



# Machine Learning - Classification

Amir H. Payberah  
payberah@kth.se  
14/11/2018

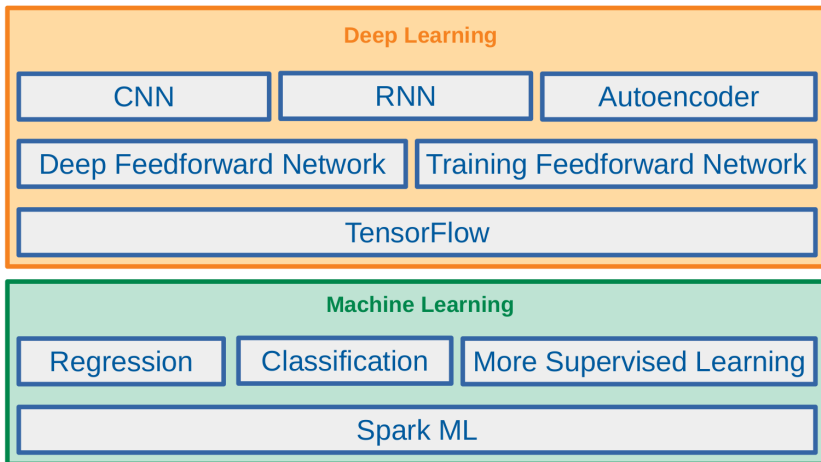




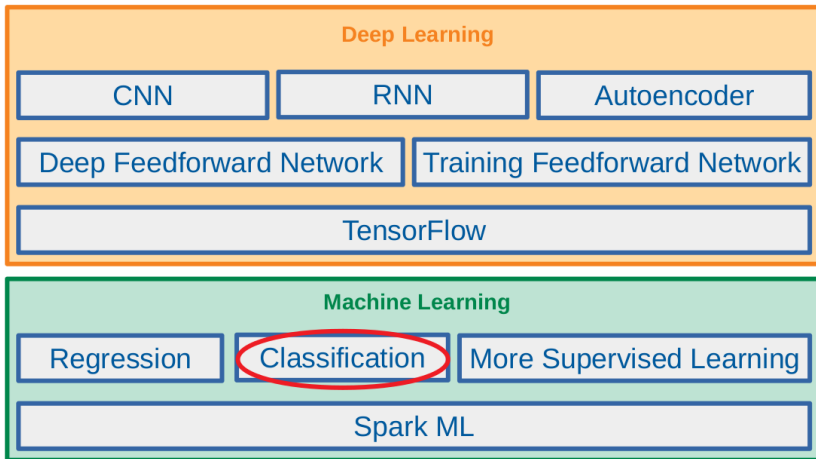
## The Course Web Page

<https://id2223kth.github.io>

# Where Are We?



## Where Are We?



# Let's Start with an Example

## Example (1/4)

- Given the dataset of  $m$  cancer tests.

| Tumor size | Cancer   |
|------------|----------|
| 330        | 1        |
| 120        | 0        |
| 400        | 1        |
| $\vdots$   | $\vdots$ |

## Example (1/4)

- ▶ Given the dataset of  $m$  cancer tests.

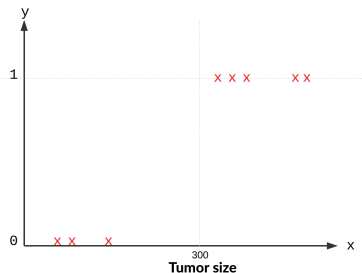
| Tumor size | Cancer   |
|------------|----------|
| 330        | 1        |
| 120        | 0        |
| 400        | 1        |
| $\vdots$   | $\vdots$ |

- ▶ Predict the risk of cancer, as a function of the tumor size?

## Example (2/4)

| Tumor size | Cancer   |
|------------|----------|
| 330        | 1        |
| 120        | 0        |
| 400        | 1        |
| $\vdots$   | $\vdots$ |

$$\mathbf{x} = \begin{bmatrix} 330 \\ 120 \\ 400 \\ \vdots \\ \vdots \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \\ \vdots \end{bmatrix}$$

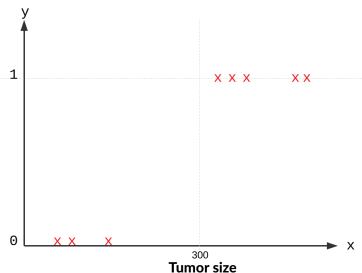




## Example (2/4)

| Tumor size | Cancer   |
|------------|----------|
| 330        | 1        |
| 120        | 0        |
| 400        | 1        |
| $\vdots$   | $\vdots$ |

$$\mathbf{x} = \begin{bmatrix} 330 \\ 120 \\ 400 \\ \vdots \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$

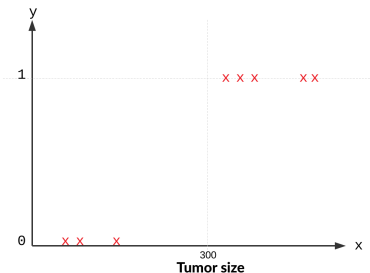


►  $\mathbf{x}^{(i)} \in \mathbb{R}$ :  $x_1^{(i)}$  is the **tumor size** of the **i**th instance in the **training set**.

## Example (3/4)

| Tumor size | Cancer   |
|------------|----------|
| 330        | 1        |
| 120        | 0        |
| 400        | 1        |
| $\vdots$   | $\vdots$ |

$$\mathbf{x} = \begin{bmatrix} 330 \\ 120 \\ 400 \\ \vdots \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$

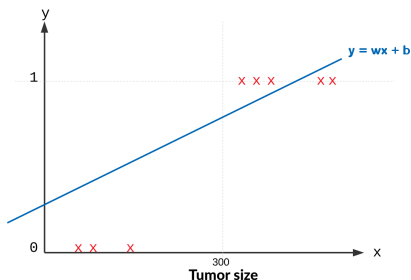


- Predict the risk of cancer  $\hat{y}$  as a function of the tumor sizes  $x_1$ , i.e.,  $\hat{y} = f(x_1)$
- E.g., what is  $\hat{y}$ , if  $x_1 = 500$ ?

## Example (3/4)

| Tumor size | Cancer   |
|------------|----------|
| 330        | 1        |
| 120        | 0        |
| 400        | 1        |
| $\vdots$   | $\vdots$ |

$$\mathbf{x} = \begin{bmatrix} 330 \\ 120 \\ 400 \\ \vdots \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$

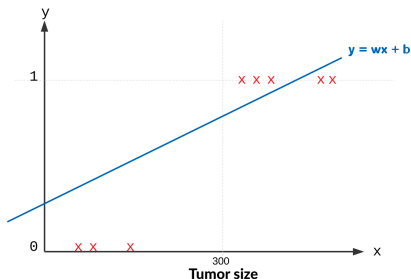


- Predict the risk of cancer  $\hat{y}$  as a function of the tumor sizes  $x_1$ , i.e.,  $\hat{y} = f(x_1)$
- E.g., what is  $\hat{y}$ , if  $x_1 = 500$ ?
- As an initial choice:  $\hat{y} = f_w(\mathbf{x}) = w_0 + w_1 x_1$

## Example (3/4)

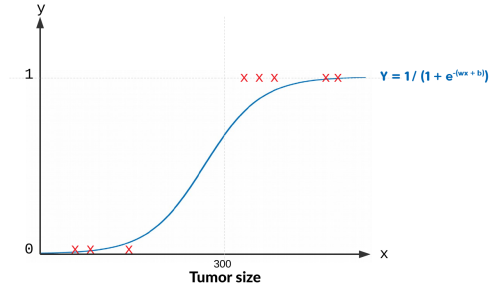
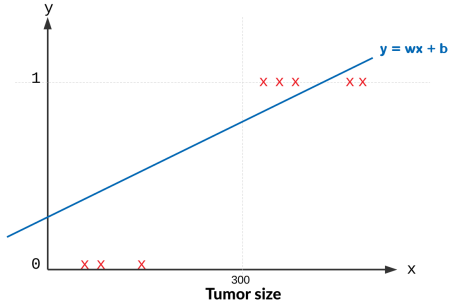
| Tumor size | Cancer   |
|------------|----------|
| 330        | 1        |
| 120        | 0        |
| 400        | 1        |
| $\vdots$   | $\vdots$ |

$$\mathbf{x} = \begin{bmatrix} 330 \\ 120 \\ 400 \\ \vdots \\ \vdots \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \\ \vdots \end{bmatrix}$$



- Predict the risk of cancer  $\hat{y}$  as a function of the tumor sizes  $\mathbf{x}_1$ , i.e.,  $\hat{y} = f(\mathbf{x}_1)$
- E.g., what is  $\hat{y}$ , if  $\mathbf{x}_1 = 500$ ?
- As an initial choice:  $\hat{y} = f_w(\mathbf{x}) = w_0 + w_1 x_1$
- Bad model!

## Example (4/4)

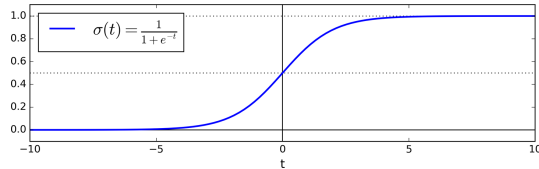


► A better model  $\hat{y} = \frac{1}{1 + e^{-(w_0 + w_1 x_1)}}$

# Sigmoid Function

- ▶ The **sigmoid function**, denoted by  $\sigma(\cdot)$ , outputs a number **between 0 and 1**.

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

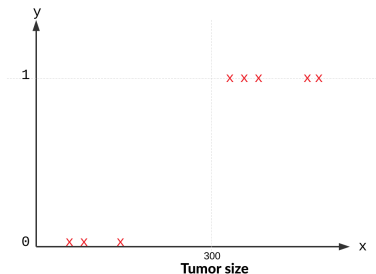


- ▶ When  $t < 0$ , then  $\sigma(t) < 0.5$
- ▶ when  $t \geq 0$ , then  $\sigma(t) \geq 0.5$

# Binomial Logistic Regression

# Binomial Logistic Regression (1/2)

- ▶ Our goal: to build a system that takes input  $\mathbf{x} \in \mathbb{R}^n$  and predicts output  $\hat{y} \in \{0, 1\}$ .
- ▶ To specify which of 2 categories an input  $\mathbf{x}$  belongs to.







## Binomial Logistic Regression (2/2)

- ▶ **Linear regression**: the model computes the **weighted sum of the input features** (plus a bias term).

$$\hat{y} = w_0x_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = \mathbf{w}^T \mathbf{x}$$

## Binomial Logistic Regression (2/2)

- ▶ **Linear regression**: the model computes the **weighted sum of the input features** (plus a bias term).

$$\hat{y} = w_0x_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = \mathbf{w}^T \mathbf{x}$$

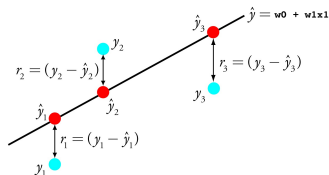
- ▶ **Binomial logistic regression**: the model computes a **weighted sum of the input features** (plus a bias term), but it **outputs the logistic of this result**.

$$z = w_0x_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = \mathbf{w}^T \mathbf{x}$$

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

# How to Learn Model Parameters $\mathbf{w}$ ?

# Linear Regression - Cost Function



- ▶ One reasonable model should make  $\hat{y}$  close to  $y$ , at least for the training dataset.
- ▶ Cost function  $J(\mathbf{w})$ : the mean squared error (MSE)

$$\text{cost}(\hat{y}^{(i)}, y^{(i)}) = (\hat{y}^{(i)} - y^{(i)})^2$$

$$J(\mathbf{w}) = \frac{1}{m} \sum_i^m \text{cost}(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{m} \sum_i^m (\hat{y}^{(i)} - y^{(i)})^2$$

# Binomial Logistic Regression - Cost Function (1/5)

- Naive idea: minimizing the Mean Square Error (MSE)

$$\begin{aligned}\text{cost}(\hat{y}^{(i)}, y^{(i)}) &= (\hat{y}^{(i)} - y^{(i)})^2 \\ J(\mathbf{w}) &= \frac{1}{m} \sum_i^m \text{cost}(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{m} \sum_i^m (\hat{y}^{(i)} - y^{(i)})^2\end{aligned}$$

# Binomial Logistic Regression - Cost Function (1/5)

- Naive idea: minimizing the Mean Square Error (MSE)

$$\text{cost}(\hat{y}^{(i)}, y^{(i)}) = (\hat{y}^{(i)} - y^{(i)})^2$$
$$J(\mathbf{w}) = \frac{1}{m} \sum_i^m \text{cost}(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{m} \sum_i^m (\hat{y}^{(i)} - y^{(i)})^2$$

$$J(\mathbf{w}) = \text{MSE}(\mathbf{w}) = \frac{1}{m} \sum_i^m \left( \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}^{(i)}}} - y^{(i)} \right)^2$$

# Binomial Logistic Regression - Cost Function (1/5)

- Naive idea: minimizing the Mean Square Error (MSE)

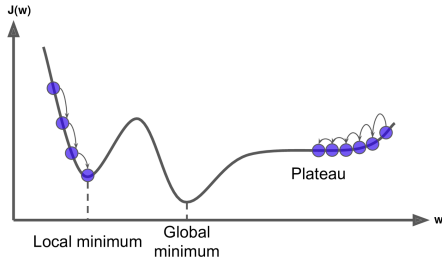
$$\text{cost}(\hat{y}^{(i)}, y^{(i)}) = (\hat{y}^{(i)} - y^{(i)})^2$$
$$J(\mathbf{w}) = \frac{1}{m} \sum_i^m \text{cost}(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{m} \sum_i^m (\hat{y}^{(i)} - y^{(i)})^2$$

$$J(\mathbf{w}) = \text{MSE}(\mathbf{w}) = \frac{1}{m} \sum_i^m \left( \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}^{(i)}}} - y^{(i)} \right)^2$$

- This cost function is a non-convex function for parameter optimization.

# Binomial Logistic Regression - Cost Function (2/5)

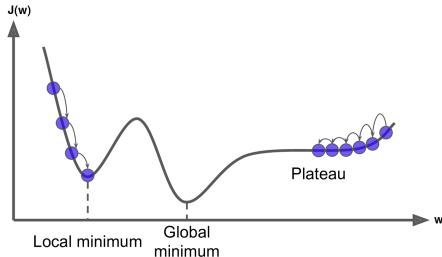
- ▶ What do we mean by **non-convex**?
- ▶ If a line joining two points on the curve, **crosses the curve**.
- ▶ The algorithm may converge to a **local minimum**.





# Binomial Logistic Regression - Cost Function (2/5)

- ▶ What do we mean by **non-convex**?
- ▶ If a line joining two points on the curve, **crosses the curve**.
- ▶ The algorithm may converge to a **local minimum**.
- ▶ We want a **convex** logistic regression **cost function**  $J(\mathbf{w})$ .





## Binomial Logistic Regression - Cost Function (3/5)

- ▶ The predicted value  $\hat{y} = \sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$
- ▶  $\text{cost}(\hat{y}^{(i)}, y^{(i)}) = ?$

## Binomial Logistic Regression - Cost Function (3/5)

- ▶ The predicted value  $\hat{y} = \sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$
- ▶  $\text{cost}(\hat{y}^{(i)}, y^{(i)}) = ?$
- ▶ The  $\text{cost}(\hat{y}^{(i)}, y^{(i)})$  should be
  - Close to 0, if the predicted value  $\hat{y}$  will be close to true value  $y$ .
  - Large, if the predicted value  $\hat{y}$  will be far from the true value  $y$ .

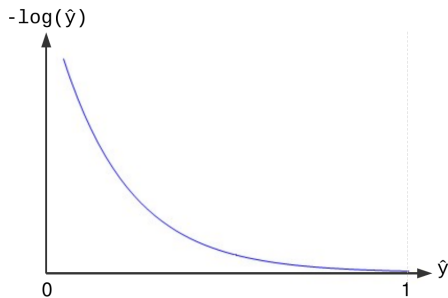
## Binomial Logistic Regression - Cost Function (3/5)

- ▶ The predicted value  $\hat{y} = \sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$
- ▶  $\text{cost}(\hat{y}^{(i)}, y^{(i)}) = ?$
- ▶ The  $\text{cost}(\hat{y}^{(i)}, y^{(i)})$  should be
  - Close to 0, if the predicted value  $\hat{y}$  will be close to true value  $y$ .
  - Large, if the predicted value  $\hat{y}$  will be far from the true value  $y$ .

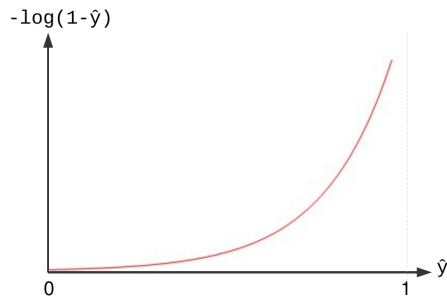
$$\text{cost}(\hat{y}^{(i)}, y^{(i)}) = \begin{cases} -\log(\hat{y}^{(i)}) & \text{if } y^{(i)} = 1 \\ -\log(1 - \hat{y}^{(i)}) & \text{if } y^{(i)} = 0 \end{cases}$$

# Binomial Logistic Regression - Cost Function (4/5)

$$\text{cost}(\hat{y}^{(i)}, y^{(i)}) = \begin{cases} -\log(\hat{y}^{(i)}) & \text{if } y^{(i)} = 1 \\ -\log(1 - \hat{y}^{(i)}) & \text{if } y^{(i)} = 0 \end{cases}$$



when  $y = 1$



when  $y = 0$

# Binomial Logistic Regression - Cost Function (5/5)

- We can define  $J(\mathbf{w})$  as below

$$\text{cost}(\hat{y}^{(i)}, y^{(i)}) = \begin{cases} -\log(\hat{y}^{(i)}) & \text{if } y^{(i)} = 1 \\ -\log(1 - \hat{y}^{(i)}) & \text{if } y^{(i)} = 0 \end{cases}$$

## Binomial Logistic Regression - Cost Function (5/5)

- We can define  $J(\mathbf{w})$  as below

$$\text{cost}(\hat{y}^{(i)}, y^{(i)}) = \begin{cases} -\log(\hat{y}^{(i)}) & \text{if } y^{(i)} = 1 \\ -\log(1 - \hat{y}^{(i)}) & \text{if } y^{(i)} = 0 \end{cases}$$

$$J(\mathbf{w}) = \frac{1}{m} \sum_i^m \text{cost}(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{m} \sum_i^m (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$$



# How to Learn Model Parameters $\mathbf{w}$ ?

- ▶ We want to choose  $\mathbf{w}$  so as to minimize  $J(\mathbf{w})$ .
- ▶ An approach to find  $\mathbf{w}$ : gradient descent
  - Batch gradient descent
  - Stochastic gradient
  - Mini-batch gradient descent





# Binomial Logistic Regression Gradient Descent (1/3)

► Goal: find  $\mathbf{w}$  that minimizes  $J(\mathbf{w}) = \sum_i^m (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$ .



# Binomial Logistic Regression Gradient Descent (1/3)

- ▶ **Goal:** find  $\mathbf{w}$  that **minimizes**  $J(\mathbf{w}) = \sum_i^m (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$ .
- ▶ Start at a **random point**, and repeat the following **steps**, until the **stopping criterion** is satisfied:

# Binomial Logistic Regression Gradient Descent (1/3)

- ▶ **Goal:** find  $\mathbf{w}$  that **minimizes**  $J(\mathbf{w}) = \sum_i^m (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$ .
- ▶ Start at a **random point**, and repeat the following **steps**, until the **stopping criterion** is satisfied:
  1. Determine a **descent direction**  $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$

# Binomial Logistic Regression Gradient Descent (1/3)

- ▶ **Goal:** find  $\mathbf{w}$  that **minimizes**  $J(\mathbf{w}) = \sum_i^m (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$ .
- ▶ Start at a **random point**, and repeat the following **steps**, until the **stopping criterion** is satisfied:
  1. Determine a **descent direction**  $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$
  2. Choose a **step size**  $\eta$

# Binomial Logistic Regression Gradient Descent (1/3)

- ▶ **Goal:** find  $\mathbf{w}$  that **minimizes**  $J(\mathbf{w}) = \sum_i^m (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$ .
- ▶ Start at a **random point**, and repeat the following **steps**, until the **stopping criterion** is satisfied:
  1. Determine a **descent direction**  $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$
  2. Choose a **step size**  $\eta$
  3. **Update** the parameters:  $\mathbf{w}^{(\text{next})} = \mathbf{w} - \eta \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$  (**simultaneously** for all parameters)

## Binomial Logistic Regression Gradient Descent (2/3)

- 1. Determine a **descent direction**  $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$ .

$$\hat{y} = \sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

$$J(\mathbf{w}) = \frac{1}{m} \sum_i^m \text{cost}(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{m} \sum_i^m (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$$

# Binomial Logistic Regression Gradient Descent (2/3)

- 1. Determine a **descent direction**  $\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$ .

$$\hat{y} = \sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

$$J(\mathbf{w}) = \frac{1}{m} \sum_i \text{cost}(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{m} \sum_i (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$$

$$\begin{aligned} \frac{\partial J(\mathbf{w})}{\partial w_j} &= \frac{1}{m} \sum_i \left( y^{(i)} \frac{1}{\hat{y}^{(i)}} - (1 - y^{(i)}) \frac{1}{1 - \hat{y}^{(i)}} \right) \frac{\partial \hat{y}^{(i)}}{\partial w_j} \\ &= \frac{1}{m} \sum_i \left( y^{(i)} \frac{1}{\hat{y}^{(i)}} - (1 - y^{(i)}) \frac{1}{1 - \hat{y}^{(i)}} \right) \hat{y}^{(i)} (1 - \hat{y}^{(i)}) \frac{\partial \mathbf{w}^T \mathbf{x}}{\partial w_j} \\ &= \frac{1}{m} \sum_i \left( y^{(i)} (1 - \hat{y}^{(i)}) - (1 - y^{(i)}) \hat{y}^{(i)} \right) x_j \\ &= \frac{1}{m} \sum_i (\hat{y}^{(i)} - y^{(i)}) x_j \end{aligned}$$

# Binomial Logistic Regression Gradient Descent (3/3)

- ▶ 2. Choose a step size  $\eta$
- ▶ 3. Update the parameters:  $w_j^{(\text{next})} = w_j - \eta \frac{\partial J(\mathbf{w})}{\partial w_j}$ 
  - $0 \leq j \leq n$ , where  $n$  is the number of features.



# Binomial Logistic Regression Gradient Descent - Example (1/4)

| Tumor size | Cancer |
|------------|--------|
| 330        | 1      |
| 120        | 0      |
| 400        | 1      |

$$\mathbf{X} = \left[ \begin{array}{c|c} 1 & 330 \\ 1 & 120 \\ 1 & 400 \end{array} \right] \quad \mathbf{y} = \left[ \begin{array}{c} 1 \\ 0 \\ 1 \end{array} \right]$$

- Predict the risk of cancer  $\hat{y}$  as a function of the tumor sizes  $x_1$ .
- E.g., what is  $\hat{y}$ , if  $x_1 = 500$ ?

## Binomial Logistic Regression Gradient Descent - Example (2/4)

$$\mathbf{X} = \left[ \begin{array}{c|c} 1 & 330 \\ 1 & 120 \\ 1 & 400 \end{array} \right] \quad \mathbf{y} = \left[ \begin{array}{c} 1 \\ 0 \\ 1 \end{array} \right]$$

$$\hat{y} = \sigma(\mathbf{w}_0 + \mathbf{w}_1 \mathbf{x}_1) = \frac{1}{1 + e^{-(\mathbf{w}_0 + \mathbf{w}_1 \mathbf{x}_1)}}$$

$$J(\mathbf{w}) = \frac{1}{m} \sum_i^m (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$$

## Binomial Logistic Regression Gradient Descent - Example (2/4)

$$\mathbf{X} = \left[ \begin{array}{c|c} 1 & 330 \\ 1 & 120 \\ 1 & 400 \end{array} \right] \quad \mathbf{y} = \left[ \begin{array}{c} 1 \\ 0 \\ 1 \end{array} \right]$$

$$\hat{y} = \sigma(\mathbf{w}_0 + \mathbf{w}_1 \mathbf{x}_1) = \frac{1}{1 + e^{-(\mathbf{w}_0 + \mathbf{w}_1 \mathbf{x}_1)}}$$

$$J(\mathbf{w}) = \frac{1}{m} \sum_i^m (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$$

$$\begin{aligned} \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_0} &= \frac{1}{3} \sum_i^3 (\hat{y}^{(i)} - y^{(i)}) \mathbf{x}_0 \\ &= \frac{1}{3} \left[ \left( \frac{1}{1 + e^{-(\mathbf{w}_0 + 330\mathbf{w}_1)}} - 1 \right) + \left( \frac{1}{1 + e^{-(\mathbf{w}_0 + 120\mathbf{w}_1)}} - 0 \right) + \left( \frac{1}{1 + e^{-(\mathbf{w}_0 + 400\mathbf{w}_1)}} - 1 \right) \right] \end{aligned}$$

## Binomial Logistic Regression Gradient Descent - Example (3/4)

$$\mathbf{X} = \left[ \begin{array}{c|c} 1 & 330 \\ 1 & 120 \\ 1 & 400 \end{array} \right] \quad \mathbf{y} = \left[ \begin{array}{c} 1 \\ 0 \\ 1 \end{array} \right]$$

$$\hat{y} = \sigma(w_0 + w_1 x_1) = \frac{1}{1 + e^{-(w_0 + w_1 x_1)}}$$

$$J(\mathbf{w}) = \frac{1}{m} \sum_i^m (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$$

## Binomial Logistic Regression Gradient Descent - Example (3/4)

$$\mathbf{X} = \left[ \begin{array}{c|c} 1 & 330 \\ 1 & 120 \\ 1 & 400 \end{array} \right] \quad \mathbf{y} = \left[ \begin{array}{c} 1 \\ 0 \\ 1 \end{array} \right]$$

$$\hat{y} = \sigma(\mathbf{w}_0 + \mathbf{w}_1 \mathbf{x}_1) = \frac{1}{1 + e^{-(\mathbf{w}_0 + \mathbf{w}_1 \mathbf{x}_1)}}$$

$$J(\mathbf{w}) = \frac{1}{m} \sum_i (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$$

$$\begin{aligned} \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_1} &= \frac{1}{3} \sum_i (\hat{y}^{(i)} - y^{(i)}) \mathbf{x}_1 \\ &= \frac{1}{3} \left[ 330 \left( \frac{1}{1 + e^{-(\mathbf{w}_0 + 330\mathbf{w}_1)}} - 1 \right) + 120 \left( \frac{1}{1 + e^{-(\mathbf{w}_0 + 120\mathbf{w}_1)}} - 0 \right) + 400 \left( \frac{1}{1 + e^{-(\mathbf{w}_0 + 400\mathbf{w}_1)}} - 1 \right) \right] \end{aligned}$$

## Binomial Logistic Regression Gradient Descent - Example (4/4)

$$\begin{aligned}w_0^{(\text{next})} &= w_0 - \eta \frac{\partial J(\mathbf{w})}{\partial w_0} \\w_1^{(\text{next})} &= w_1 - \eta \frac{\partial J(\mathbf{w})}{\partial w_1}\end{aligned}$$



# Binomial Logistic Regression in Spark

```
case class cancer(x1: Long, y: Long)

val trainData = Seq(cancer(330, 1), cancer(120, 0), cancer(400, 1)).toDF
val testData = Seq(cancer(500, 0)).toDF
```



# Binomial Logistic Regression in Spark

```
case class cancer(x1: Long, y: Long)

val trainData = Seq(cancer(330, 1), cancer(120, 0), cancer(400, 1)).toDF
val testData = Seq(cancer(500, 0)).toDF

import org.apache.spark.ml.feature.VectorAssembler

val va = new VectorAssembler().setInputCols(Array("x1")).setOutputCol("features")

val train = va.transform(trainData)
val test = va.transform(testData)
```



# Binomial Logistic Regression in Spark

```
case class cancer(x1: Long, y: Long)

val trainData = Seq(cancer(330, 1), cancer(120, 0), cancer(400, 1)).toDF
val testData = Seq(cancer(500, 0)).toDF
```

```
import org.apache.spark.ml.feature.VectorAssembler

val va = new VectorAssembler().setInputCols(Array("x1")).setOutputCol("features")

val train = va.transform(trainData)
val test = va.transform(testData)
```

```
import org.apache.spark.ml.classification.LogisticRegression

val lr = new LogisticRegression().setFeaturesCol("features").setLabelCol("y")
    .setMaxIter(10).setRegParam(0.3).setElasticNetParam(0.8)

val lrModel = lr.fit(train)
lrModel.transform(test).show
```

# Binomial Logistic Regression

## Probabilistic Interpretation



## Probability and Likelihood (1/2)

- Let  $X : \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)}\}$  be a discrete random variable drawn independently from a distribution probability  $p$  depending on a parameter  $\theta$ .



## Probability and Likelihood (1/2)

- Let  $X : \{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$  be a discrete random variable drawn independently from a distribution probability  $p$  depending on a parameter  $\theta$ .
- For six tosses of a coin,  $X : \{h, t, t, t, h, t\}$ ,  $h$ : head, and  $t$ : tail.
  - Suppose you have a coin with probability  $\theta$  to land heads and  $(1 - \theta)$  to land tails.

## Probability and Likelihood (1/2)

- ▶ Let  $X : \{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$  be a discrete random variable drawn independently from a distribution probability  $p$  depending on a parameter  $\theta$ .
  - For six tosses of a coin,  $X : \{h, t, t, t, h, t\}$ ,  $h$ : head, and  $t$ : tail.
  - Suppose you have a coin with probability  $\theta$  to land heads and  $(1 - \theta)$  to land tails.
- ▶  $p(X \mid \theta = \frac{2}{3})$  is the probability of  $X$  given  $\theta = \frac{2}{3}$ .

## Probability and Likelihood (1/2)

- ▶ Let  $X : \{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$  be a **discrete random variable** drawn **independently** from a **distribution probability  $p$**  depending on a **parameter  $\theta$** .
  - For six tosses of a coin,  $X : \{h, t, t, t, h, t\}$ , **h**: head, and **t**: tail.
  - Suppose you have a **coin** with probability  $\theta$  to land heads and  $(1 - \theta)$  to land tails.
- ▶  $p(X \mid \theta = \frac{2}{3})$  is the **probability** of **X** given  $\theta = \frac{2}{3}$ .
- ▶  $p(X = h \mid \theta)$  is the **likelihood** of  $\theta$  given  $X = h$ .

## Probability and Likelihood (1/2)

- ▶ Let  $X : \{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$  be a **discrete random variable** drawn **independently** from a **distribution probability**  $p$  depending on a **parameter**  $\theta$ .
  - For six tosses of a coin,  $X : \{h, t, t, t, h, t\}$ ,  $h$ : head, and  $t$ : tail.
  - Suppose you have a **coin** with probability  $\theta$  to land heads and  $(1 - \theta)$  to land tails.
- ▶  $p(X \mid \theta = \frac{2}{3})$  is the **probability** of  $X$  given  $\theta = \frac{2}{3}$ .
- ▶  $p(X = h \mid \theta)$  is the **likelihood** of  $\theta$  given  $X = h$ .
- ▶ **Likelihood (L)**: a function of the **parameters** ( $\theta$ ) of a probability model, given **specific observed data**, e.g.,  $X = h$ .

$$L(\theta) = p(X \mid \theta)$$

## Probability and Likelihood (2/2)

- If samples in  $\mathbf{X}$  are **independent** we have:

$$\begin{aligned} L(\theta) &= p(\mathbf{X} \mid \theta) = p(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)} \mid \theta) \\ &= p(\mathbf{x}^{(1)} \mid \theta) p(\mathbf{x}^{(2)} \mid \theta) \cdots p(\mathbf{x}^{(m)} \mid \theta) = \prod_{i=1}^m p(\mathbf{x}^{(i)} \mid \theta) \end{aligned}$$





# Likelihood and Log-Likelihood

- ▶ The Likelihood product is prone to numerical underflow.

$$L(\theta) = p(X | \theta) = \prod_{i=1}^m p(x^{(i)} | \theta)$$

# Likelihood and Log-Likelihood

- ▶ The Likelihood product is prone to numerical underflow.

$$L(\theta) = p(X | \theta) = \prod_{i=1}^m p(x^{(i)} | \theta)$$

- ▶ To overcome this problem we can use the logarithm of the likelihood.
  - Transforms a product into a sum.

$$\log(L(\theta)) = \log(p(X | \theta)) = \sum_{i=1}^m \log p(x^{(i)} | \theta)$$

# Likelihood and Log-Likelihood

- ▶ The Likelihood product is prone to numerical underflow.

$$L(\theta) = p(X | \theta) = \prod_{i=1}^m p(x^{(i)} | \theta)$$

- ▶ To overcome this problem we can use the logarithm of the likelihood.
  - Transforms a product into a sum.

$$\log(L(\theta)) = \log(p(X | \theta)) = \sum_{i=1}^m \log p(x^{(i)} | \theta)$$

- ▶ Negative Log-Likelihood:  $-\log L(\theta) = -\sum_{i=1}^m \log p(x^{(i)} | \theta)$

## Binomial Logistic Regression and Log-Likelihood (1/2)

- Let's consider the value of  $\hat{y}^{(i)}$  as the **probability**:

$$\begin{cases} p(y^{(i)} = 1 \mid \mathbf{x}^{(i)}; \mathbf{w}) = \hat{y}^{(i)} \\ p(y^{(i)} = 0 \mid \mathbf{x}^{(i)}; \mathbf{w}) = 1 - \hat{y}^{(i)} \end{cases} \Rightarrow p(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}) = (\hat{y}^{(i)})^{y^{(i)}} (1 - \hat{y}^{(i)})^{(1-y^{(i)})}$$

## Binomial Logistic Regression and Log-Likelihood (1/2)

- ▶ Let's consider the value of  $\hat{y}^{(i)}$  as the **probability**:

$$\begin{cases} p(y^{(i)} = 1 \mid \mathbf{x}^{(i)}; \mathbf{w}) = \hat{y}^{(i)} \\ p(y^{(i)} = 0 \mid \mathbf{x}^{(i)}; \mathbf{w}) = 1 - \hat{y}^{(i)} \end{cases} \Rightarrow p(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}) = (\hat{y}^{(i)})^{y^{(i)}} (1 - \hat{y}^{(i)})^{(1-y^{(i)})}$$

- ▶ So the **likelihood** is:

$$L(\mathbf{w}) = p(y \mid \mathbf{x}; \mathbf{w}) = \prod_{i=1}^m p(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}) = \prod_{i=1}^m (\hat{y}^{(i)})^{y^{(i)}} (1 - \hat{y}^{(i)})^{(1-y^{(i)})}$$

# Binomial Logistic Regression and Log-Likelihood (1/2)

- ▶ Let's consider the value of  $\hat{y}^{(i)}$  as the **probability**:

$$\begin{cases} p(y^{(i)} = 1 \mid \mathbf{x}^{(i)}; \mathbf{w}) = \hat{y}^{(i)} \\ p(y^{(i)} = 0 \mid \mathbf{x}^{(i)}; \mathbf{w}) = 1 - \hat{y}^{(i)} \end{cases} \Rightarrow p(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}) = (\hat{y}^{(i)})^{y^{(i)}} (1 - \hat{y}^{(i)})^{(1-y^{(i)})}$$

- ▶ So the **likelihood** is:

$$L(\mathbf{w}) = p(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) = \prod_{i=1}^m p(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}) = \prod_{i=1}^m (\hat{y}^{(i)})^{y^{(i)}} (1 - \hat{y}^{(i)})^{(1-y^{(i)})}$$

- ▶ And the **negative log-likelihood**:

$$-\log(L(\mathbf{w})) = -\sum_{i=1}^m \log p(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}) = -\sum_{i=1}^m y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

## Binomial Logistic Regression and Log-Likelihood (2/2)

- The **negative log-likelihood**:

$$-\log(L(\mathbf{w})) = -\sum_{i=1}^m \log p(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}) = -\sum_{i=1}^m y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

## Binomial Logistic Regression and Log-Likelihood (2/2)

- ▶ The **negative log-likelihood**:

$$-\log(L(\mathbf{w})) = -\sum_{i=1}^m \log p(y^{(i)} | \mathbf{x}^{(i)}; \mathbf{w}) = -\sum_{i=1}^m y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

- ▶ This equation is the same as the **logistic regression cost function**.

$$J(\mathbf{w}) = \frac{1}{m} \sum_i^m (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$$



## Binomial Logistic Regression and Log-Likelihood (2/2)

- ▶ The **negative log-likelihood**:

$$-\log(L(\mathbf{w})) = -\sum_{i=1}^m \log p(y^{(i)} | \mathbf{x}^{(i)}; \mathbf{w}) = -\sum_{i=1}^m y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

- ▶ This equation is the same as the the **logistic regression cost function**.

$$J(\mathbf{w}) = \frac{1}{m} \sum_i^m (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$$

- ▶ **Minimize the negative log-likelihood** to **minimize the cost function**  $J(\mathbf{w})$ .

# Binomial Logistic Regression and Cross-Entropy (1/2)

- ▶ Negative log-likelihood is also called the **cross-entropy**
- ▶ **Cross-entropy**: quantify the **difference (error)** between **two probability distributions**.
- ▶ **How close** is the **predicted distribution** to the **true distribution**?

$$H(p, q) = - \sum_j p_j \log(q_j)$$

- ▶ Where **p** is the **true distribution**, and **q** is the **predicted distribution**.



## Binomial Logistic Regression and Cross-Entropy (2/2)

$$H(p, q) = - \sum_j p_j \log(q_j)$$

## Binomial Logistic Regression and Cross-Entropy (2/2)

$$H(p, q) = - \sum_j p_j \log(q_j)$$

- The **true probability** distribution:  $p(y = 1) = y$  and  $p(y = 0) = 1 - y$

## Binomial Logistic Regression and Cross-Entropy (2/2)

$$H(p, q) = - \sum_j p_j \log(q_j)$$

- ▶ The **true probability** distribution:  $p(y = 1) = y$  and  $p(y = 0) = 1 - y$
- ▶ The **predicted probability** distribution:  $q(y = 1) = \hat{y}$  and  $q(y = 0) = 1 - \hat{y}$

## Binomial Logistic Regression and Cross-Entropy (2/2)

$$H(p, q) = - \sum_j p_j \log(q_j)$$

- ▶ The **true probability** distribution:  $p(y = 1) = y$  and  $p(y = 0) = 1 - y$
- ▶ The **predicted probability** distribution:  $q(y = 1) = \hat{y}$  and  $q(y = 0) = 1 - \hat{y}$
- ▶  $p \in \{y, 1 - y\}$  and  $q \in \{\hat{y}, 1 - \hat{y}\}$

## Binomial Logistic Regression and Cross-Entropy (2/2)

$$H(p, q) = - \sum_j p_j \log(q_j)$$

- ▶ The **true probability** distribution:  $p(y = 1) = y$  and  $p(y = 0) = 1 - y$
- ▶ The **predicted probability** distribution:  $q(y = 1) = \hat{y}$  and  $q(y = 0) = 1 - \hat{y}$
- ▶  $p \in \{y, 1 - y\}$  and  $q \in \{\hat{y}, 1 - \hat{y}\}$
- ▶ So, the **cross-entropy** of  $p$  and  $q$  is nothing but the **logistic cost function**.

$$H(p, q) = - \sum_j p_j \log(q_j) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) = \text{cost}(y, \hat{y})$$



$$H(p, q) = - \sum_j p_j \log(q_j)$$

- ▶ The **true probability** distribution:  $p(y = 1) = y$  and  $p(y = 0) = 1 - y$
- ▶ The **predicted probability** distribution:  $q(y = 1) = \hat{y}$  and  $q(y = 0) = 1 - \hat{y}$
- ▶  $p \in \{y, 1 - y\}$  and  $q \in \{\hat{y}, 1 - \hat{y}\}$
- ▶ So, the **cross-entropy** of  $p$  and  $q$  is nothing but the **logistic cost function**.

$$H(p, q) = - \sum_j p_j \log(q_j) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) = \text{cost}(y, \hat{y})$$

$$J(\mathbf{w}) = \frac{1}{m} \sum_i^m \text{cost}(y, \hat{y}) = \frac{1}{m} \sum_i^m H(p, q) = -\frac{1}{m} \sum_i^m (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$$

- Minimize the cross-entropy to minimize the cost function  $J(\mathbf{w})$ .



# Multinomial Logistic Regression



# Multinomial Logistic Regression

- ▶ **Multinomial classifiers** can distinguish between **more than two classes**.
- ▶ Instead of  $y \in \{0, 1\}$ , we have  $y \in \{0, 1, \dots, k\}$ .



## Binomial vs. Multinomial Logistic Regression (1/2)

- ▶ In a **binary class**  $y \in \{0, 1\}$ , the **estimator** is  $\hat{y} = p(y = 1 \mid \mathbf{x}; \mathbf{w})$ .
  - We find **one** set of parameters  $\mathbf{w}$ .

$$\mathbf{w}^T = [w_0, w_1, \dots, w_n]$$

## Binomial vs. Multinomial Logistic Regression (1/2)

- ▶ In a **binary class**  $y \in \{0, 1\}$ , the **estimator** is  $\hat{y} = p(y = 1 \mid \mathbf{x}; \mathbf{w})$ .
  - We find **one** set of parameters  $\mathbf{w}$ .

$$\mathbf{w}^T = [w_0, w_1, \dots, w_n]$$

- ▶ In **multiclass**  $y \in \{1, 2, \dots, k\}$ , we need to estimate the result for each **individual label**, i.e.,  $\hat{y}_j = p(y = j \mid \mathbf{x}; \mathbf{w})$ .

## Binomial vs. Multinomial Logistic Regression (1/2)

- ▶ In a **binary class**  $y \in \{0, 1\}$ , the **estimator** is  $\hat{y} = p(y = 1 \mid \mathbf{x}; \mathbf{w})$ .
  - We find **one** set of parameters  $\mathbf{w}$ .

$$\mathbf{w}^T = [w_0, w_1, \dots, w_n]$$

- ▶ In **multiclass**  $y \in \{1, 2, \dots, k\}$ , we need to estimate the result for each **individual label**, i.e.,  $\hat{y}_j = p(y = j \mid \mathbf{x}; \mathbf{w})$ .
  - We find **k** set of parameters  $\mathbf{W}$ .

$$\mathbf{W} = \begin{bmatrix} [w_{0,1}, w_{1,1}, \dots, w_{n,1}] \\ [w_{0,2}, w_{1,2}, \dots, w_{n,2}] \\ \vdots \\ [w_{0,k}, w_{1,k}, \dots, w_{n,k}] \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \vdots \\ \mathbf{w}_k^T \end{bmatrix}$$

## Binomial vs. Multinomial Logistic Regression (2/2)

- In a **binary class**,  $y \in \{0, 1\}$ , we use the **sigmoid** function.

$$\mathbf{w}^T \mathbf{x} = w_0 x_0 + w_1 x_1 + \cdots + w_n x_n$$

$$\hat{y} = p(y = 1 \mid \mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

## Binomial vs. Multinomial Logistic Regression (2/2)

- In a **binary class**,  $y \in \{0, 1\}$ , we use the **sigmoid** function.

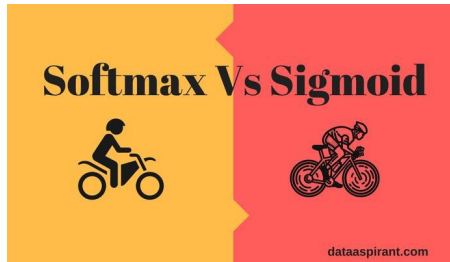
$$\mathbf{w}^T \mathbf{x} = w_0 x_0 + w_1 x_1 + \cdots + w_n x_n$$
$$\hat{y} = p(y = 1 \mid \mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

- In **multiclass**,  $y \in \{1, 2, \dots, k\}$ , we use the **softmax** function.

$$\mathbf{w}_j^T \mathbf{x} = w_{0,j} x_0 + w_{1,j} x_1 + \cdots + w_{n,j} x_n, 1 \leq j \leq k$$
$$\hat{y}_j = p(y = j \mid \mathbf{x}; \mathbf{w}_j) = \sigma(\mathbf{w}_j^T \mathbf{x}) = \frac{e^{\mathbf{w}_j^T \mathbf{x}}}{\sum_{i=1}^k e^{\mathbf{w}_i^T \mathbf{x}}}$$

# Sigmoid vs. Softmax

- ▶ **Sigmoid** function:  $\sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$
- ▶ **Softmax** function:  $\sigma(\mathbf{w}_j^T \mathbf{x}) = \frac{e^{\mathbf{w}_j^T \mathbf{x}}}{\sum_{i=1}^k e^{\mathbf{w}_i^T \mathbf{x}}}$ 
  - Calculate the probabilities of each target class over all possible target classes.
  - The softmax function for two classes is equivalent to the sigmoid function.





# How Does Softmax Work? - Step 1

- For each instance  $\mathbf{x}^{(i)}$ , computes the **score**  $\mathbf{w}_j^T \mathbf{x}^{(i)}$  for each **class**  $j$ .

$$\mathbf{w}_j^T \mathbf{x}^{(i)} = w_{0,j} x_0^{(i)} + w_{1,j} x_1^{(i)} + \dots + w_{n,j} x_n^{(i)}$$

- Note that each class  $j$  has its **own dedicated parameter** vector  $\mathbf{w}_j$ .

$$\mathbf{W} = \begin{bmatrix} [w_{0,1}, w_{1,1}, \dots, w_{n,1}] \\ [w_{0,2}, w_{1,2}, \dots, w_{n,2}] \\ \vdots \\ [w_{0,k}, w_{1,k}, \dots, w_{n,k}] \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \vdots \\ \mathbf{w}_k^T \end{bmatrix}$$

- $$\hat{y}_j^{(i)} = p(y^{(i)} = j \mid \mathbf{x}^{(i)}; \mathbf{w}_j) = \sigma(\mathbf{w}_j^T \mathbf{x}^{(i)}) = \frac{e^{\mathbf{w}_j^T \mathbf{x}^{(i)}}}{\sum_{l=1}^k e^{\mathbf{w}_l^T \mathbf{x}^{(i)}}}$$



## How Does Softmax Work? - Step 3

- Predicts the class with the **highest estimated probability**.

## Softmax Model Estimation and Prediction - Example (1/2)

- Assume we have a **training set** consisting of  $m = 4$  instances from  $k = 3$  **classes**.

$$\mathbf{x}^{(1)} \rightarrow \text{class1}, \mathbf{y}^{(1)\top} = [1 \ 0 \ 0]$$

$$\mathbf{x}^{(2)} \rightarrow \text{class2}, \mathbf{y}^{(2)\top} = [0 \ 1 \ 0]$$

$$\mathbf{x}^{(3)} \rightarrow \text{class3}, \mathbf{y}^{(3)\top} = [0 \ 0 \ 1]$$

$$\mathbf{x}^{(4)} \rightarrow \text{class3}, \mathbf{y}^{(4)\top} = [0 \ 0 \ 1]$$

$$\mathbf{Y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

# Softmax Model Estimation and Prediction - Example (1/2)

- Assume we have a **training set** consisting of  $m = 4$  instances from  $k = 3$  **classes**.

$$\mathbf{x}^{(1)} \rightarrow \text{class1}, \mathbf{y}^{(1)\top} = [1 \ 0 \ 0]$$

$$\mathbf{x}^{(2)} \rightarrow \text{class2}, \mathbf{y}^{(2)\top} = [0 \ 1 \ 0]$$

$$\mathbf{x}^{(3)} \rightarrow \text{class3}, \mathbf{y}^{(3)\top} = [0 \ 0 \ 1]$$

$$\mathbf{x}^{(4)} \rightarrow \text{class3}, \mathbf{y}^{(4)\top} = [0 \ 0 \ 1]$$

$$\mathbf{Y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

- Assume **training set**  $\mathbf{X}$  and random parameters  $\mathbf{W}$  are as below:

$$\mathbf{X} = \begin{bmatrix} 1 & 0.1 & 0.5 \\ 1 & 1.1 & 2.3 \\ 1 & -1.1 & -2.3 \\ 1 & -1.5 & -2.5 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} 0.01 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.3 \\ 0.1 & 0.2 & 0.3 \end{bmatrix}$$

# Softmax Model Estimation and Prediction - Example (2/2)

► Now, let's compute the softmax activation:

$$\hat{y}_j^{(i)} = p(y^{(i)} = j \mid \mathbf{x}^{(i)}; \mathbf{w}_j) = \sigma(\mathbf{w}_j^T \mathbf{x}^{(i)}) = \frac{e^{\mathbf{w}_j^T \mathbf{x}^{(i)}}}{\sum_{l=1}^k e^{\mathbf{w}_l^T \mathbf{x}^{(i)}}}$$

$$\hat{\mathbf{Y}} = \begin{bmatrix} \mathbf{y}^{(1)T} \\ \mathbf{y}^{(2)T} \\ \mathbf{y}^{(3)T} \\ \mathbf{y}^{(4)T} \end{bmatrix} = \begin{bmatrix} 0.29 & 0.34 & \mathbf{0.36} \\ 0.21 & 0.33 & \mathbf{0.46} \\ \mathbf{0.43} & 0.33 & 0.24 \\ \mathbf{0.45} & 0.33 & 0.22 \end{bmatrix}$$

$$\text{the predicted classes} = \begin{bmatrix} 2 \\ 2 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{The correct classes} = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 2 \end{bmatrix}$$

# Softmax Model Estimation and Prediction - Example (2/2)

- Now, let's compute the softmax activation:

$$\hat{y}_j^{(i)} = p(y^{(i)} = j \mid \mathbf{x}^{(i)}; \mathbf{w}_j) = \sigma(\mathbf{w}_j^T \mathbf{x}^{(i)}) = \frac{e^{\mathbf{w}_j^T \mathbf{x}^{(i)}}}{\sum_{l=1}^k e^{\mathbf{w}_l^T \mathbf{x}^{(i)}}}$$

$$\hat{\mathbf{Y}} = \begin{bmatrix} \mathbf{y}^{(1)T} \\ \mathbf{y}^{(2)T} \\ \mathbf{y}^{(3)T} \\ \mathbf{y}^{(4)T} \end{bmatrix} = \begin{bmatrix} 0.29 & 0.34 & \mathbf{0.36} \\ 0.21 & 0.33 & \mathbf{0.46} \\ \mathbf{0.43} & 0.33 & 0.24 \\ \mathbf{0.45} & 0.33 & 0.22 \end{bmatrix}$$

the predicted classes =  $\begin{bmatrix} 2 \\ 2 \\ 0 \\ 0 \end{bmatrix}$

The correct classes =  $\begin{bmatrix} 0 \\ 1 \\ 2 \\ 2 \end{bmatrix}$

- They are terribly wrong.

# Softmax Model Estimation and Prediction - Example (2/2)

- Now, let's compute the softmax activation:

$$\hat{y}_j^{(i)} = p(y^{(i)} = j \mid \mathbf{x}^{(i)}; \mathbf{w}_j) = \sigma(\mathbf{w}_j^T \mathbf{x}^{(i)}) = \frac{e^{\mathbf{w}_j^T \mathbf{x}^{(i)}}}{\sum_{l=1}^k e^{\mathbf{w}_l^T \mathbf{x}^{(i)}}}$$

$$\hat{\mathbf{Y}} = \begin{bmatrix} \mathbf{y}^{(1)T} \\ \mathbf{y}^{(2)T} \\ \mathbf{y}^{(3)T} \\ \mathbf{y}^{(4)T} \end{bmatrix} = \begin{bmatrix} 0.29 & 0.34 & \mathbf{0.36} \\ 0.21 & 0.33 & \mathbf{0.46} \\ \mathbf{0.43} & 0.33 & 0.24 \\ \mathbf{0.45} & 0.33 & 0.22 \end{bmatrix} \quad \text{the predicted classes} = \begin{bmatrix} 2 \\ 2 \\ 0 \\ 0 \end{bmatrix} \quad \text{The correct classes} = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 2 \end{bmatrix}$$

- They are terribly wrong.
- We need to update the weights based on the cost function.



# Softmax Model Estimation and Prediction - Example (2/2)

- Now, let's compute the softmax activation:

$$\hat{y}_j^{(i)} = p(y^{(i)} = j \mid \mathbf{x}^{(i)}; \mathbf{w}_j) = \sigma(\mathbf{w}_j^T \mathbf{x}^{(i)}) = \frac{e^{\mathbf{w}_j^T \mathbf{x}^{(i)}}}{\sum_{l=1}^k e^{\mathbf{w}_l^T \mathbf{x}^{(i)}}}$$

$$\hat{\mathbf{Y}} = \begin{bmatrix} \mathbf{y}^{(1)T} \\ \mathbf{y}^{(2)T} \\ \mathbf{y}^{(3)T} \\ \mathbf{y}^{(4)T} \end{bmatrix} = \begin{bmatrix} 0.29 & 0.34 & \mathbf{0.36} \\ 0.21 & 0.33 & \mathbf{0.46} \\ \mathbf{0.43} & 0.33 & 0.24 \\ \mathbf{0.45} & 0.33 & 0.22 \end{bmatrix} \quad \text{the predicted classes} = \begin{bmatrix} 2 \\ 2 \\ 0 \\ 0 \end{bmatrix} \quad \text{The correct classes} = \begin{bmatrix} 0 \\ 1 \\ 2 \\ 2 \end{bmatrix}$$

- They are terribly wrong.
- We need to update the weights based on the cost function.
- What is the cost function?



# Multinomial Logistic Regression - Cost Function (1/2)

- ▶ The **objective** is to have a model that estimates a **high probability** for the target class, and consequently a **low probability** for the other classes.

# Multinomial Logistic Regression - Cost Function (1/2)

- ▶ The **objective** is to have a model that estimates a **high probability** for the target class, and consequently a **low probability** for the other classes.
- ▶ **Cost function**: the **cross entropy** between the **correct classes** and **predicted class** for all classes.

$$J(\mathbf{w}_j) = -\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^k y_j^{(i)} \log(\hat{y}_j^{(i)})$$

# Multinomial Logistic Regression - Cost Function (1/2)

- ▶ The **objective** is to have a model that estimates a **high probability** for the target class, and consequently a **low probability** for the other classes.
- ▶ **Cost function**: the **cross entropy** between the **correct classes** and **predicted class** for all classes.

$$J(\mathbf{w}_j) = -\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^k y_j^{(i)} \log(\hat{y}_j^{(i)})$$

- ▶  $y_j^{(i)}$  is 1 if the target class for the  $i$ th instance is  $j$ , otherwise, it is 0.



## Multinomial Logistic Regression - Cost Function (2/2)

- If there are two classes ( $k = 2$ ), this cost function is equivalent to the **logistic regression's cost function**.

$$J(\mathbf{w}) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$$



# How to Learn Model Parameters $\mathbf{W}$ ?

- Goal: find  $\mathbf{W}$  that minimizes  $J(\mathbf{W})$ .



# How to Learn Model Parameters $\mathbf{W}$ ?

- ▶ **Goal:** find  $\mathbf{W}$  that **minimizes**  $J(\mathbf{W})$ .
- ▶ Start at a **random point**, and repeat the following **steps**, until the **stopping criterion** is satisfied:



# How to Learn Model Parameters $\mathbf{W}$ ?

- ▶ **Goal:** find  $\mathbf{W}$  that **minimizes**  $J(\mathbf{W})$ .
- ▶ Start at a **random point**, and repeat the following **steps**, until the **stopping criterion** is satisfied:
  1. Determine a **descent direction**  $\frac{\partial J(\mathbf{W})}{\partial \mathbf{w}}$





# How to Learn Model Parameters $\mathbf{W}$ ?

- ▶ **Goal:** find  $\mathbf{W}$  that **minimizes**  $J(\mathbf{W})$ .
- ▶ Start at a **random point**, and repeat the following **steps**, until the **stopping criterion** is satisfied:
  1. Determine a **descent direction**  $\frac{\partial J(\mathbf{W})}{\partial \mathbf{w}}$
  2. Choose a **step size**  $\eta$



# How to Learn Model Parameters $\mathbf{W}$ ?

- ▶ **Goal:** find  $\mathbf{W}$  that **minimizes**  $J(\mathbf{W})$ .
- ▶ Start at a **random point**, and repeat the following **steps**, until the **stopping criterion** is satisfied:
  1. Determine a **descent direction**  $\frac{\partial J(\mathbf{W})}{\partial \mathbf{w}}$
  2. Choose a **step size**  $\eta$
  3. **Update** the parameters:  $\mathbf{w}^{(\text{next})} = \mathbf{w} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{w}}$  (**simultaneously** for all parameters)



# Multinomial Logistic Regression in Spark

```
val training = spark.read.format("libsvm").load("multiclass_data.txt")
```



# Multinomial Logistic Regression in Spark

```
val training = spark.read.format("libsvm").load("multiclass_data.txt")
```

```
import org.apache.spark.ml.classification.LogisticRegression
```

```
val lr = new LogisticRegression().setMaxIter(10).setRegParam(0.3).setElasticNetParam(0.8)  
val lrModel = lr.fit(training)
```



# Multinomial Logistic Regression in Spark

```
val training = spark.read.format("libsvm").load("multiclass_data.txt")
```

```
import org.apache.spark.ml.classification.LogisticRegression
```

```
val lr = new LogisticRegression().setMaxIter(10).setRegParam(0.3).setElasticNetParam(0.8)  
val lrModel = lr.fit(training)
```

```
println(s"Coefficients: \n${lrModel.coefficientMatrix}")  
println(s"Intercepts: \n${lrModel.interceptVector}")
```

# Performance Measures



# Performance Measures

- ▶ Evaluate the performance of a model.
- ▶ Depends on the application and its requirements.
- ▶ There are many different types of classification algorithms, but the evaluation of them share similar principles.



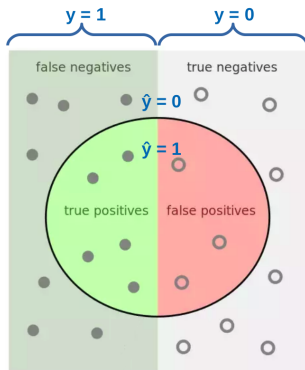
## Evaluation of Classification Models (1/3)

- ▶ In a **classification problem**, there exists a **true output  $y$**  and a **model-generated predicted output  $\hat{y}$**  for each data point.
- ▶ The results for each instance point can be assigned to one of **four categories**:
  - True Positive (TP)
  - True Negative (TN)
  - False Positive (FP)
  - False Negative (FN)



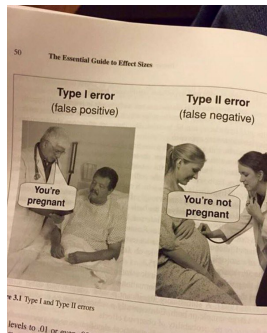
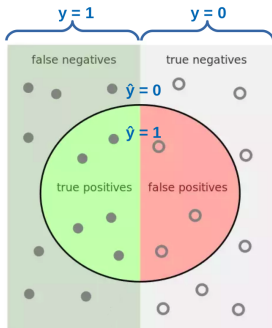
## Evaluation of Classification Models (2/3)

- ▶ True Positive (TP): the label  $y$  is positive and prediction  $\hat{y}$  is also positive.
- ▶ True Negative (TN): the label  $y$  is negative and prediction  $\hat{y}$  is also negative.



## Evaluation of Classification Models (3/3)

- **False Positive (FP)**: the **label  $y$**  is **negative** but **prediction  $\hat{y}$**  is **positive** (**type I error**).
- **False Negative (FN)**: the **label  $y$**  is **positive** but **prediction  $\hat{y}$**  is **negative** (**type II error**).





# Why Pure Accuracy Is Not A Good Metric?

- ▶ **Accuracy**: how **close** the **prediction** is to the **true value**.



# Why Pure Accuracy Is Not A Good Metric?

- ▶ **Accuracy**: how **close** the **prediction** is to the **true value**.
- ▶ Assume a highly **unbalanced dataset**
- ▶ E.g., a dataset where **95%** of the data points are **not fraud** and **5%** of the data points are **fraud**.



# Why Pure Accuracy Is Not A Good Metric?

- ▶ **Accuracy**: how **close** the **prediction** is to the **true value**.
- ▶ Assume a highly **unbalanced dataset**
- ▶ E.g., a dataset where **95%** of the data points are **not fraud** and **5%** of the data points are **fraud**.
- ▶ A **naive classifier** that **predicts not fraud**, regardless of input, will be **95% accurate**.

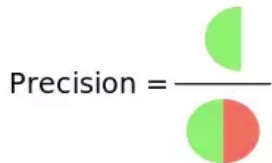
# Why Pure Accuracy Is Not A Good Metric?

- ▶ **Accuracy**: how **close** the **prediction** is to the **true value**.
- ▶ Assume a highly **unbalanced dataset**
- ▶ E.g., a dataset where **95%** of the data points are **not fraud** and **5%** of the data points are **fraud**.
- ▶ A **naive classifier** that **predicts not fraud**, regardless of input, will be **95% accurate**.
- ▶ For this reason, metrics like **precision** and **recall** are typically used.

# Precision

- It is the **accuracy** of the **positive predictions**.

$$\text{Precision} = p(y = 1 \mid \hat{y} = 1) = \frac{TP}{TP + FP}$$

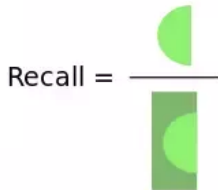


$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

# Recall

- ▶ Is the **ratio** of **positive instances** that are **correctly detected** by the classifier.
- ▶ Also called **sensitivity** or **true positive rate (TPR)**.

$$\text{Recall} = p(\hat{y} = 1 \mid y = 1) = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

A Venn diagram illustrating the formula for Recall. It consists of a large green rectangle representing the total number of positive instances (TP + FN). Inside this rectangle, on the right side, is a green circle representing the total number of positive instances. The left half of the circle is shaded a darker green, representing the True Positives (TP). A horizontal line is drawn across the middle of the diagram, with the word 'Recall =' to its left. The line extends from the text to the right edge of the circle, effectively dividing the circle into the shaded TP half and the unshaded FN half.



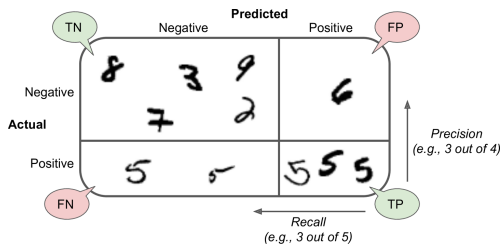
# F1 Score

- ▶ **F1 score**: combine precision and recall into a single metric.
- ▶ The **F1 score** is the harmonic mean of precision and recall.
- ▶ Whereas the regular mean treats all values equally, the harmonic mean gives much more weight to low values.
- ▶ F1 only gets high score if both recall and precision are high.

$$F1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

# Confusion Matrix

- ▶ The **confusion matrix** is  $K \times K$ , where  $K$  is the **number of classes**.
- ▶ It shows the **number of correct and incorrect predictions** made by the classification model **compared to the actual outcomes** in the data.



# Confusion Matrix - Example

|        |          | Predicted                      |                                |  |
|--------|----------|--------------------------------|--------------------------------|--|
|        |          | Negative                       | Positive                       |  |
| Actual | Negative | <div>TN</div> <div>8 3 9</div> | <div>FP</div> <div>6</div>     |  |
|        | Positive | <div>FN</div> <div>5 5</div>   | <div>TP</div> <div>5 5 5</div> |  |

Precision (e.g., 3 out of 4)

Recall (e.g., 3 out of 5)

$$TP = 3, TN = 5, FP = 1, FN = 2$$

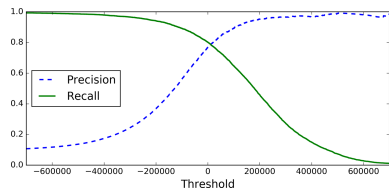
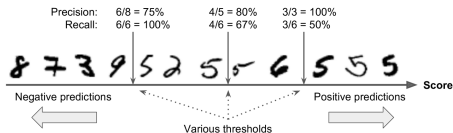
$$\text{Precision} = \frac{TP}{TP + FP} = \frac{3}{3 + 1} = \frac{3}{4}$$

$$\text{Recall (TPR)} = \frac{TP}{TP + FN} = \frac{3}{3 + 2} = \frac{3}{5}$$

$$\text{FPR} = \frac{FP}{TN + FP} = \frac{1}{5 + 1} = \frac{1}{6}$$

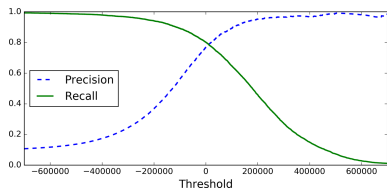
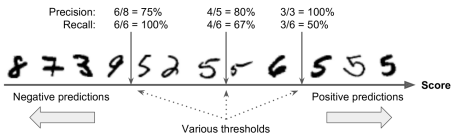
# Precision-Recall Tradeoff

- Precision-recall tradeoff: increasing precision reduces recall, and vice versa.



# Precision-Recall Tradeoff

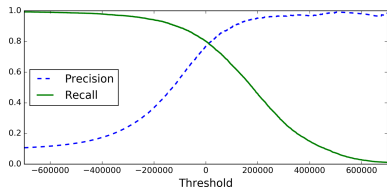
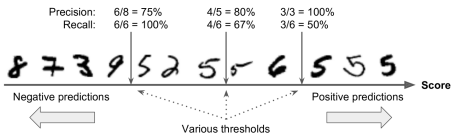
- **Precision-recall tradeoff:** increasing precision reduces recall, and vice versa.
- Assume a classifier that detects number 5 from the other digits.
  - If an instance score is greater than a threshold, it assigns it to the positive class, otherwise to the negative class.



-

# Precision-Recall Tradeoff

- **Precision-recall tradeoff**: increasing precision reduces recall, and vice versa.
- Assume a classifier that detects number 5 from the other digits.
  - If an instance score is greater than a threshold, it assigns it to the positive class, otherwise to the negative class.
- Raising the threshold (move it to the arrow on the right), the false positive (the 6) becomes a true negative, thereby increasing precision.
- Lowering the threshold increases recall and reduces precision.



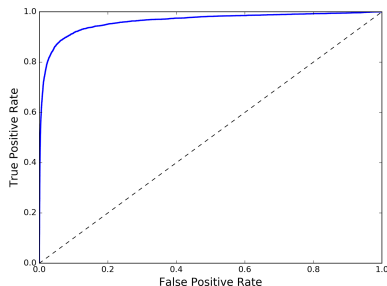
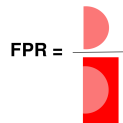
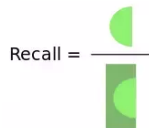
- $$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

**FPR =** 



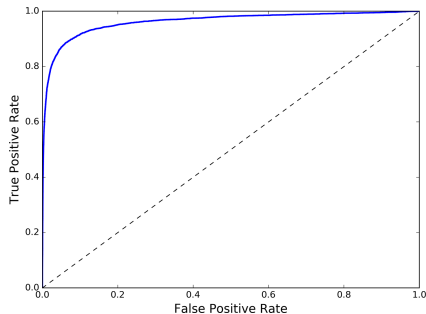
# The ROC Curve (1/2)

- ▶ True positive rate (TPR) (recall):  $p(\hat{y} = 1 \mid y = 1)$
- ▶ False positive rate (FPR):  $p(\hat{y} = 1 \mid y = 0)$
- ▶ The **receiver operating characteristic (ROC)** curves summarize the **trade-off** between the **TPR** and **FPR** for a model using different probability **thresholds**.



## The ROC Curve (2/2)

- ▶ Here is a **tradeoff**: the **higher** the **TPR**, the **more** **FPR** the classifier produces.
- ▶ The **dotted line** represents the ROC curve of a **purely random** classifier.
- ▶ A **good classifier** moves toward the **top-left** corner.
- ▶ **Area under the curve (AUC)**





# Binomial Logistic Regression Measurements in Spark

```
val lr = new LogisticRegression()  
val lrModel = lr.fit(training)
```

# Binomial Logistic Regression Measurements in Spark

```
val lr = new LogisticRegression()
val lrModel = lr.fit(training)

val trainingSummary = lrModel.binarySummary

// obtain the objective per iteration.
val objectiveHistory = trainingSummary.objectiveHistory
objectiveHistory.foreach(loss => println(loss))

// obtain the ROC as a dataframe and areaUnderROC.
val roc = trainingSummary.roc
roc.show()
println(s"areaUnderROC: ${trainingSummary.areaUnderROC}")

// set the model threshold to maximize F-Measure
val fMeasure = trainingSummary.fMeasureByThreshold
val maxFMeasure = fMeasure.select(max("F-Measure")).head().getDouble(0)
val bestThreshold = fMeasure.where($"F-Measure" === maxFMeasure)
    .select("threshold").head().getDouble(0)
lrModel.setThreshold(bestThreshold)
```



## Multinomial Logistic Regression in Spark (1/2)

```
val trainingSummary = lrModel.summary

// for multiclass, we can inspect metrics on a per-label basis
println("False positive rate by label:")
trainingSummary.falsePositiveRateByLabel.zipWithIndex.foreach { case (rate, label) =>
  println(s"label $label: $rate")
}

println("True positive rate by label:")
trainingSummary.truePositiveRateByLabel.zipWithIndex.foreach { case (rate, label) =>
  println(s"label $label: $rate")
}
```

## Multinomial Logistic Regression in Spark (2/2)

```
println("Precision by label:")
trainingSummary.precisionByLabel.zipWithIndex.foreach { case (prec, label) =>
  println(s"label $label: $prec")
}

println("Recall by label:")
trainingSummary.recallByLabel.zipWithIndex.foreach { case (rec, label) =>
  println(s"label $label: $rec")
}

val accuracy = trainingSummary.accuracy
val falsePositiveRate = trainingSummary.weightedFalsePositiveRate
val truePositiveRate = trainingSummary.weightedTruePositiveRate
val fMeasure = trainingSummary.weightedFMeasure
val precision = trainingSummary.weightedPrecision
val recall = trainingSummary.weightedRecall
```

# Summary



# Summary

- ▶ Binomial logistic regression
  - $y \in \{0, 1\}$
  - Sigmoid function
  - $-\log$  as the cost function
  - Minimize the cross-entropy
  
- ▶ Multinomial logistic regression
  - $y \in \{0, 1, \dots, k\}$
  - Softmax function
  - $-\log$  as the cost function
  
- ▶ Performance measurements
  - TP, TF, FP, FN
  - Precision, recall, F1
  - Threshold and ROC





## Reference

- ▶ Ian Goodfellow et al., Deep Learning (Ch. 4, 5)
- ▶ Aurélien Géron, Hands-On Machine Learning (Ch. 3)
- ▶ Matei Zaharia et al., Spark - The Definitive Guide (Ch. 26)

Questions?