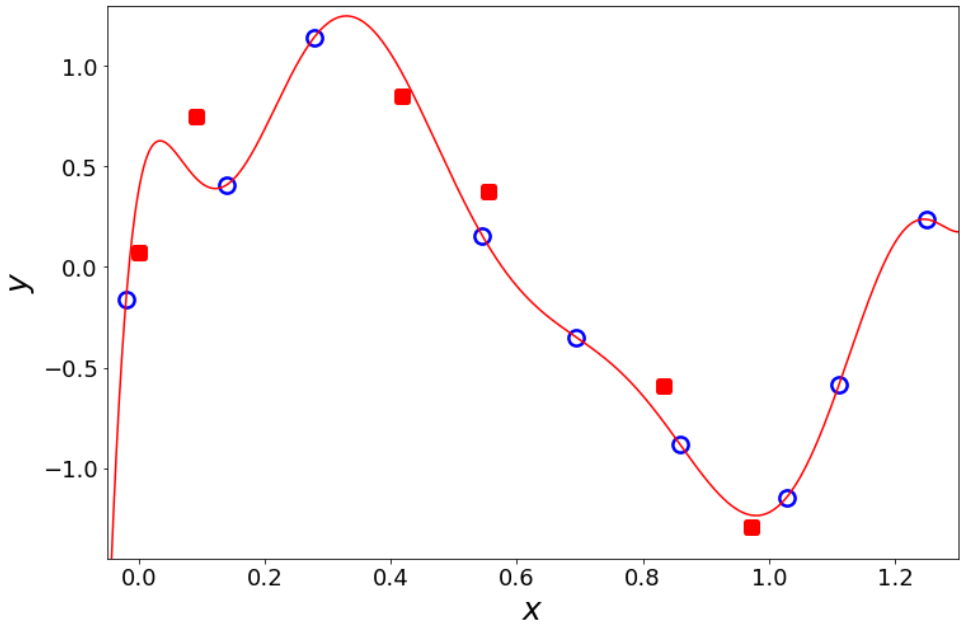
- Randomly split data into training and testing sets (e.g., 70% for training and 30% for testing)
- The training/test split should preserve a consistent distribution (use stratified sampling).
- Performance of a model must be evaluated in a held-out testing set.
- The test dataset should NEVER be used to train the model.

Limitations of Hold-out Method



MSE (training) = 0

MSE (testing) = 136.76



MSE (training) = 0

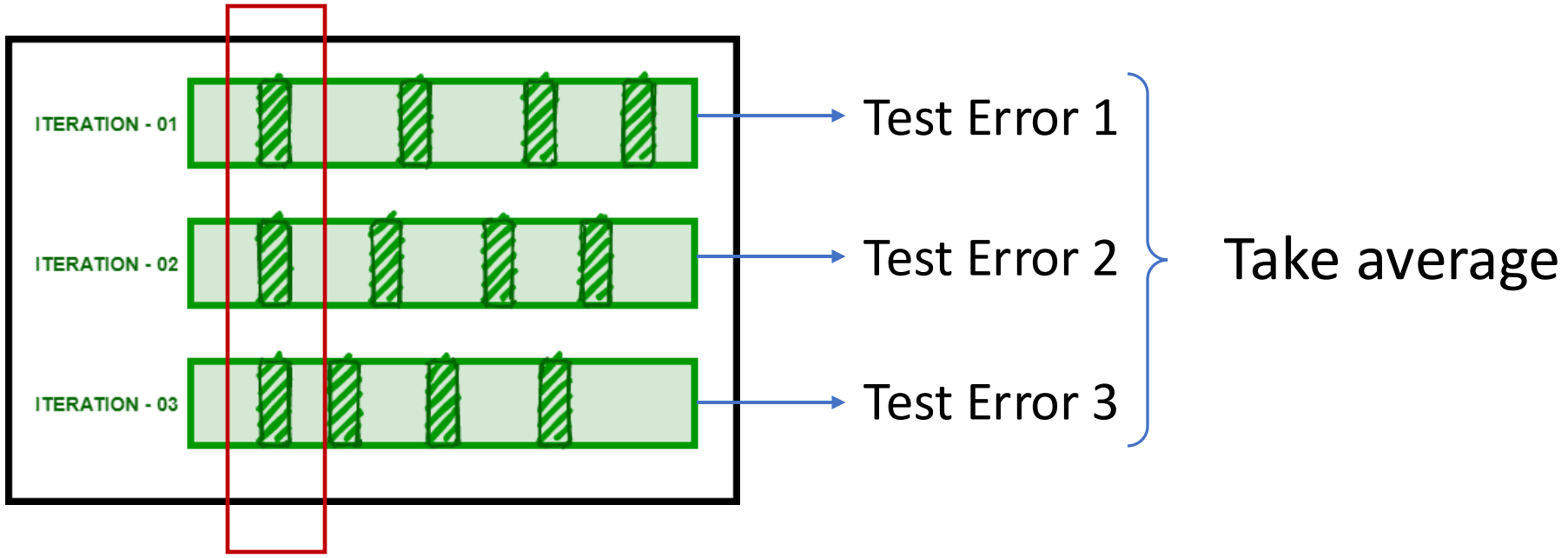
MSE (test) = 0.0537

Same data points with two different data split

By chance, we could get small testing error even for a model that overfits in a particular training/test split.

Repeated Hold-out Method for Performance Evaluation

Idea: Repeat the pipeline for K times,
then take average of the performance over the test set.



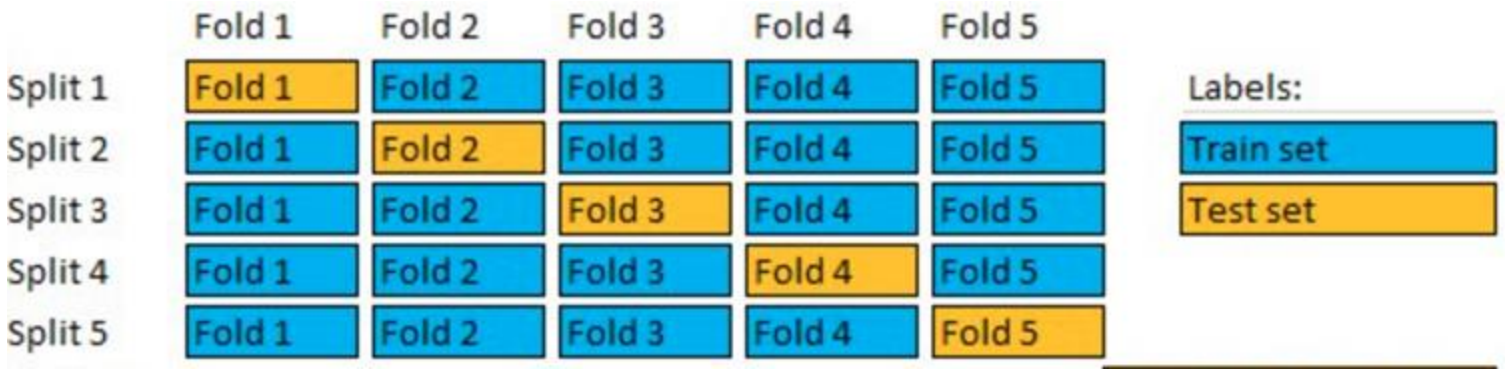
Possible to have overlaps

Even the model overfits to a particular training/test split, it affects little the final average performance.

K-Fold Cross Validation for Performance Evaluation

Idea: Split the data into K subsets of equal size, use one subset for testing and others for training in each iteration.

(commonly used K: 5, 10, 20)



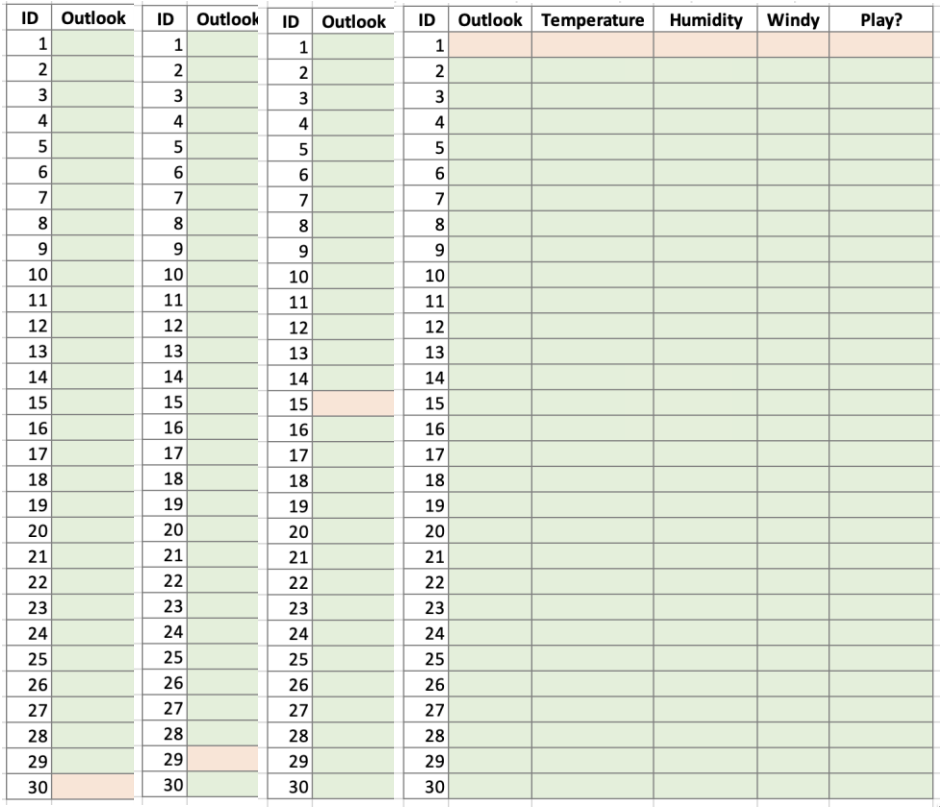
$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

Avoids overlapping test set.

A more systematical way of performance evaluation

A Special Case: Leave-One-Out Method

Idea: When $K = |D_{train}|$ for the K-fold cross validation, we call it leave-one-out method. (each subset only contains one sample)



Iterations: 1 2 i N

Iteration 1: Train model with $N-1$ data, compute test error E_1 with the remaining one

Iteration i: Train model with $N-1$ data, compute test error E_i with the remaining one

Final error: $E = \frac{1}{N}\sum_{i=1}^{N} E_i$

A Special Case: Leave-One-Out Method

Idea: When $K = N$ (size of D) for the K-fold cross validation, we call it leave-one-out method. (each subset only contains one sample)

- Advantage:
  - Model is trained using $N - 1$ data points, close to that trained using D. *(our goal: evaluating how good the model is trained using D)*
  - Therefore, it is more accurate measurement of the performance.
- Disadvantage:
  - Computationally expensive! Need to train the model for N times. *(N can be greater than 1 million for large datasets)*

How to Prevent Overfitting?

Recall the two major reasons of overfitting:

- Data is not enough or is not representative.
  → *Improve quality of training data / collect more data.*
- The model is too complex.
  → *Reduce model complexity (use simpler models).*
  → *Use regularization methods.*

Common regularizations :

- L1 regularization (*Ridge regression*): $\min_{\mathbf{w}} TrainLoss(\mathbf{w}) + \lambda\|\mathbf{w}\|_1$
- L2 regularization (*Lasso regression*): $\min_{\mathbf{w}} TrainLoss(\mathbf{w}) + \lambda\|\mathbf{w}\|_2^2$
- We can also use both: $\min_{\mathbf{w}} TrainLoss(\mathbf{w}) + \lambda_1\|\mathbf{w}\|_1 + \lambda_2\|\mathbf{w}\|_2^2$

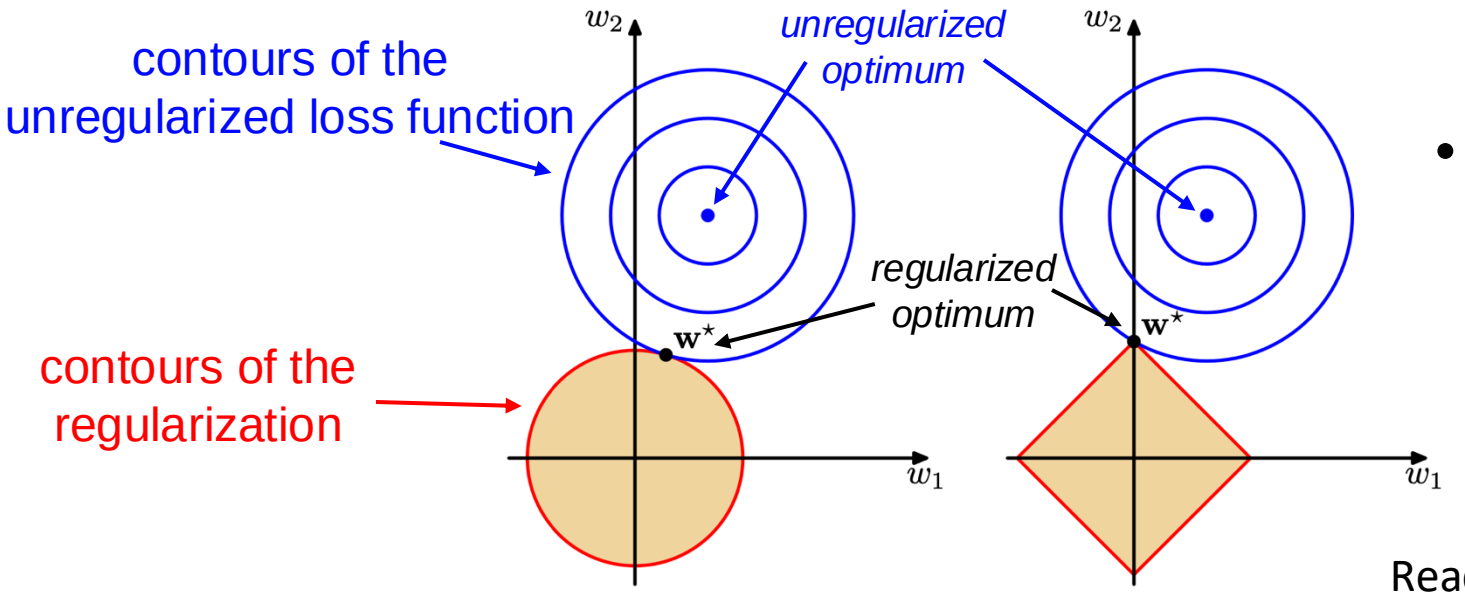
Original loss function

L1 norm

square of L2 norm

Regularizations

- L2 regularization: $\|\mathbf{w}\|_2^2 = |w_1|^2 + |w_2|^2 + \cdots + |w_d|^2$
  - *"spread out" the weights, more robust to error in individual features.*
- L1 regularization: $\|\mathbf{w}\|_1 = |w_1| + |w_2| + \cdots + |w_d|$
  - *encourages sparsity, useful for feature selection.*



- Sacrificing the accuracy over training data for better generalization.
- How to determine λ?
  - They are hyperparameters.
  - Do hyperparameter tuning!
  - Methods: *grid search*, random search, Bayesian search

Read more:
https://scikit-learn.org/stable/modules/grid_search.html

Figure adopted from Fig. 3.4 of Christopher M. Bishop, *Pattern Recognition and Machine Learning.*

Performance Measures

How to compare the performance of two models?

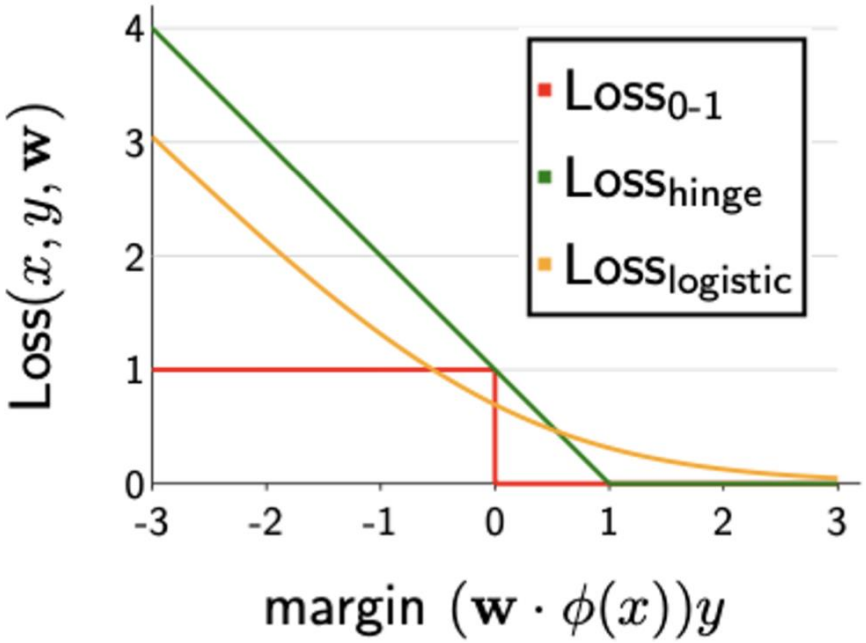
So far, we only used training loss/error and test loss/error.

For regression: we can use the MSE to measure the performance of different models.

How about classification?

- Logistic regression: logistic loss
- SVM: hinge loss
- Decision tree

We need standard performance metrics



Confusion Matrix for Classification Problems

For each model, we can construct a confusion matrix:

	Predicted Positive	Predicted Negative
Ground-truth Positive	TP	FN
Ground-truth Negative	FP	TN

Confusion matrix for binary classification

TP: number of true positives
TN: number of true negatives

FP: number of false positives
FN: number of false negatives

$N = TP + TN + FP + FN$:
total number of data samples

Confusion Matrix for Classification Problems

Example: Construct a fusion matrix given the following test set and the model predictions

Outlook	Other features	Play?	Prediction	
sunny	…	No	Yes	*false positive*
sunny	…	No	No	*true negative*
overcast	…	Yes	Yes	*true positive*
rain	…	Yes	No	*false negative*
rain	…	Yes	No	*false negative*
rain	…	No	Yes	*false positive*
overcast	…	Yes	Yes	*true positive*
sunny	…	No	No	*true negative*
sunny	…	Yes	No	*false negative*
rain	…	Yes	Yes	*true positive*

	Predicted Positive	Predicted Negative
Ground-truth Positive	$TP = 3$	$FN = 3$
Ground-truth Negative	$FP = 2$	$TN = 2$

$$N = TP + TN + FP + FN = 10$$

Evaluation Metrics for Binary Classification Problems

Given this confusion matrix, we can define some evaluation metrics:

	Predicted Positive	Predicted Negative
Ground-truth Positive	TP	FN
Ground-truth Negative	FP	TN

Number of samples that are correctly predicted

1. Accuracy:

$$\mathrm{acc}(f; D) = \frac{TP + TN}{N}$$

The ratio of correct samples

Evaluation Metrics for Binary Classification Problems

Given this confusion matrix, we can define some evaluation metrics:

	Predicted Positive	Predicted Negative
Ground-truth Positive	TP	FN
Ground-truth Negative	FP	TN

Number of samples that are predicted to be positive

Number of samples that are really positive (ground-truth)

2. Precision:

$$\text{Precision}(f; D) = \frac{TP}{TP + FP}$$

Out of the predicted positives, how many are really positive?

3. Recall:

$$\text{Recall}(f; D) = \frac{TP}{TP + FN}$$

Out of the positive samples, how many are predicted positive?

Evaluation Metrics for Binary Classification Problems

Precision and Recall are contradictory

Generally, when the recall is high, the precision is often low, and vise versa.

4. F1 Score:

$$\text{Precision}(f; D) = \frac{TP}{TP + FP} \qquad \text{Recall}(f; D) = \frac{TP}{TP + FN}$$

$$\text{F1}(f; D) = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \times TP}{N + TP - TN}$$

A combination of precision and recall

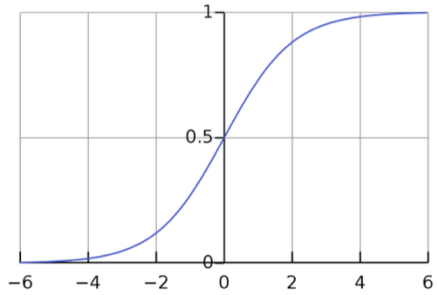
F1 is the harmonic mean of precision and recall: $\frac{1}{F1} = \frac{1}{2}\left(\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}\right)$

Evaluation Metrics for Binary Classification Problems

However, the outcome of our predictors are continuous "scores"

Outlook	Other features	Play?	Score
sunny	…	No	0.96
sunny	…	No	0.11
overcast	…	Yes	0.76
rain	…	Yes	0.37
rain	…	Yes	0.42
rain	…	No	0.28
overcast	…	Yes	0.77
sunny	…	No	0.03
sunny	…	Yes	0.39
rain	…	Yes	0.56

Thresholding



E.g., in logistic regression:

Score ≥ 0.5: positive

Score < 0.5: negative

- Different thresholds give different evaluation results.
- 0.5 is often a good choice but may not always be the case.

Prediction
Yes
No
Yes
No
No
Yes
Yes
No
No
Yes

Evaluation Metrics for Binary Classification Problems

In practice, we sort the predicted "scores" in descending order

#	Score
1	0.96
2	0.11
3	0.76
4	0.37
5	0.42
6	0.28
7	0.77
8	0.03
9	0.39
10	0.56

sorting

Samples more likely to be positive

Samples more likely to be negative

#	Score
1	0.96
7	0.77
3	0.76
10	0.56
5	0.42
9	0.39
4	0.37
6	0.28
2	0.11
8	0.03

Before cut point: Positive predictions

cut point

After cut point: Negative predictions

Evaluation Metrics for Binary Classification Problems

True Positive Rate and False Positive Rate

#	Label	Score	
1	Yes	0.96	+
7	No	0.77	+
3	Yes	0.76	+
			cut point
10	No	0.56	-
5	Yes	0.42	-
9	Yes	0.39	-
4	No	0.37	-
6	No	0.28	-
2	No	0.11	-
8	No	0.03	-

sorted predictions

	Predicted Positive	Predicted Negative
Ground-truth Positive	TP	FN
Ground-truth Negative	FP	TN

True Positive Rate: $\text{TPR} = \frac{TP}{TP+FN}$

False Positive Rate: $\text{FPR} = \frac{FP}{FP+TN}$

Exercise: Compute TPR and FPR for the example shown in left

TP=2, FN=2, FP=1, TN=5. Therefore, TPR=2/4 and FPR=1/6

Evaluation Metrics for Binary Classification Problems

Receiver Operating Characteristic (ROC) Curve:

#	Label	Score	
		cut point	
1	Yes	0.96	-
7	No	0.77	-
3	Yes	0.76	-
10	No	0.56	-
5	Yes	0.42	-
9	Yes	0.39	-
4	No	0.37	-
6	No	0.28	-
2	No	0.11	-
8	No	0.03	-

sorted predictions

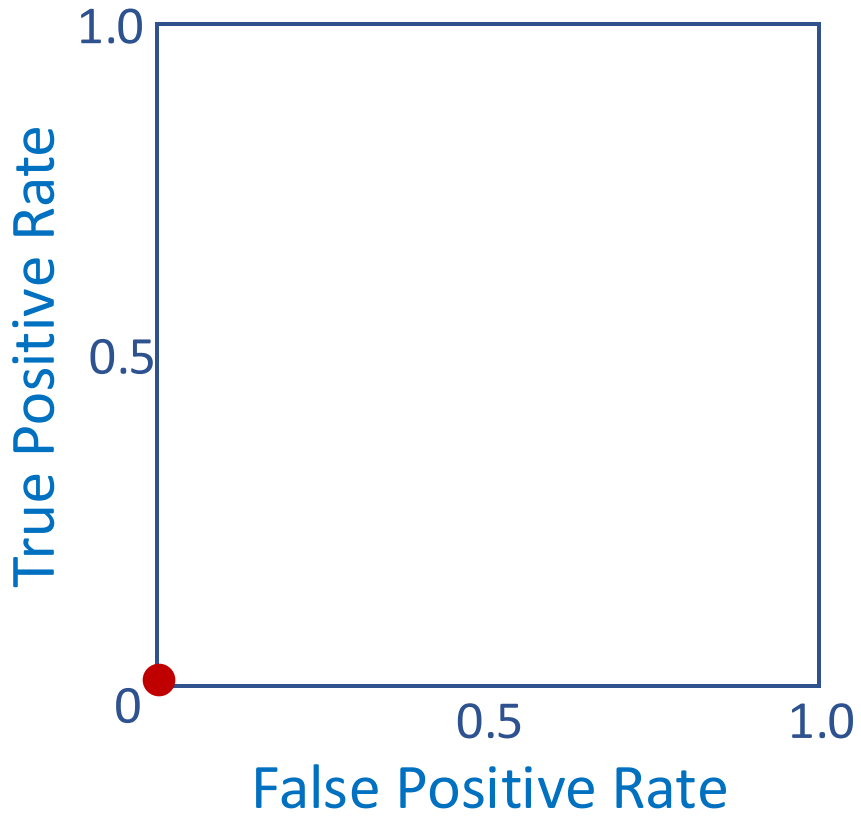
For each cut point, we can plot its corresponding TPR and FPR in a figure:

TP=0; FN=4

FP=0; TN=6

$$\text{TPR} = \frac{TP}{TP + FN} = 0$$

$$\text{FPR} = \frac{FP}{FP + TN} = 0$$



Evaluation Metrics for Binary Classification Problems

Receiver Operating Characteristic (ROC) Curve:

#	Label	Score	
1	Yes	0.96	+
7	No	0.77	-
3	Yes	0.76	-
10	No	0.56	-
5	Yes	0.42	-
9	Yes	0.39	-
4	No	0.37	-
6	No	0.28	-
2	No	0.11	-
8	No	0.03	-

cut point

sorted predictions

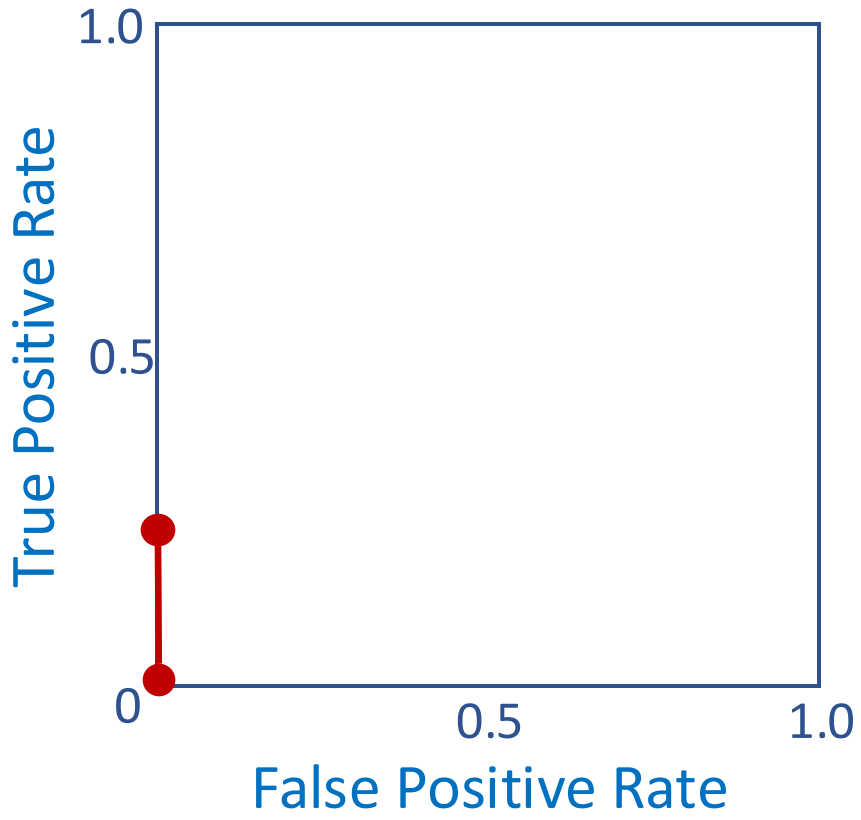
TP=1; FN=3

FP=0; TN=6

$$\text{TPR} = \frac{TP}{TP + FN} = 0.25$$

$$\text{FPR} = \frac{FP}{FP + TN} = 0$$

For each cut point, we can plot its corresponding TPR and FPR in a figure:



Evaluation Metrics for Binary Classification Problems

Receiver Operating Characteristic (ROC) Curve:

#	Label	Score	
1	Yes	0.96	+
7	No	0.77	+
			cut point
3	Yes	0.76	-
10	No	0.56	-
5	Yes	0.42	-
9	Yes	0.39	-
4	No	0.37	-
6	No	0.28	-
2	No	0.11	-
8	No	0.03	-

sorted predictions

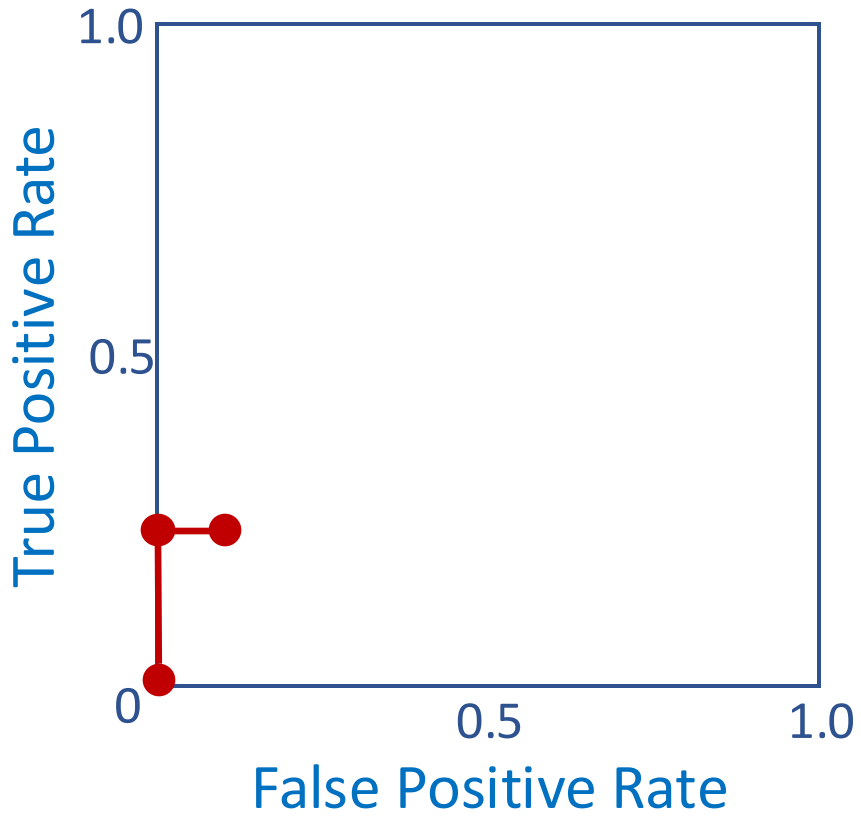
TP=1; FN=3

FP=1; TN=5

$$\text{TPR} = \frac{TP}{TP + FN} = 0.25$$

$$\text{FPR} = \frac{FP}{FP + TN} = 0.17$$

For each cut point, we can plot its corresponding TPR and FPR in a figure:



Evaluation Metrics for Binary Classification Problems

Receiver Operating Characteristic (ROC) Curve:

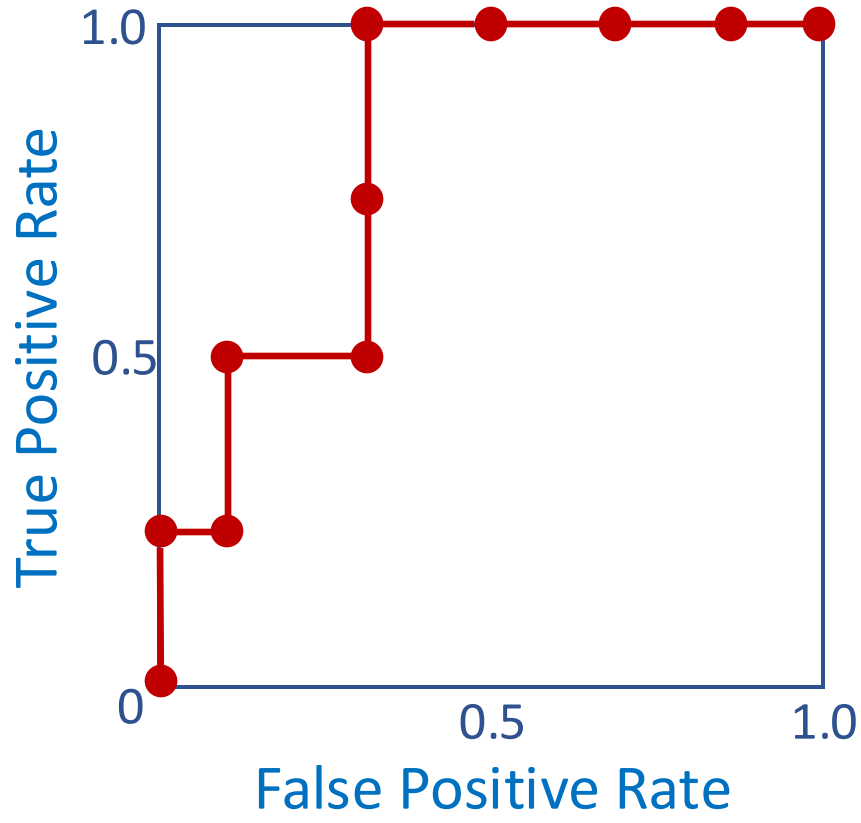
#	Label	Score	
1	Yes	0.96	+
7	No	0.77	+
3	Yes	0.76	+
10	No	0.56	+
5	Yes	0.42	+
9	Yes	0.39	+
4	No	0.37	+
6	No	0.28	+
2	No	0.11	+
8	No	0.03	+

cut point

sorted predictions

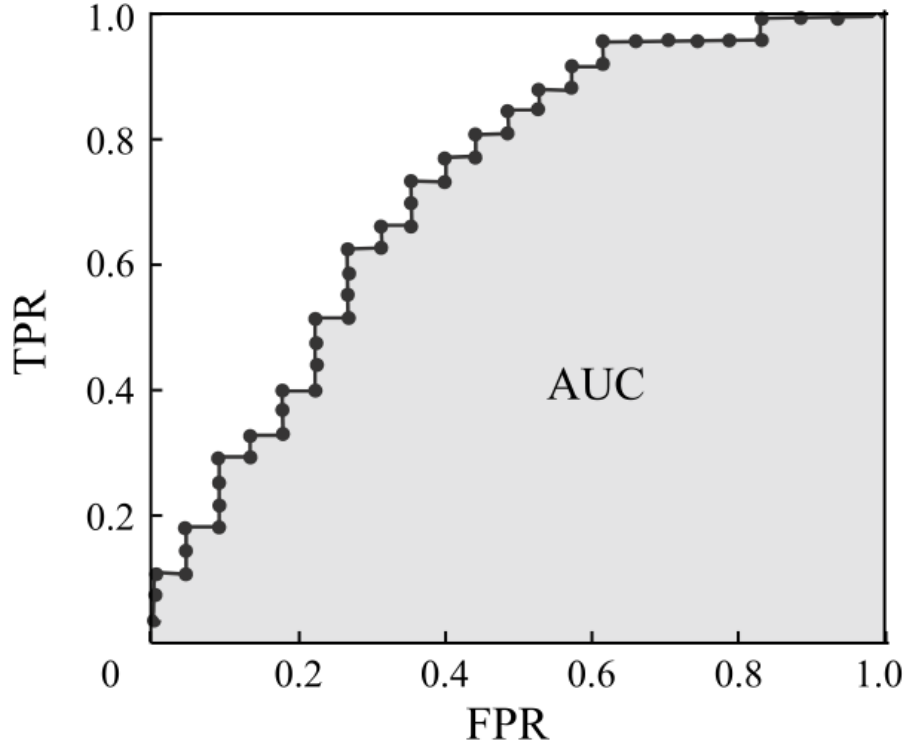
Continue this process:

For each cut point, we can plot its corresponding TPR and FPR in a figure:



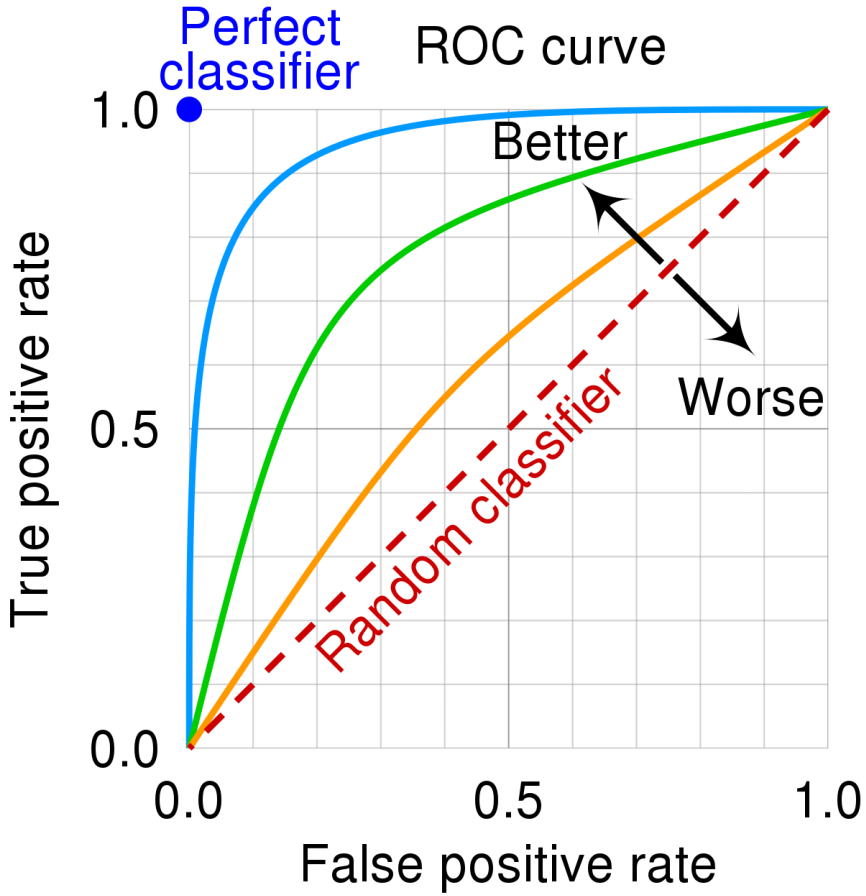
Evaluation Metrics for Binary Classification Problems

ROC and AUC Score



We can get a smother curve with more samples

AUC Score: Area under the ROC Curve



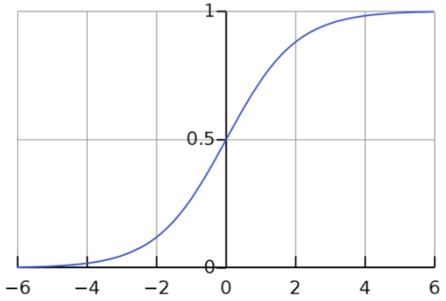
AUC: The larger, the better

Multiclass Classification: Methods and Evaluation

Recap: Logistic Regression for Binary Classification

Consider this transfromation retruns a "probability":

- $p(y = +1|x) = \sigma(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^\top \mathbf{x})}$
- $p(y = -1|x) = 1 - \sigma(\mathbf{w}^\top \mathbf{x}) = \frac{\exp(-\mathbf{w}^\top \mathbf{x})}{1+\exp(-\mathbf{w}^\top \mathbf{x})} = \frac{1}{1+\exp(+\mathbf{w}^\top \mathbf{x})}$



- The *log likelihood* (numerically more stable):

$$\ell(\mathbf{w}) = \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} \mathbb{1}[y_i = +1] \log p(y_i = +1|\mathbf{x}) + \mathbb{1}[y_i = -1] \log p(y_i = -1|\mathbf{x})$$

- We can maximize the likelihood, or equivalently, minimize the negative likelihood:

$$\text{Loss}(\mathbf{x}, y, \mathbf{w}) = \log(1 + \exp(-(\mathbf{w}^\top \mathbf{x})y))$$

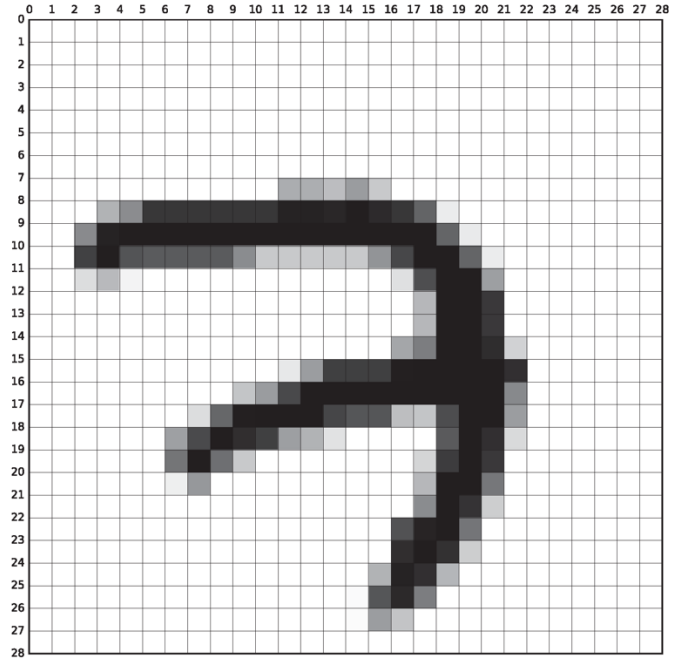
Features			Label
surface	texture	density	ripe?
hard	clear	0.77	yes
hard	clear	0.56	yes
hard	slightly blurry	0.64	no
hard	clear	0.61	yes
hard	clear	0.63	yes
hard	slightly blurry	0.66	no
hard	slightly blurry	0.67	no
hard	clear	0.44	yes
soft	clear	0.24	no
soft	clear	0.40	yes
hard	blurry	0.25	no
soft	blurry	0.34	no
soft	slightly blurry	0.48	yes

Binary class labels: Yes(+)/No(-)

Multiclass Classification Applications

- Hand-written digit recognition

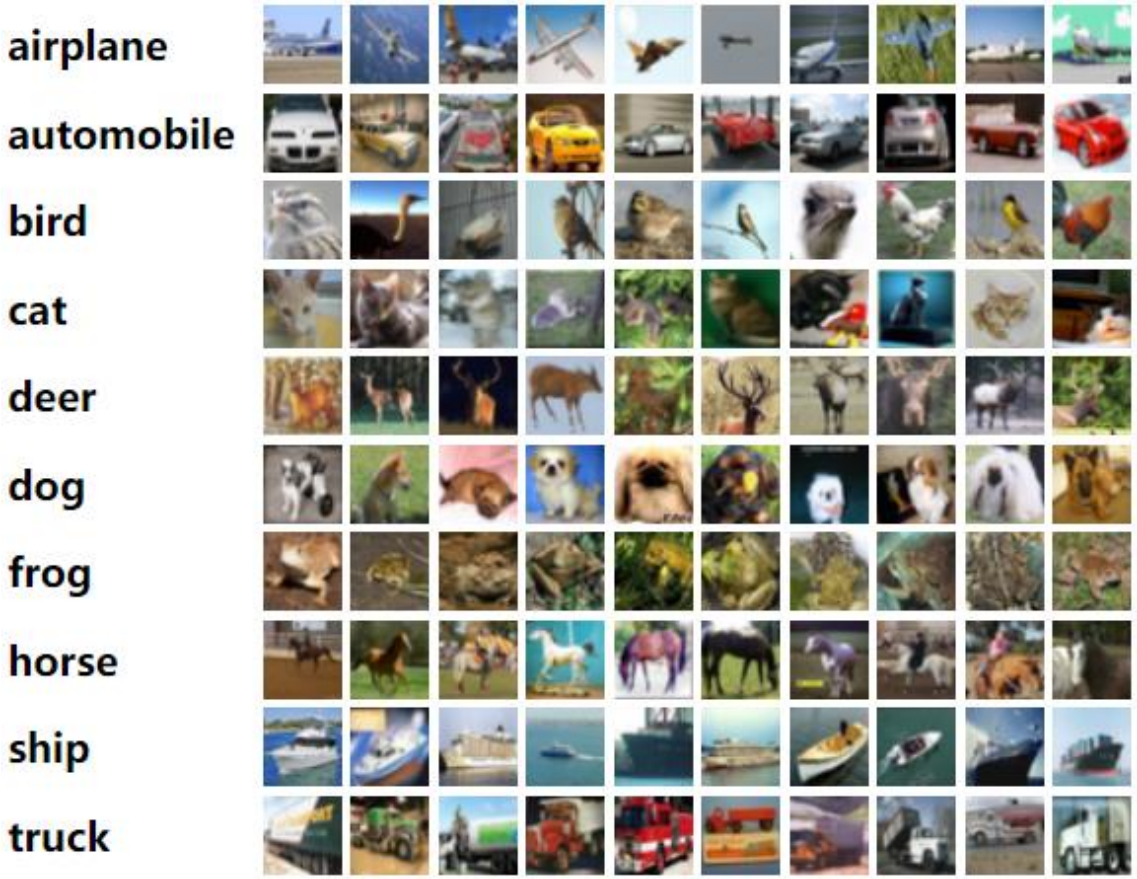
10 classes

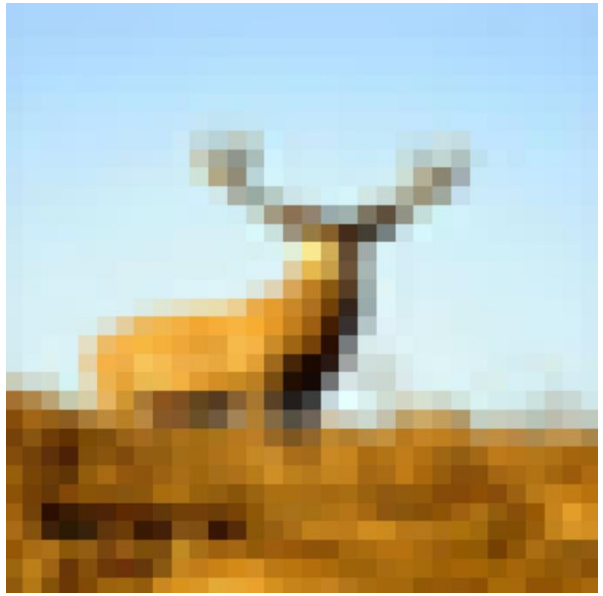


A sample in test set

Multiclass Classification Applications

- Image classification





A sample in test set

Multiclass Classification

- Some algorithms can be directly applied to multiclass classification
  - Example: Decision Tree

- More often, we extend binary classification methods to multiclass classification.

- Decomposition: dividing the multiclass classification problem into several binary classification problems.

```
ID3(D,X) =
  Let T be a new tree
  If all instances in D have same class c
    Label(T) = c; Return T
  If X = ∅ or no attribute has positive information gain
    Label(T) = most common class in D; return T
  X ← attribute with highest information gain
  Label(T) = X
  For each value x of X
    D_x ← instances in D with X = x
    If D_x is empty
      Let T_x be a new tree
      Label(T_x) = most common class in D
    Else
      T_x = ID3(D_x, X − {X})
    Add a branch from T to T_x labeled by x
  Return T
```

Decomposition-based Multiclass Classification

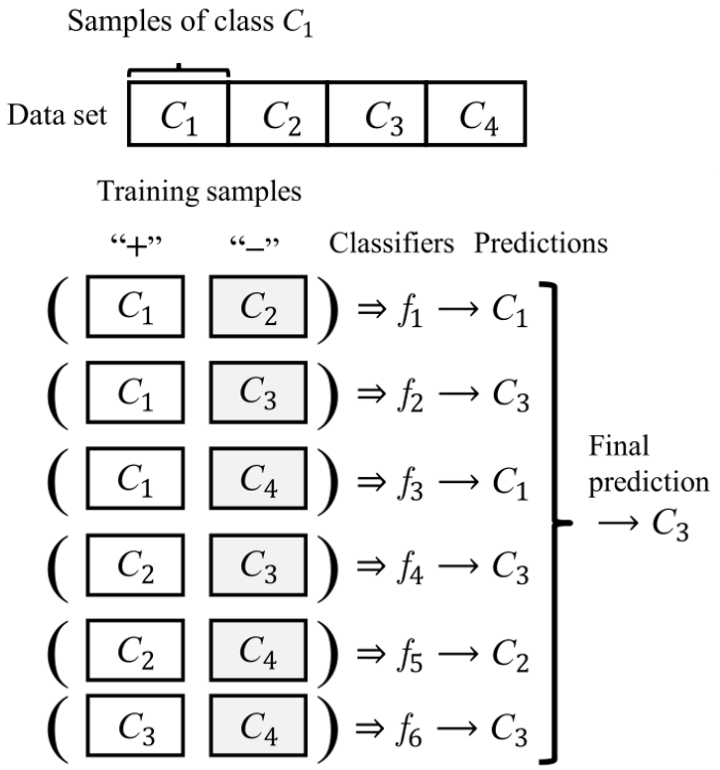
- Consider N classes: C_1, C_2, …, C_N.
- Decomposition: dividing the multiclass classification problem into several binary classification problems.
- In testing phase, we ensemble the outputs collected from all binary classifiers into the final multiclass predictions.
- **Key question: How to divide and how to ensemble.**
- We focus on two classical dividing strategies:
  One vs One (OvO) & **One vs Rest (OvR)**

Decomposition-based Multiclass Classification

- Given a dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$, where $y_i \in \{C_1, C_2, \ldots, C_N\}$
- One vs One (OvO) puts the N classes into pairs: in total $N(N-1)/2$ binary classification problems.

Example: OvO trains a classifier to distinguish class C_i and C_j , where it regards C_i as positive and C_j as negative.

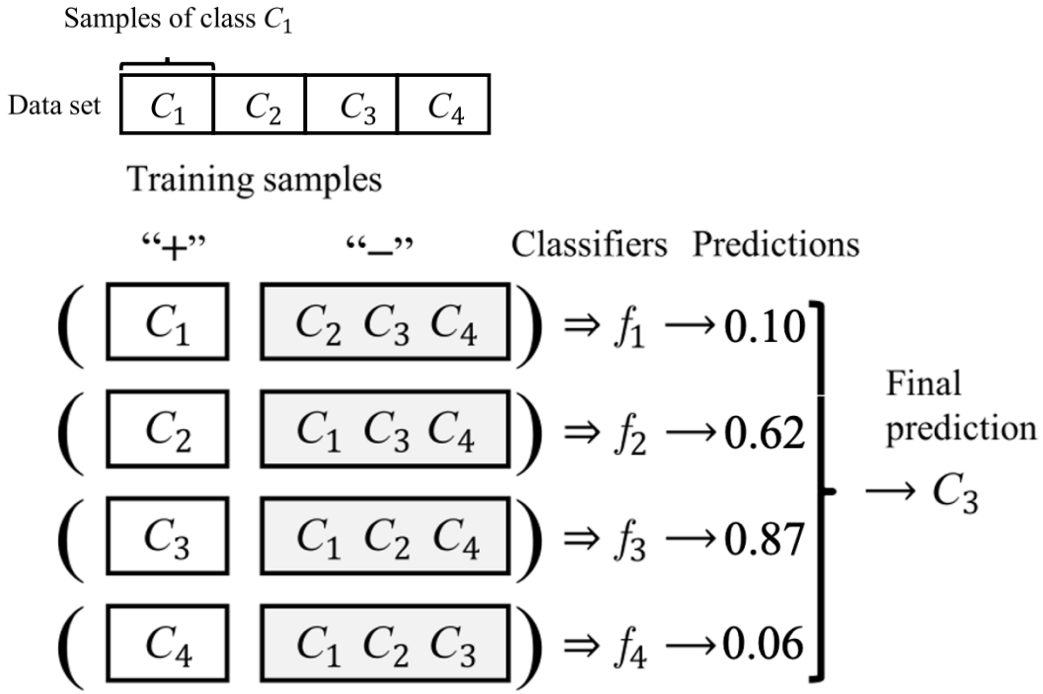
OvO strategy



Decomposition-based Multiclass Classification

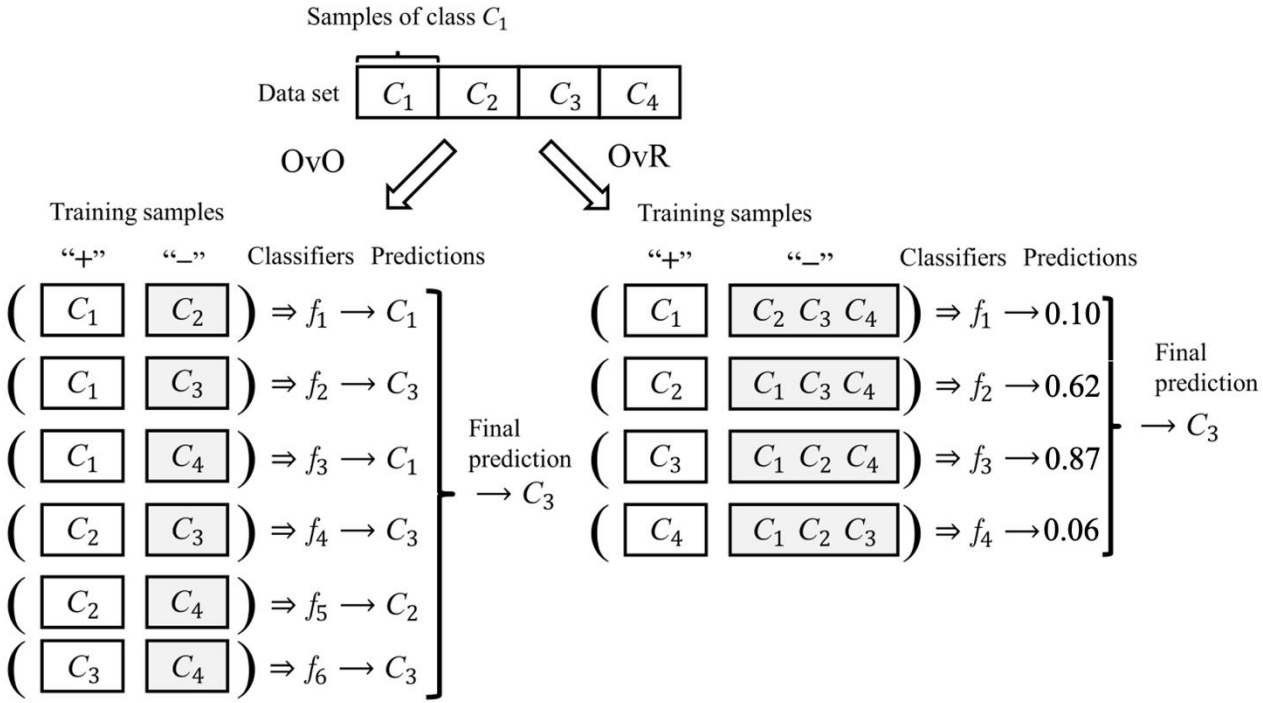
- Given a dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$, where $y_i \in \{C_1, C_2, \ldots, C_N\}$
- OvR trains N classifiers by considering each class as positive in turn, and the rest classes are considered as negative.

OvR strategy



Decomposition-based Multiclass Classification

- Given a dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$, where $y_i \in \{C_1, C_2, \ldots, C_N\}$



- OvR trains only N classifier but OvO trains $N(N-1)/2$ classifier.
- OvO is usually more expansive in storage and test time.
- In training, OvO uses data of two classes, but OvR uses all data. Usually, OvO can be trained faster.
- Performance are usually similar.

Evaluation Methods for Multiclass Classification

- Confusion matrix for multiclass classification

No	Actual	Predicted
1	Airplane	Airplane
2	Car	Boat
3	Car	Car
4	Car	Car
5	Car	Boat
6	Airplane	Boat
7	Boat	Boat
8	Car	Airplane
9	Airplane	Airplane
10	Car	Car

Three labels: Airplane, Car, Boat

		Predicted		
		Airplane	Boat	Car
Actual	Airplane	2	1	0
	Boat	0	1	0
	Car	1	2	3

Number of images of cars being predicted as boat

Confusion matrix: rows are actual labels (ground truth), columns are predictions

Example and figures from https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f

Evaluation Methods for Multiclass Classification

- We can compute the metrics for each class

		Predicted		
		Airplane	Boat	Car
Actual	Airplane	2	1	0
	Boat	0	1	0
	Car	1	2	3

Label	True Positive (TP)	False Positive (FP)	False Negative (FN)	Precision	Recall	F1 Score
Airplane	2	1	1	0.67	0.67	2 * (0.67 * 0.67) / (0.67 + 0.67) = **0.67**
Boat	1	3	0	0.25	1.00	2*(0.25 * 1.00) / (0.25 + 1.00) = **0.40**
Car	3	0	3	1.00	0.50	2 * (1.00 * 0.50) / (1.00 + 0.50) = **0.67**

Number of images of other classes being predicted as boat

How to evaluate the overall performance for this multiclass classification model as a whole?

Taking averages!

- Macro average
- Weighted average
- Micro average

Example and figures from https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f

Evaluation Methods for Multiclass Classification

- Macro Average

Label	True Positive (TP)	False Positive (FP)	False Negative (FN)	Precision	Recall	F1 Score
Airplane	2	1	1	0.67	0.67	2 * (0.67 * 0.67) / (0.67 + 0.67) = **0.67**
Boat	1	3	0	0.25	1.00	2*(0.25 * 1.00) / (0.25 + 1.00) = **0.40**
Car	3	0	3	1.00	0.50	2 * (1.00 * 0.50) / (1.00 + 0.50) = **0.67**

Label	Per-Class F1 Score	Macro-Averaged F1 Score
Airplane	0.67	$\frac{0.67 + 0.40 + 0.67}{3}$ = **0.58**
Boat	0.40	
Car	0.67	

Taking the average of per-class metrics (Precision, Recall, and F1).

$$\text{Macro-Precision} = \frac{\text{Precision}_1 + \text{Precision}_2 + \cdots + \text{Precision}_N}{N}$$

$$\text{Macro-Recall} = \frac{\text{Recall}_1 + \text{Recall}_2 + \cdots + \text{Recall}_N}{N}$$

$$\text{Macro-F1} = \frac{\text{F1}_1 + \text{F1}_2 + \cdots + \text{F1}_N}{N}$$

Example and figures from https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f

Evaluation Methods for Multiclass Classification

- Weighted Average

  Taking the weighted average of per-class metrics (Precision, Recall, and F1).

		Predicted		
		Airplane	Boat	Car
Actual	Airplane	2	1	0
	Boat	0	1	0
	Car	1	2	3

Label	True Positive (TP)	False Positive (FP)	False Negative (FN)	Precision	Recall	F1 Score
Airplane	2	1	1	0.67	0.67	2 * (0.67 * 0.67) / (0.67 + 0.67) = **0.67**
Boat	1	3	0	0.25	1.00	2*(0.25 * 1.00) / (0.25 + 1.00) = **0.40**
Car	3	0	3	1.00	0.50	2 * (1.00 * 0.50) / (1.00 + 0.50) = **0.67**

Label	Per-Class F1 Score	Support	Support Proportion	Weighted Average F1 Score
Airplane	0.67	3	0.3	$(0.67 * 0.3) + (0.40 * 0.1) + (0.67 * 0.6)$ = **0.64**
Boat	0.40	1	0.1	
Car	0.67	6	0.6	
Total	-	10	1.0	

frequency of actual occurrence of each class

$$\text{Weighted-Precision} = \frac{p_1\text{Precision}_1 + \cdots + p_N\text{Precision}_N}{N}$$

$$\text{Weighted-Recall} = \frac{p_1\text{Recall}_1 + \cdots + p_N\text{Recall}_N}{N}$$

$$\text{Weighted-F1} = \frac{p_1\text{F1}_1 + \cdots + p_N\text{F1}_N}{N}$$

Example and figures from https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f

Evaluation Methods for Multiclass Classification

- Micro Average

Label	True Positive (TP)	False Positive (FP)	False Negative (FN)	Precision	Recall	F1 Score
Airplane	2	1	1	0.67	0.67	2 * (0.67 * 0.67) / (0.67 + 0.67) = **0.67**
Boat	1	3	0	0.25	1.00	2*(0.25 * 1.00) / (0.25 + 1.00) = **0.40**
Car	3	0	3	1.00	0.50	2 * (1.00 * 0.50) / (1.00 + 0.50) = **0.67**

Label	True Positive (TP)	False Positive (FP)	False Negative (FN)	Micro-Averaged Values
Airplane	2	1	1	Precision = $\frac{6}{6+4}$ = **0.60**
Boat	1	3	0	Recall = $\frac{6}{6+4}$ = **0.60**
Car	3	0	3	F1 Score = $\frac{6}{6+\frac{1}{2}(4+4)}$ = **0.60**
TOTAL	**6**	**4**	**4**	

Add up the confusion matrix for each binary classification and compute the metrics

$$\text{Micro-Precision} = \frac{TP_1+\cdots+TP_N}{(TP_1+FP_1)+\cdots+(TP_N+FP_N)}$$

$$\text{Micro-Recall} = \frac{TP_1+\cdots+TP_N}{(TP_1+FN_1)+\cdots+(TP_N+FN_N)}$$

$$\text{Micro-F1} = \frac{2 \times \text{Micro-Precision} \times \text{Micro-Recall}}{\text{Micro-Precision} + \text{Micro-Recall}}$$

Example and figures from https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f

Evaluation Methods for Multiclass Classification

- Which one to use?
  - Macro averaged: all classes are equally important.
  - Weighted averaged: assigns greater weighting to classes with more samples in the dataset.
  - Micro averaged: overall performance regardless of the class.

Feature Engineering

Feature Engineering

- Recall from the summary of the linear models:

$\underbrace{\mathbf{w} \cdot \boxed{\phi(x)}}_{\text{score}}$

Two components: $\phi(x)$ and $\mathbf{w}$

	Regression	Classification
Prediction $f_{\mathbf{w}}(x)$	score	sign(score)
Relate to target y	residual $(\text{score} - y)$	margin $(\text{score}\, y)$
Loss functions	squared absolute deviation	zero-one hinge logistic
Algorithm	gradient descent	gradient descent

So far, we just used $\phi(x) = x$ and focused on learning of $\mathbf{w}$.

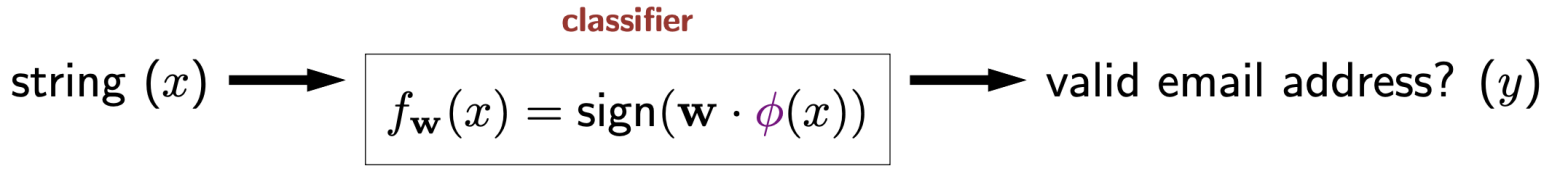
Another important part of ML pipeline:

Feature extraction (feature engineering) specifies $\phi(x)$ based on some domain knowledge.

Often the bottleneck for real applications

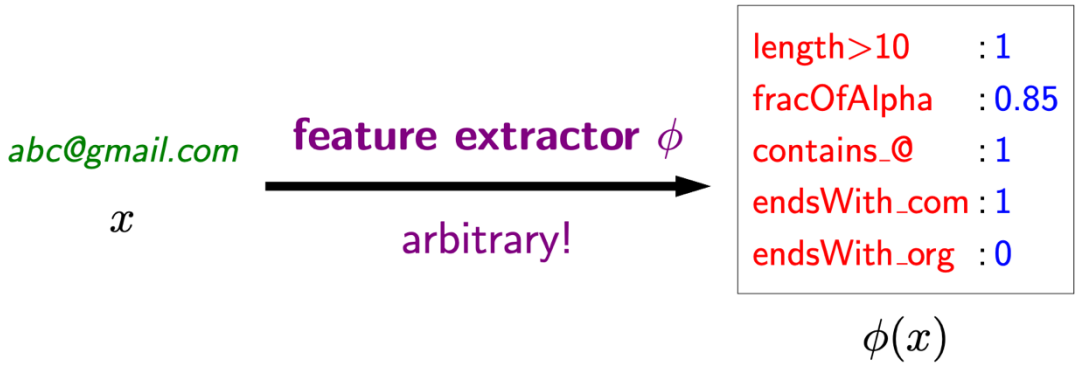
Feature Engineering

- An example task: predict whether a string is an email address



Questions: what properties of x might be relevant for predicting y?

Feature extractor ϕ produces (feature name, feature value) pairs



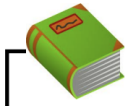
What features to include?
Need some organizational principles.

Feature engineering is sort of an "art".

Example and figures from https://stanford-cs221.github.io/spring2022-extra/modules/machine-learning/nonlinear-and-neural.pdf

Feature Engineering

- Feature templates



Definition: feature template

A **feature template** is a group of features all computed in a similar way.

abc@gmail.com → last three characters equals ___

endsWith_aaa : 0
endsWith_aab : 0
endsWith_aac : 0
...
endsWith_com : 1
...
endsWith_zzz : 0

Define types of pattern to look for, not particular patterns

Example and figures from https://stanford-cs221.github.io/spring2022-extra/modules/machine-learning/nonlinear-and-neural.pdf

Feature Engineering

- Feature templates Example

Latitude: 37.4068176

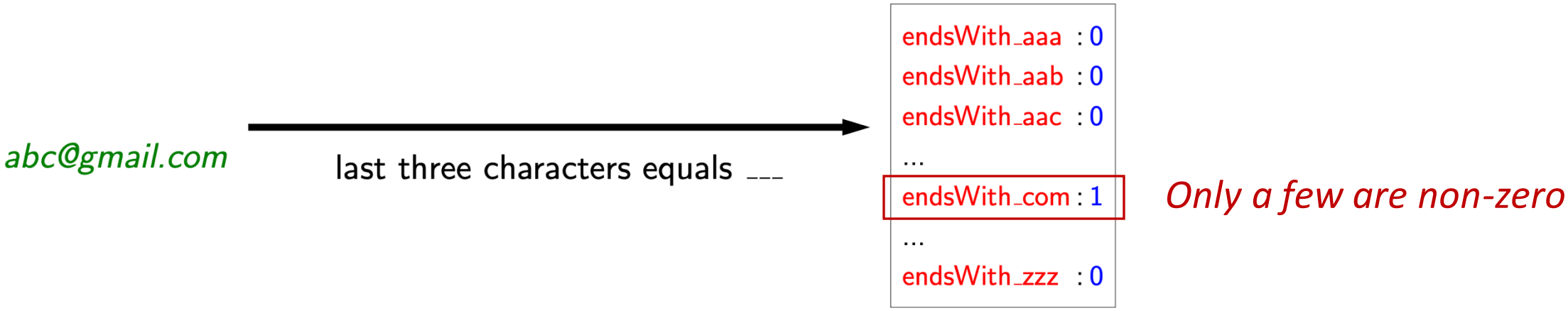
Longitude: -122.1715122

Feature template	Example feature name
Pixel intensity of image at row ___ and column ___ (___ channel)	Pixel intensity of image at row *10* and column *93* (*red* channel) : 0.8
Latitude is in [___, ___] and longitude is in [___, ___]	Latitude is in [*37.4*, *37.5*] and longitude is in [*-122.2*, *-122.1*] : 1

Example and figures from https://stanford-cs221.github.io/spring2022-extra/modules/machine-learning/nonlinear-and-neural.pdf

Sparse Features

- Dictionary is more efficient to represent sparse features (few non-zeros)



A more compact way to represent such sparse features:

(dictionary) {"endsWith_com": 1}

Instead of an array: [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]

Example and figures from https://stanford-cs221.github.io/spring2022-extra/modules/machine-learning/nonlinear-and-neural.pdf

Hypothesis Space

- Which feature extractor ϕ to use?

  - Recall that a hypothesis space (*a.k.a.* hypothesis class) is the set of possible predictors with a fixed $\phi(x)$ and varying w:

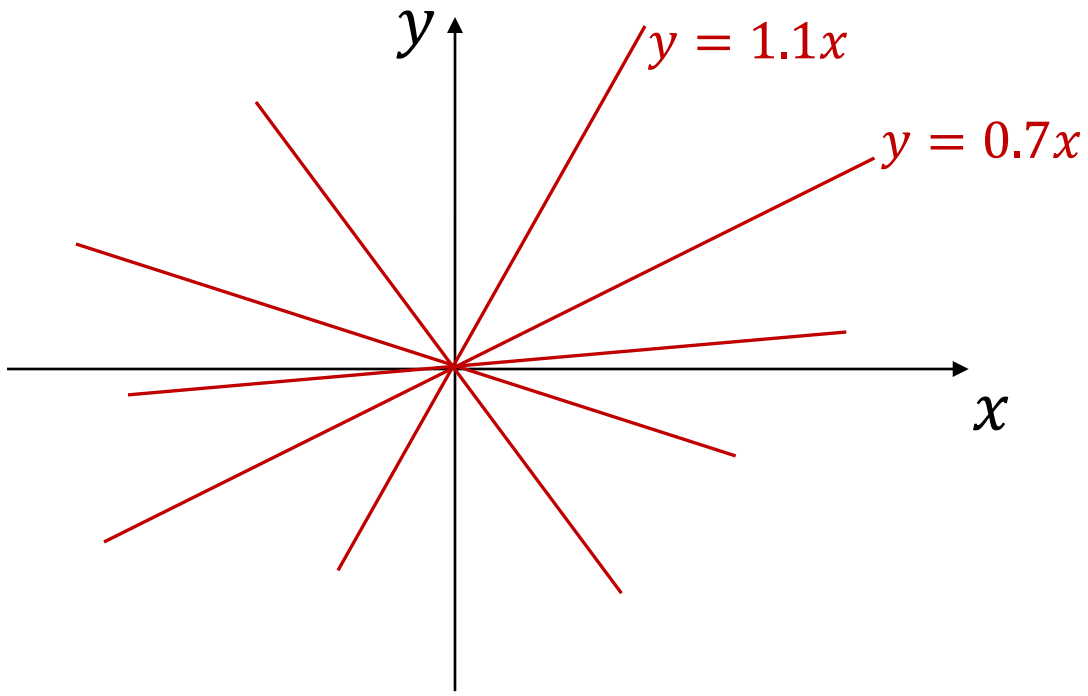
$$\mathcal{H} = \{f_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^d\}$$

  where $f_{\mathbf{w}}(x) = \mathbf{w}^{\mathrm{T}}\phi(x)$ or $\mathrm{sign}(\mathbf{w}^{\mathrm{T}}\phi(x))$ is the predictor.

Hypothesis Space

- Example of hypothesis space:
  - Linear regression: $x \in \mathbb{R}$ and $y \in \mathbb{R}$
  - Linear function: $\phi(x) = x$, the hypothesis class is $\mathcal{H}_1 = \{w_1 x : w_1 \in \mathbb{R}\}$

All lines go through the origin



Constructing a feature vector:

Considering the hypothesis class defined by this feature map.

Hypothesis Space

- Example of hypothesis space:
  - Linear regression: $x \in \mathbb{R}$ and $y \in \mathbb{R}$
  - Linear function: $\phi(x) = x$, the hypothesis class is $\mathcal{H}_1 = \{w_1 x : w_1 \in \mathbb{R}\}$

All lines go through the origin

  - Quadratic function: $\phi(x) = [x, x^2]$, the hypothesis class is

$$\mathcal{H}_2 = \{w_1 x + w_2 x^2 : w_1 \in \mathbb{R}, w_2 \in \mathbb{R}\}$$

All quadratic functions that go through the origin
Also includes all linear functions

- $\mathcal{H}_2$ is a larger set than $\mathcal{H}_1$: more expressive (can represent more things).
- The feature map ϕ defnes the hypothesis class.

Hypothesis Space

- Prediction is driven by score: $\mathbf{w}^T \phi(x)$
  - Is score linear in $\boldsymbol{w}$? *Yes!*
  - Is score linear in $\phi(x)$? *Yes!*
  - Is score linear in x? *Not necessarily!* E.g., $\phi(x) = [x, x^2]$

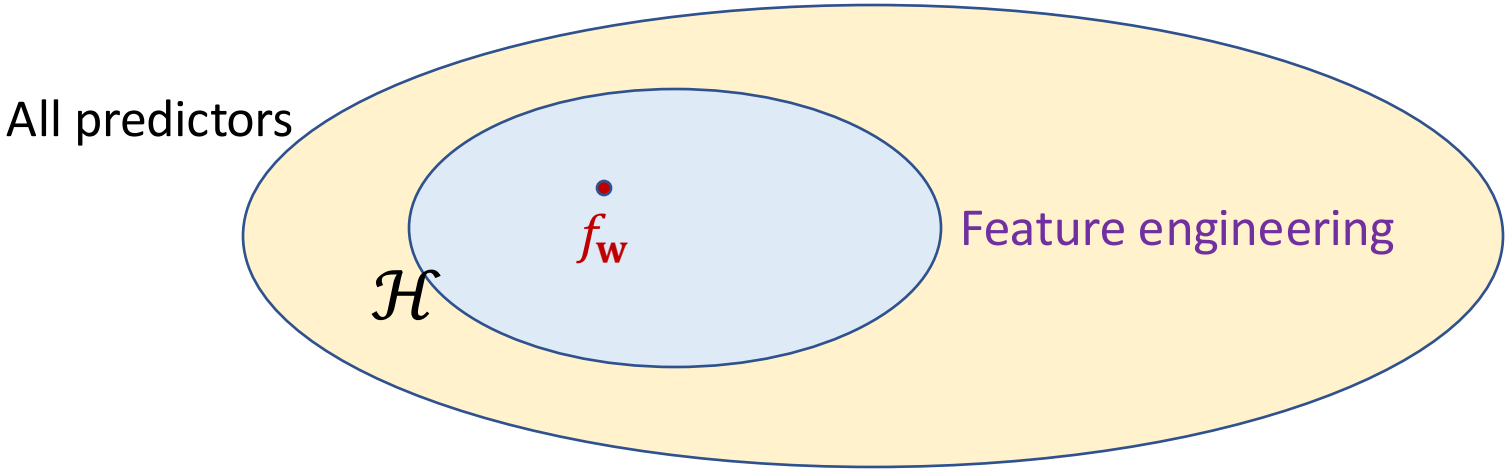
- Predictor $f_{\mathbf{w}}(x)$ can be expressive non-linear functions and decision boundaries of x.

  online demos: https://playground.tensorflow.org/
  https://www.youtube.com/watch?v=3liCbRZPrZA

- Score $\mathbf{w}^T \phi(x)$ is linear function of $\mathbf{w}$, which allows efficient learning.

Feature Engineering + Learning

$$\mathcal{H} = \{f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w}^{\mathrm{T}}\phi(x)) : \mathbf{w} \in \mathbb{R}^d\}$$

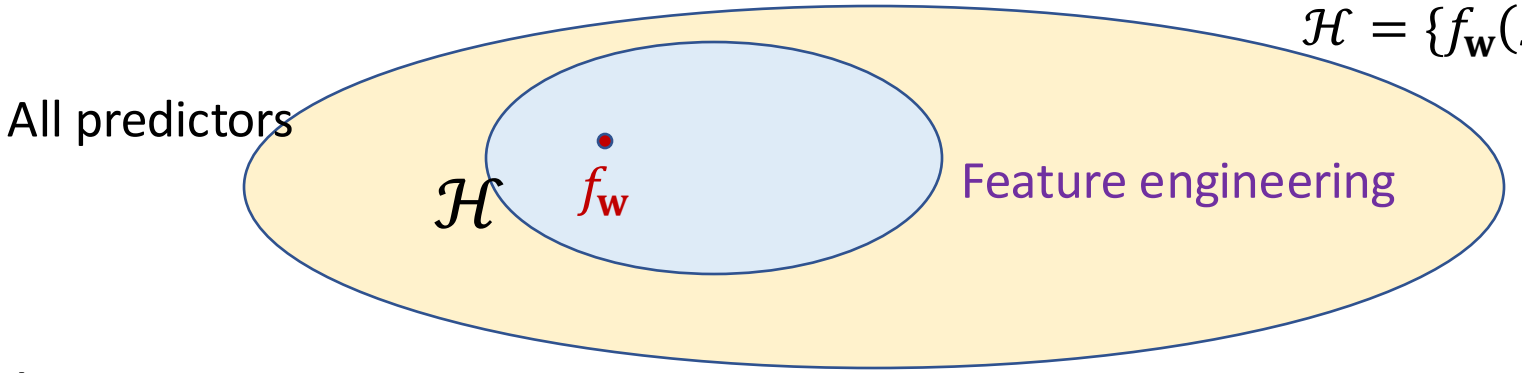


- Feature engineering/extraction: choose $\mathcal{H}$ based on domain knowledge.
- Learning: choose $f_{\mathbf{w}} \in \mathcal{H}$ based on data.

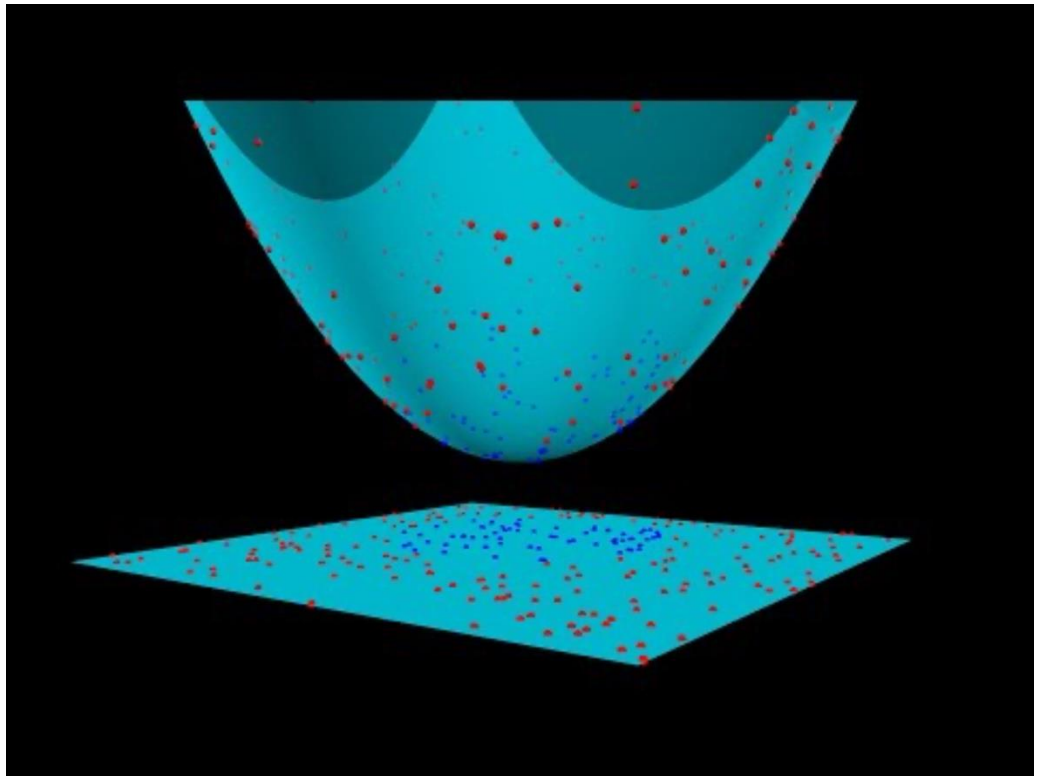
We want $\mathcal{H}$ to contain good predictors but not be too big

Feature Engineering + Learning

$$\mathcal{H} = \{f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w}^{\mathrm{T}}\phi(x)) : \mathbf{w} \in \mathbb{R}^d\}$$



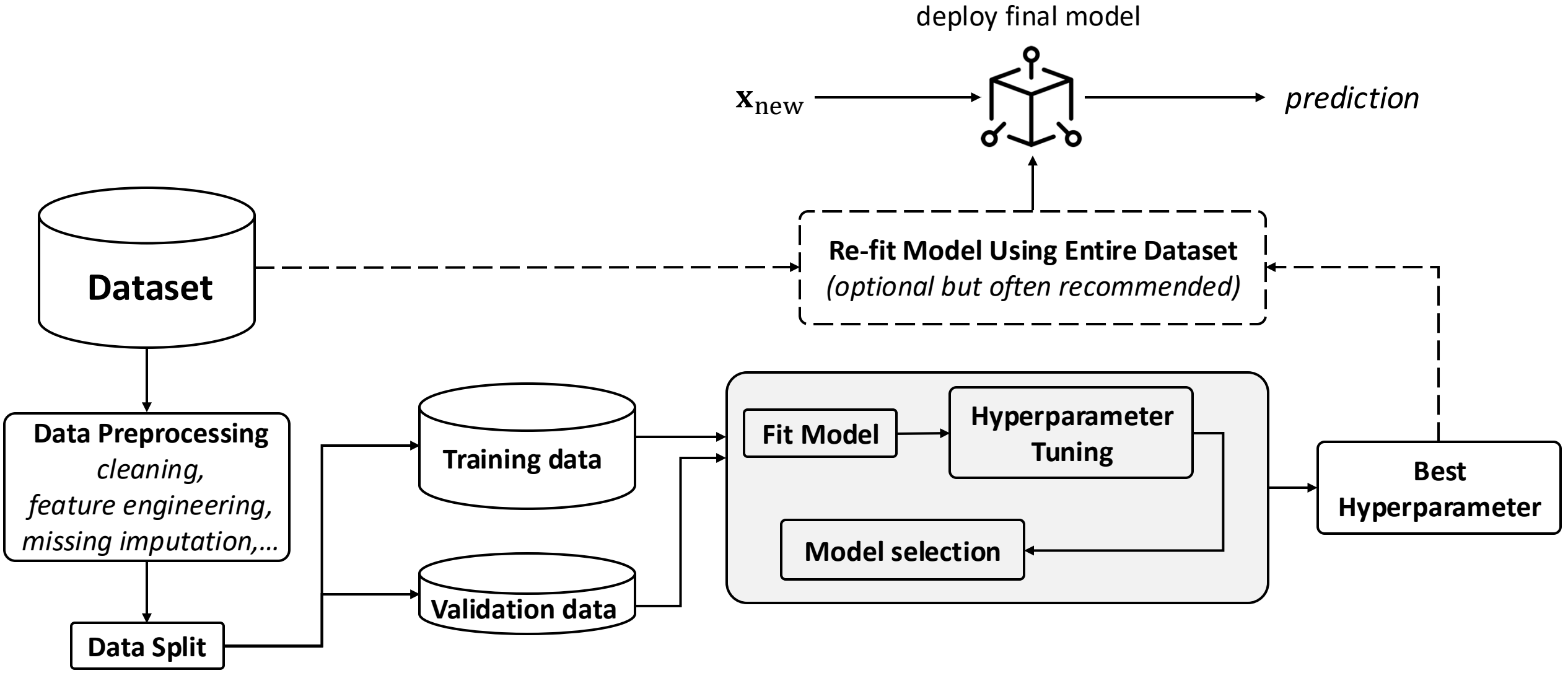
- Example:



Demo: polynomial kernel in SVM

https://www.youtube.com/embed/3liCbRZPrZA

The Typical Machine Learning Workflow



Let your voice be heard!

Thank you for your feedback 🙌