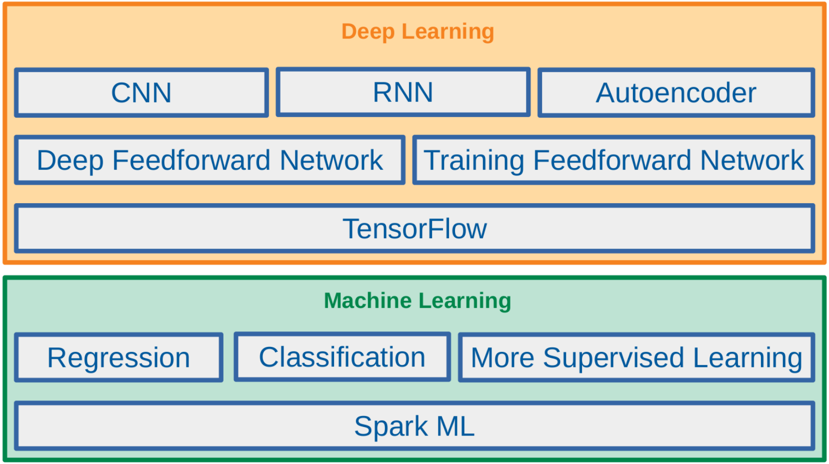
Machine Learning - Classification

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

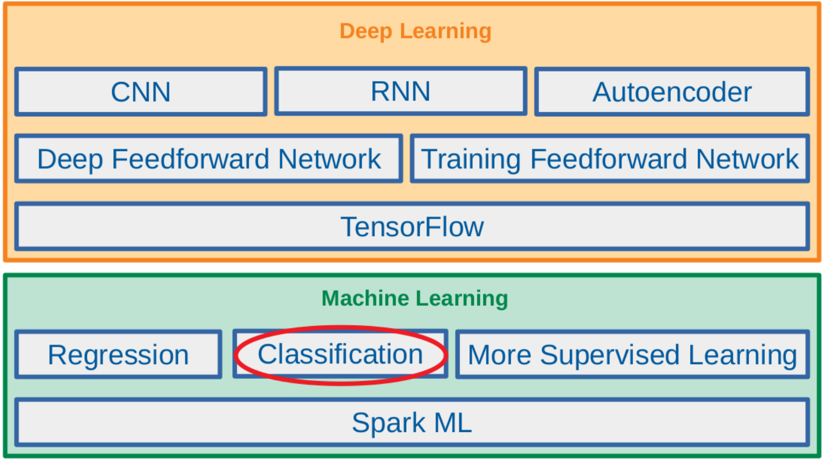
The Course Web Page

https://id2223kth.github.io

Where Are We?


Deep Learning
CNN
RNN
Autoencoder
Deep Feedforward Network
Training Feedforward Network
TensorFlow
Machine Learning
Regression
Classification
More Supervised Learning
Spark ML

Where Are We?


Deep Learning
CNN
RNN
Autoencoder
Deep Feedforward Network
Training Feedforward Network
TensorFlow
Machine Learning
Regression
Classification
More Supervised Learning
Spark ML

Let's Start with an Example

Example (1/4)

- Given the dataset of `m` cancer tests.

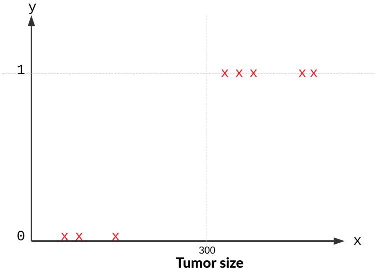
Tumor size	Cancer
330	1
120	0
400	1
⋮	⋮

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

Example (2/4)

Tumor size	Cancer
330	1
120	0
400	1
⋮	⋮

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

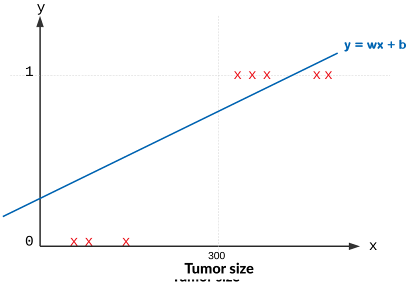


- $\mathbf{x}^{(\mathtt{i})} \in \mathbb{R}$: $\mathtt{x}_1^{(\mathtt{i})}$ is the tumor size of the $\mathtt{i}$th instance in the training set.

Example (3/4)

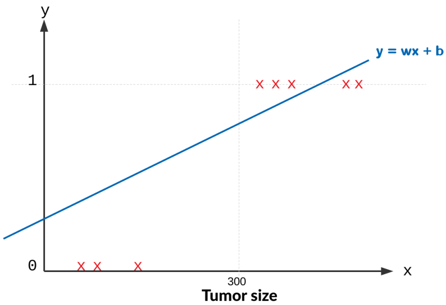
Tumor size	Cancer
330	1
120	0
400	1
⋮	⋮

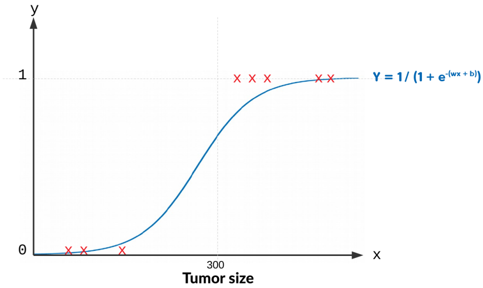
$$\mathbf{x} = \begin{bmatrix} 330 \\ 120 \\ 400 \\ \vdots \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$



- 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: $\hat{y} = f_w(\mathbf{x}) = w_0 + w_1 x_1$
- Bad model!

Example (4/4)



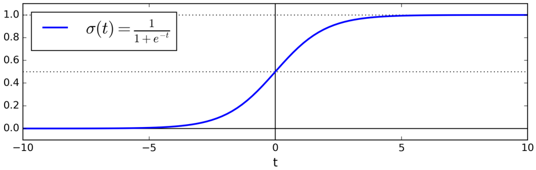


- A better model $\hat{y} = \frac{1}{1+e^{-(w_0+w_1x_1)}}$

Sigmoid Function

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

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

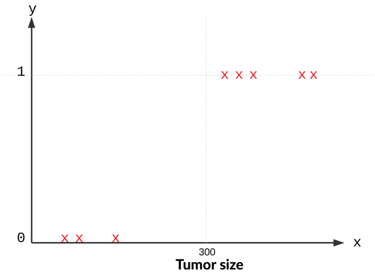


- When $\mathtt{t} < 0$, then $\sigma(\mathtt{t}) < 0.5$
- when $\mathtt{t} \geq 0$, then $\sigma(\mathtt{t}) \geq 0.5$

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).

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

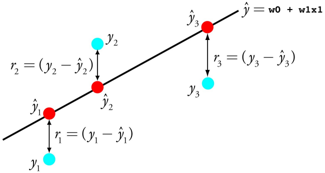
- Binomial logistic regression: the model computes a weighted sum of the input features (plus a bias term), but it outputs the logistic of this result.

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

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

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)

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

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

Binomial Logistic Regression - Cost Function (1/5)

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

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

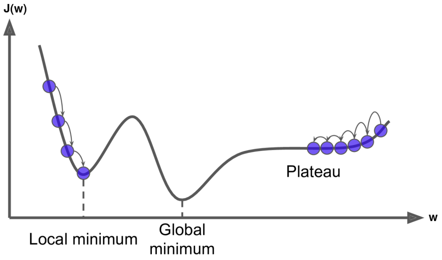
$$J(\mathbf{w}) = \frac{1}{m}\sum_{i}^{m}\mathtt{cost}(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{m}\sum_{i}^{m}(\hat{y}^{(i)} - y^{(i)})^2$$

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

- 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.
- We want a convex logistic regression cost function $J(\mathbf{w})$.



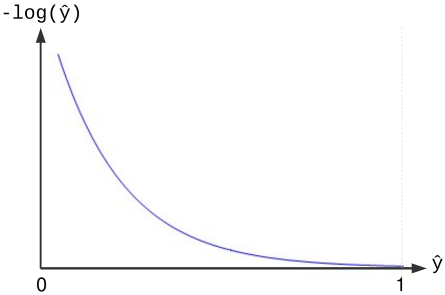
Binomial Logistic Regression - Cost Function (3/5)

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

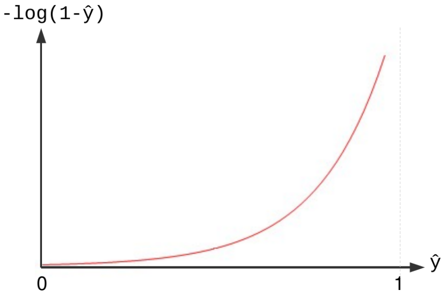
$$\texttt{cost}(\hat{y}^{(i)}, y^{(i)}) = \begin{cases} -\texttt{log}(\hat{y}^{(i)}) & \text{if} \quad y^{(i)} = 1 \\ -\texttt{log}(1 - \hat{y}^{(i)}) & \text{if} \quad y^{(i)} = 0 \end{cases}$$

Binomial Logistic Regression - Cost Function (4/5)

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



when $y = 1$



when $y = 0$

Binomial Logistic Regression - Cost Function (5/5)

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

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

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

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 descent
  - Mini-batch gradient descent

Binomial Logistic Regression Gradient Descent (1/3)

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

Binomial Logistic Regression Gradient Descent (2/3)

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

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

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

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

Binomial Logistic Regression Gradient Descent (3/3)

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

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

Tumor size	Cancer
330	1
120	0
400	1

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

- 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)

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

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

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

$$\begin{aligned}\frac{\partial J(\mathbf{w})}{\partial w_0} &= \frac{1}{3}\sum_{i}^{3}(\hat{y}^{(i)} - y^{(i)})x_0 \\ &= \frac{1}{3}[(\frac{1}{1 + e^{-(w_0 + 330 w_1)}} - 1) + (\frac{1}{1 + e^{-(w_0 + 120 w_1)}} - 0) + (\frac{1}{1 + e^{-(w_0 + 400 w_1)}} - 1)]\end{aligned}$$

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

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

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

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

$$\frac{\partial J(\mathbf{w})}{\partial w_1} = \frac{1}{3} \sum_{i}^{3} (\hat{y}^{(i)} - y^{(i)}) x_1$$

$$= \frac{1}{3} [330(\frac{1}{1 + e^{-(w_0 + 330 w_1)}} - 1) + 120(\frac{1}{1 + e^{-(w_0 + 120 w_1)}} - 0) + 400(\frac{1}{1 + e^{-(w_0 + 400 w_1)}} - 1)]$$

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

$$\mathtt{w}_0^{(\mathtt{next})} = \mathtt{w}_0 - \eta \frac{\partial \mathrm{J}(\mathbf{w})}{\partial \mathtt{w}_0}$$

$$\mathtt{w}_1^{(\mathtt{next})} = \mathtt{w}_1 - \eta \frac{\partial \mathrm{J}(\mathbf{w})}{\partial \mathtt{w}_1}$$

Binomial Logistic Regression in Spark

```scala
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
```

```scala
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)
```

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

- $\mathtt{p}(\mathtt{X} \mid \theta = \frac{2}{3})$ is the probability of $\mathtt{X}$ given $\theta = \frac{2}{3}$.

- $\mathtt{p}(\mathtt{X} = \mathtt{h} \mid \theta)$ is the likelihood of θ given $\mathtt{X} = \mathtt{h}$.

- Likelihood ($\mathtt{L}$): a function of the parameters (θ) of a probability model, given specific observed data, e.g., $\mathtt{X} = \mathtt{h}$.

$$\mathtt{L}(\theta) = \mathtt{p}(\mathtt{X} \mid \theta)$$

Probability and Likelihood (2/2)

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

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

Likelihood and Log-Likelihood

- The Likelihood product is prone to numerical underflow.

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

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

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

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

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

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

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

- So the likelihood is:

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

- 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:

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

- 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
- Coss-entropy: quantify the difference (error) between two probability distributions.
- How close is the predicted distribution to the true distribution?

$$\mathtt{H}(\mathtt{p}, \mathtt{q}) = -\sum_{\mathtt{j}} \mathtt{p}_{\mathtt{j}} \mathtt{log}(\mathtt{q}_{\mathtt{j}})$$

- Where $\mathtt{p}$ is the true distriution, and $\mathtt{q}$ is the predicted distribution.

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.

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

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

- 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 $\mathtt{y} \in \{0, 1\}$, we have $\mathtt{y} \in \{1, 2, \cdots, \mathtt{k}\}$.

Binomial vs. Multinomial Logistic Regression (1/2)

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

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

- In multinomial classifier, $y \in \{1, 2, \cdots, k\}$, 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 $\mathbf{W}$.

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

Binomial vs. Multinomial Logistic Regression (2/2)

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

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

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

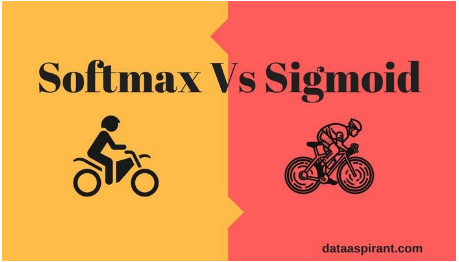
- In multiclasses, $y \in \{1, 2, \cdots, k\}$, we use the softmax function.

$$\mathbf{w}_j^\mathsf{T}\mathbf{x} = w_{0,j} x_0 + w_{1,j} x_1 + \cdots + w_{n,j} x_n, 1 \leq j \leq k$$

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

Sigmoid vs. Softmax

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



How Does Softmax Work? - Step 1

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

$$\mathbf{w}_j^\intercal \mathbf{x}^{(i)} = w_{0,j} x_0^{(i)} + w_{1,j} x_1^{(i)} + \cdots + w_{n_j} x_n^{(i)}$$

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

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

How Does Softmax Work? - Step 2

- For each instance $\mathbf{x}^{(i)}$, apply the softmax function on its scores: $\mathbf{w}_1^T\mathbf{x}^{(i)}, \cdots, \mathbf{w}_k^T\mathbf{x}^{(i)}$
- Estimates the probability that the instance $\mathbf{x}^{(i)}$ belongs to class j.

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

- k: the number of classes.
- $\mathbf{w}_j^T\mathbf{x}^{(i)}$: the scores of class j for the instance $\mathbf{x}^{(i)}$.
- $\sigma(\mathbf{w}_j^T\mathbf{x}^{(i)})$: the estimated probability that $\mathbf{x}^{(i)}$ belongs to class j.

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 $\mathtt{m} = 4$ instances from $\mathtt{k} = 3$ classes.

$$\begin{aligned}
\mathbf{x}^{(1)} &\rightarrow \mathtt{class1}, \mathbf{y}^{(1)\intercal} = [1\ 0\ 0] \\
\mathbf{x}^{(2)} &\rightarrow \mathtt{class2}, \mathbf{y}^{(2)\intercal} = [0\ 1\ 0] \\
\mathbf{x}^{(3)} &\rightarrow \mathtt{class3}, \mathbf{y}^{(3)\intercal} = [0\ 0\ 1] \\
\mathbf{x}^{(4)} &\rightarrow \mathtt{class3}, \mathbf{y}^{(4)\intercal} = [0\ 0\ 1]
\end{aligned}
\qquad
\mathbf{Y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

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

$$\mathbf{X} = \left[\begin{array}{c|cc} 1 & 0.1 & 0.5 \\ 1 & 1.1 & 2.3 \\ 1 & -1.1 & -2.3 \\ 1 & -1.5 & -2.5 \end{array}\right]
\qquad
\mathbf{W} = \begin{bmatrix} 0.01 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.3 \\ 0.1 & 0.2 & 0.3 \end{bmatrix}$$

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

- Now, let's compute the softmax activation:

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

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

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

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

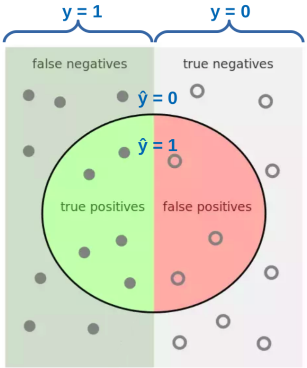
- Evaluate the performance of a model.
- Depends on the application and its requirements.
- There are many different types of classification algorithms, but the evaluation of them share similar principles.

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)

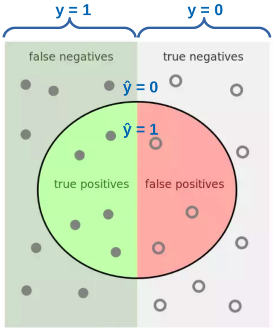
Evaluation of Classification Models (2/3)

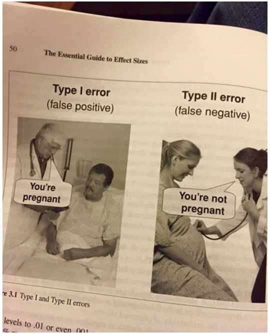
- True Positive (TP): the label y is positive and prediction $\hat{y}$ is also positive.
- True Negative (TN): the label y is negative and prediction $\hat{y}$ is also negative.



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).





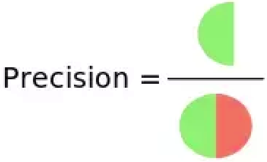
Why Pure Accuracy Is Not A Good Metric?

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

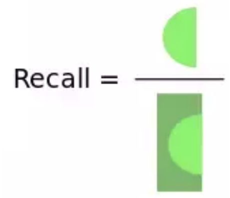
$$\texttt{Precision} = \mathrm{p}(\mathrm{y} = 1 \mid \hat{\mathrm{y}} = 1) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$



Recall

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

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



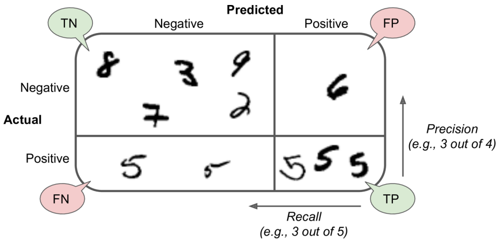
F1 Score

- $F1$ 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.

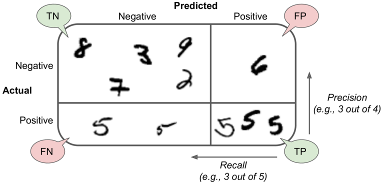
$$\mathtt{F1} = \frac{2}{\frac{1}{\mathtt{precision}} + \frac{1}{\mathtt{recall}}}$$

Confusion Matrix

- The confusion matrix is $\mathtt{K} \times \mathtt{K}$, where $\mathtt{K}$ 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



$$\mathrm{TP} = 3, \mathrm{TN} = 5, \mathrm{FP} = 1, \mathrm{FN} = 2$$

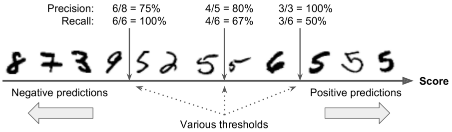
$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = \frac{3}{3+1} = \frac{3}{4}$$

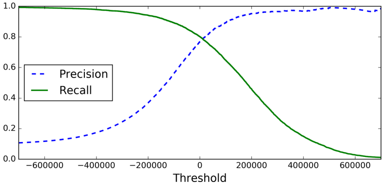
$$\mathrm{Recall\ (TPR)} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = \frac{3}{3+2} = \frac{3}{5}$$

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

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.

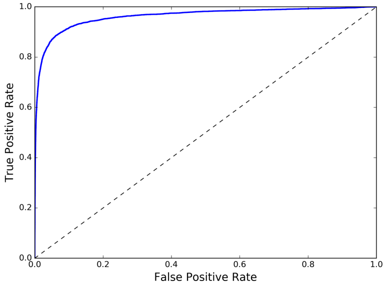




The ROC Curve (1/2)

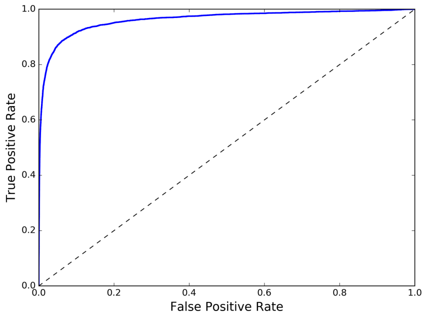
- True positive rate (TPR) (recall): $p(\hat{y} = 1 \mid y = 1)$
- False positive rate (FPR): $p(\hat{y} = 1 \mid y = 0)$

- 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

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

```scala
val trainingSummary = lrModel.binarySummary

// obtain the objective per iteration.
val objectiveHistory = trainingSummary.objectiveHistor
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)

```scala
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)

```scala
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
  - Minimize the cross-entropy
- Multinomial logistic regression
  - $y \in \{1, 2, \cdots, k\}$
  - Softmax function
  - Minimize the cross-entropy
- 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?