

## COMP7015 Artificial Intelligence

# Lecture 5: Machine Learning I — Linear Models and Decision Tree

Instructor: Dr. Kejing Yin

Department of Computer Science  
Hong Kong Baptist University

October 6, 2022

# Logistics

- Piazza for course Q&A:  
<https://piazza.com/hkbu.edu.hk/fall2022/comp7015/home>
- You are encouraged to post questions & feedback in Piazza. Usually you get quicker response in Piazza than in email.
- You can also create a post there to look for teammates for course project.
- Quiz on Oct. 20, covering:
  - Searching
  - Logics
  - Machine Learning Basics

## Toy example: how to build a system to detect cats?



List out all rules that specify the physical characteristics of a cat?

Not feasible:

- Not possible to enumerate all rules.
- Hard to distinguish with similar species.
- Cannot map image pixels to the rules.

# How we learned to recognize a cat?

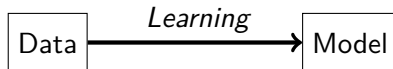
- We did not start from learning a set of rules that define a cat.
- We started by seeing cats and non-cats, in real life, in cartoon, etc.



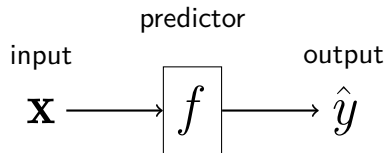
vs.



- **Machine Learning:** the same intuitive idea



# Abstraction of reflex-based models



- Machine learning learns a *model* (*predictor*) from a collection of *data*.
- With the *model*, we can perform inference to make *predictions*.
- We focus on reflex-based models, which make inference by a fixed set of fast and feedforward operations.
- a reflex-based model (*predictor*) takes some *input*  $\mathbf{x}$  and generates some *output*  $\hat{y}$ .

# Types of Machine Learning

- **Supervised Learning**

- Training with *labeled data*.
- Dataset:  $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ , where  $\mathbf{x}$  is the *feature* and  $y$  is the *label*.
- Example: training a cat detector with images of cats and other animals.

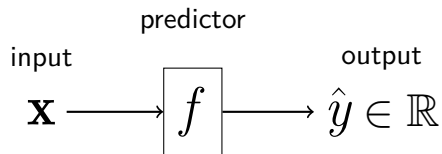
- **Unsupervised Learning**

- Training with *unlabeled data*.
- Dataset:  $\mathcal{D} = \{x_1, \dots, x_n\}$ , where  $\mathbf{x}$  is the feature.
- Example: given a set of images of animals, group images of the same species.

- **Semi-supervised Learning**

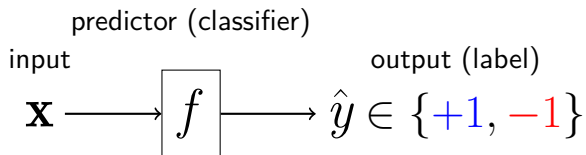
- A combination of supervised and unsupervised learning.
- Training with a small amount of *labeled data* and a large amount of *unlabeled data*.
- Dataset:  $\mathcal{D} = \{x_1, \dots, x_n\} \cup \{(x_1, y_1), \dots, (x_m, y_m)\}$ .

# Supervised Learning: Regression



- The output is a scalar.
- Example: information about house (location, area, etc.)  $\rightarrow$  Price

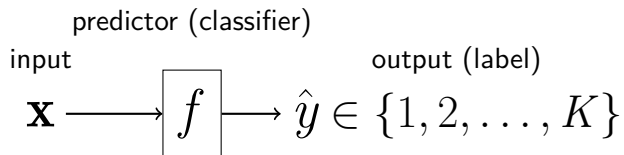
# Supervised Learning: Binary classification



- The output is a binary label of positive (+1) or negative (-1).
- Example:
  - Credit card application: information about applicants → Approve or Reject
  - 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  $K$  possibilities.
- Example:
  - Digit recognition: image of handwritten digit  $\rightarrow$  one of  $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
  - Animal classification: an image  $\rightarrow$  one of  $\{\text{Cat, Dog, Wolf, } \dots\}$
  - COVID-19 diagnosis: CT image  $\rightarrow$  **Has COVID-19** or **Does not have COVID-19**

## Application: house price prediction (1-dimensional)

- 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 neighbourhood:

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

$x_i$ : area of the  $i$ -th house;  $y_i$ : price of the  $i$ -th house;  $n$ : number of data samples

Fitting a linear regression:  $f(x_i) = \textcircled{w}x_i + \textcircled{b}$  such that  $f(x_i)$  is close to  $y_i$ .  
*coefficient/weight    bias*

# Loss function: how good is a predictor?

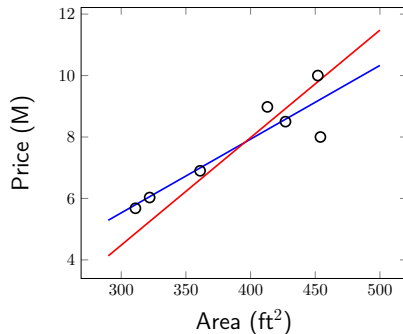
Fitting a linear regression:  $f(x_i) = wx_i + b$ , such that  $f(x_i)$  is close to  $y_i$ .

- Hypothesis class (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:  $f(x) = 0.035x - 6.02$

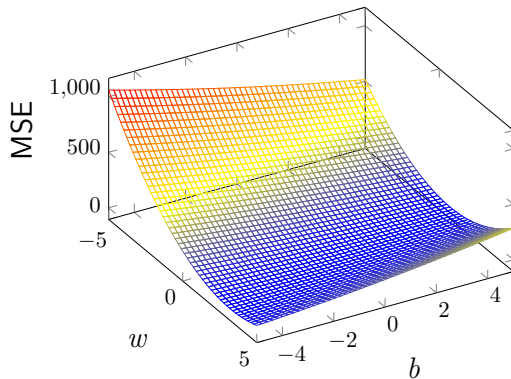
- Which one to choose?



The **square loss**:  $\ell(x_i, y_i, w, b) = \underbrace{(f_{w,b}(x_i) - y_i)^2}_{\text{residual}}$

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

# Loss function: a visualization of the 1-dimensional MSE



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

# Loss minimization framework: how to solve for the best $w$ and $b$

- 1-dimensional linear regression:  $f_{w,b}(x_i) = wx_i + b$ .
- Our goal: to make the mean square loss (MSE) as small as possible.
- **Convert it to the optimization problem:**

$$\begin{aligned}(w^*, b^*) &= \arg \min_{w,b} \text{MSE}(w, b) \\ &= \arg \min_{w,b} \frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}} (wx_i + b - y_i)^2\end{aligned}$$

- Algorithm to use: Gradient descent.

# Gradient descent algorithm

- **Graident:** the gradient  $\nabla_w f(w)$  is the direction of the greatest increase of  $f(w)$ .
- Start from an initial location, move a bit along the opposite direction of the gradient.

- **Graident descent algorithm**

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

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

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

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

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

“Rolling a ball down the hill”

# Gradient Computation

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

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

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

$$(By \text{ chain rule}) \quad = \frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}} 2(wx_i + b - y_i) \cdot \frac{\partial}{\partial w} (wx_i + b - y_i)$$

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

# 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 class:**  $\mathcal{F} = \{f_{w,b} : w \in \mathbb{R}, b \in \mathbb{R}\}$
- **Mean squared loss:**  $\text{MSE}(w, b) = \frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}} (f_{w,b}(x_i) - y_i)^2$
- **Gradient descent algorithm:**  $w \leftarrow w - \eta \nabla_w \text{MSE}(w, b)$



## 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*:  $\mathbf{x} \in \mathbb{R}^d$ .
- We say that the data samples have  $d$  *features*.
- The dataset:  $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$
- $d$ -dimensional linear regression:  $f_{\mathbf{w}, b}(x_i) = \mathbf{w}^\top \mathbf{x}_i + b$  or  $f_{\hat{\mathbf{w}}}(x_i) = \hat{\mathbf{w}}^\top \hat{\mathbf{x}}_i$ .

We can absorb the bias term into the weight vector:

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

## Linear regression for $d$ -dimensional data

- **Dataset:**  $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\};$

- **Linear regression:**  $f_{\mathbf{w}}(x_i) = \mathbf{w}^\top \mathbf{x}_i$

*(from now on, we directly use  $\mathbf{w} \in \mathbb{R}^{d+1}$  instead of  $\hat{\mathbf{w}}$  for simplicity);*

- **Hypothesis class:**  $\mathcal{F} = \{f_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^{d+1}\}$

- **Mean squared loss** (objective function):

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

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

## Linear regression for $d$ -dimensional data

- Writing it the matrix form:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top & 1 \\ \mathbf{x}_2^\top & 1 \\ \vdots & \vdots \\ \mathbf{x}_n^\top & 1 \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} & 1 \\ x_{21} & x_{22} & \cdots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nd} & 1 \end{bmatrix} \in \mathbb{R}^{n \times (d+1)}$$

$$\mathbf{y} = [y_1 \quad y_2 \quad \cdots \quad y_n]^\top \in \mathbb{R}^n$$

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

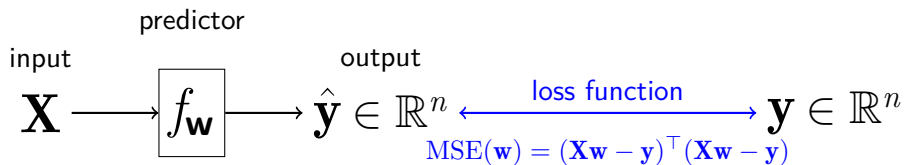
## Linear regression for $d$ -dimensional data

$$\text{MSE}(\mathbf{w}) = (\mathbf{X}\mathbf{w} - \mathbf{y})^\top (\mathbf{X}\mathbf{w} - \mathbf{y})$$

Gradient in matrix form:

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

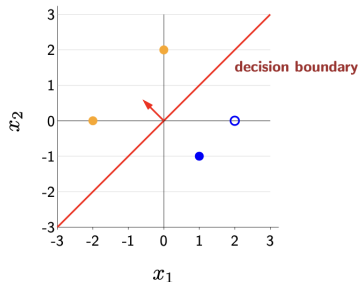
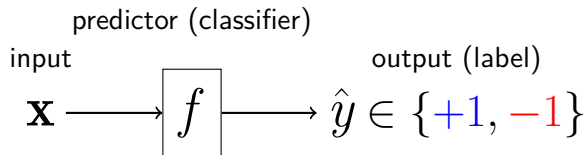
# Summary of Linear Regression



- Learning  $\mathbf{w}$  by minimizing the loss function.
- Minimizing the loss function using gradient descent.

# 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.

# Unit-step function

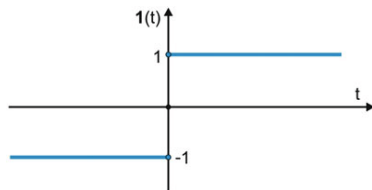
We need to transform the real number output  $\hat{y} \in \mathbb{R}$  to a binary value  $\hat{y} \in \{+1, -1\}$ .

**Unit-step function:**

$$f(\mathbf{x}) = \text{sign}(\underbrace{\mathbf{w}^\top \mathbf{x}}_{\text{Score}})$$

The score on an example  $(\mathbf{x}, y)$  is  $\mathbf{w}^\top \mathbf{x}$ : how confident we are in predicting  $+1$ .

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



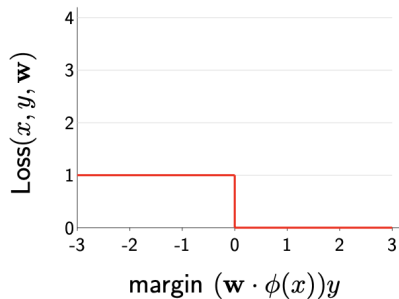
# Zero-one loss function

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

Target label:  $y$

We define the **zero-one loss** as:

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



The margin on an example  $(\mathbf{x}, y)$  is  $(\mathbf{w}^\top \mathbf{x})y$ : how correct we are.

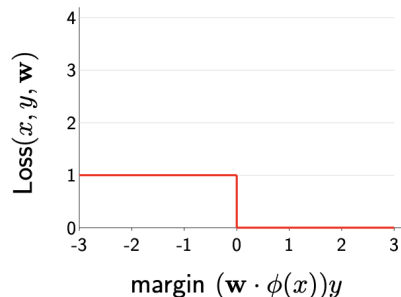


# Optimization: gradient descent with zero-one loss?

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

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

Almost zero everywhere!



What went wrong?

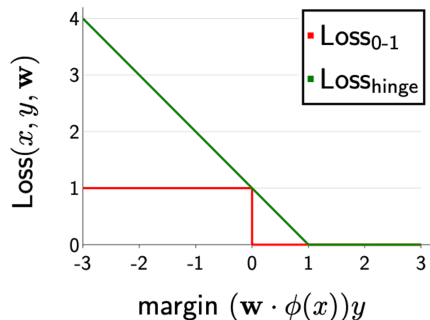
- In MSE, when we move  $\mathbf{w}$  a little bit, the residuals will change. The loss of some data points get smaller while some others get larger.
- In zero-one loss, moving  $\mathbf{w}$  a tiny bit changes very little the loss, until we make a big change that makes an example cross the decision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

# Hinge loss function

The margin on an example  $(\mathbf{x}, y)$  is  $(\mathbf{w}^\top \mathbf{x})y$ : how correct we are.

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

- In order to get zero loss, the model has to classify all points correctly and confidently (margin  $\geq 1$ ).
- If the model makes confident error, it incurs a linearly increasing penalty.
- Application: Support Vector Machine (SVM)

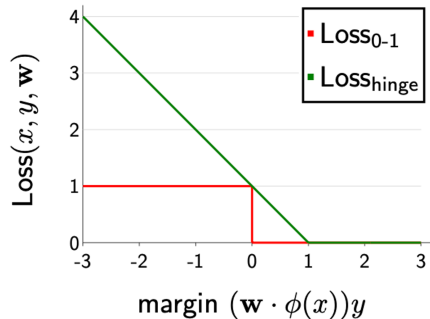


# Optimization: gradient descent with hinge loss

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

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

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



# Sigmoid function

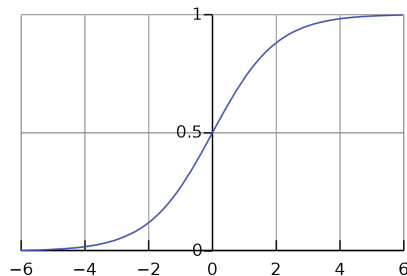
The sigmoid function (logistic function) maps a real value to be between  $(0, 1)$ :

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

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

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

This model is called **logistic regression**.

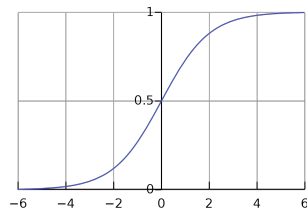


the sigmoid function

# Logistic regression

Consider this transformation 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 *likelihood* (probability of observing all data samples):

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

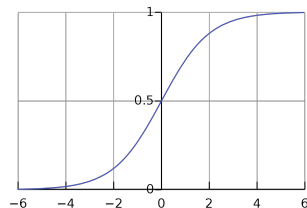
- The *log likelihood* (numerically more stable):

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

# Logistic regression

Consider this transformation 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):

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

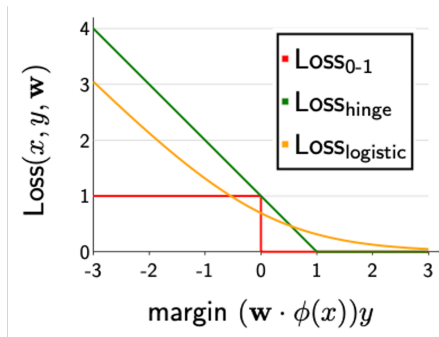
- We can maximize the likelihood, or equivalently, minimize the negative likelihood:

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

# Logistic regression

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

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



- Try to increase margin even when it already exceeds 1.
- Always have non-zero loss.
- Use gradient descent as in previous examples:  $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$

## Exercise: compute the gradient

- For the loss function of logistic regression:

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

- Compute its gradient  $\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$

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



# Summary of Part 1 (Linear Models)

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

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