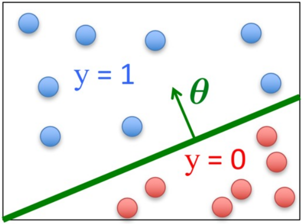
Logistic Regression

Classification

Email: Spam / Not Spam?
Online Transactions: Fraudulent (Yes / No)?
Tumor: Malignant / Benign?



$y \in \{0,1\}$

- 0: Negative Class (benign tumor)
- 1: Positive Class (malignant tumor)

→ **Binary classification**

Note: $y \in \{0,1,2,3, \ldots\}$: **Multi-class classification** is an extension of binary classification

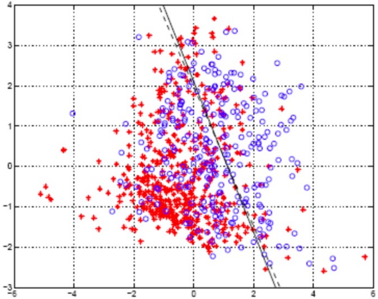
Classification Based on Probability

Instead of just predicting the class, give the probability of the instance being that class, i.e., learn $p(y|x)$

Recall that:

$$0 \leq p(\text{event}) \leq 1$$

$$p(\text{event}) + p(\neg\text{event}) = 1$$



Interpretation of Hypothesis Output

$h_{\boldsymbol{\theta}}(\boldsymbol{x})$ = estimated $p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$

Example: Cancer diagnosis from tumor size

$$\boldsymbol{x} = \left[\begin{array}{c} x_0 \\ x_1 \end{array} \right] = \left[\begin{array}{c} 1 \\ \text{tumorSize} \end{array} \right]$$

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 0.7$$

→ Tell patient that 70% chance of tumor being malignant

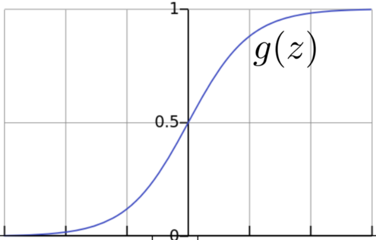
Note that: $p(y = 0 \mid \boldsymbol{x}; \boldsymbol{\theta}) + p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 1$

Therefore, $p(y = 0 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 1 - p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$

Logistic Regression

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}\right)$$

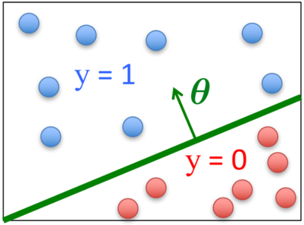
$$g(z) = \frac{1}{1+e^{-z}}$$



$\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}$ should be large negative values for negative instances

$\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}$ should be large positive values for positive instances

- Assume a threshold and...
  - Predict y = 1 if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq 0.5$
  - Predict y = 0 if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0.5$



Classification

Classification: y=0 or y=1, but $h_\theta(x)$ can be >1 or <0

Logistic regression: $0 \leq h_\theta(x) \leq 1$

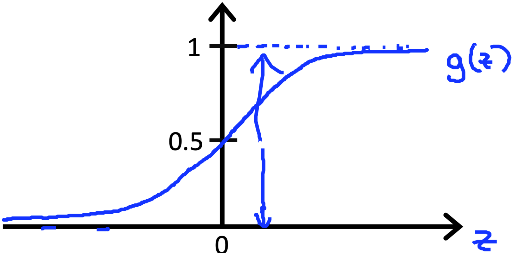
→ use **Sigmoid / Logistic Function**

Logistic Regression Model

We want our classifier to output values between 0 and 1

- When using linear regression we did $h_\theta(x) = \theta^T x$
- For classification hypothesis representation we do $h_\theta(x) = g(\theta^T x)$ where $g(z) = \frac{1}{1+e^{-z}}$ is a Sigmoid (or Logistic) function.

Thus $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$



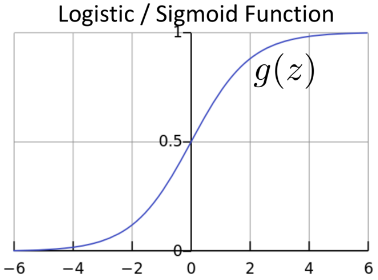
Logistic Regression

- Takes a probabilistic approach to learning discriminative functions (i.e., a classifier)
- $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ should give $p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$
  - Want $0 \leq h_{\boldsymbol{\theta}}(\boldsymbol{x}) \leq 1$
- Logistic regression model:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^\intercal \boldsymbol{x}\right)$$

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

$$\boxed{h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^\intercal \boldsymbol{x}}}}$$



Logistic Regression

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$

m examples $\qquad x \in \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix}}$ $\mathbb{R}^{n+1}$ $\qquad x_0 = 1, y \in \{0, 1\}$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

How to choose parameters θ ?

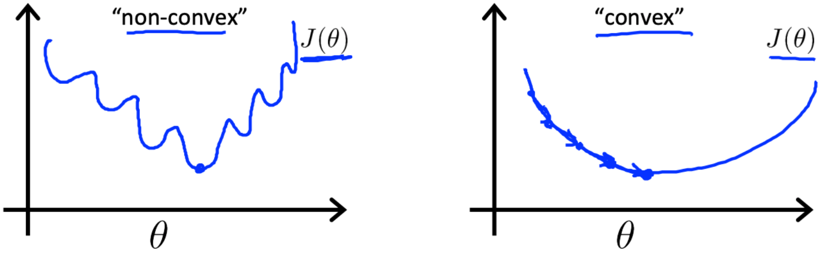
A **non-convex** loss function with MSE cost function

Linear regression: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

$$\mathrm{Cost}\left(h_\theta(x^{(i)}) - y^{(i)} \right) = \frac{1}{2} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

But if $h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$, the cost function is **non-convex** (with MSE).

→ Easy to be trapped at a local minimum → try to make it **convex.**



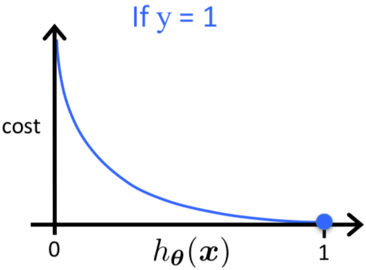
A **convex** logistic regression cost function

$$\operatorname{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right)=\begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y=1 \\ -\log(1-h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y=0 \end{cases}$$

This is the penalty the algorithm pays.

Intuition Behind the Objective Function

$$\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} \boxed{-\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad \text{if } y = 1} \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad \text{if } y = 0 \end{cases}$$

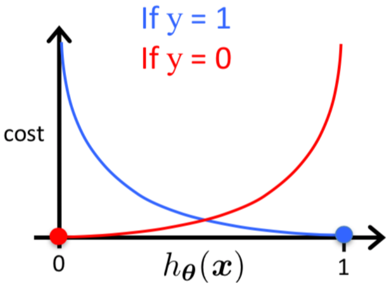


If y = 1

- Cost = 0 if prediction is correct
- As $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \to 0, \text{cost} \to \infty$
- Captures intuition that larger mistakes should get larger penalties
  - e.g., predict $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 0$, but y = 1

Intuition Behind the Objective Function

$$\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} \boxed{-\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad \text{if } y = 1} \\ \boxed{-\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad \text{if } y = 0} \end{cases}$$



If y = 0

- Cost = 0 if prediction is correct
- As $(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) \rightarrow 0, \text{cost} \rightarrow \infty$
- Captures intuition that larger mistakes should get larger penalties

Cost Function Simplification

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{cost}\left(h_\theta(x^{(i)}), y^{(i)}\right)$$

$$\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

Note: $y = 0$ or $y = 1$ always

How to rewrite (simplify) the cost function $\boldsymbol{J(\theta)}$?

Cost Function Simplification

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{cost}\left(h_\theta(x^{(i)}), y^{(i)}\right)$$

$$\text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

Note: $y = 0$ or $y = 1$ always

$$\boxed{\text{cost}\left(h_\theta(x), y\right) = -y\log(h_\theta(x)) - (1-y)\log(1 - h_\theta(x))}$$

$$\text{If } y = 1 : \text{cost}\left(h_\theta(x), y\right) = -\log(h_\theta(x))$$

$$\text{If } y = 0 : \text{cost}\left(h_\theta(x), y\right) = -\log(1 - h_\theta(x))$$

Logistic Regression Cost Function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{cost}\left(h_\theta(x^{(i)}), y^{(i)}\right)$$

$$= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right]$$

To fit parameters θ:

$$\text{Compute } \min_\theta J(\theta) \rightarrow \text{Get } \theta$$

To make prediction given new *x:*

$$\text{Output } h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} = p(y = 1|x, \theta)$$

Gradient Descent

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$\}$

where $$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}$$

Gradient Descent

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$\}$

This looks IDENTICAL to linear regression!!!

- Ignoring the 1/m constant
- However, the form of the model is very different:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^\top \boldsymbol{x}}}$$

Gradient Descent with Regularization

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

$$J_{regularized}(\theta) = J(\theta) + \lambda\sum_{j=1}^{d}\theta_j^2 = J(\theta) + \lambda||\theta_{[1:d]}||_2^2$$

Want $\min_\theta J(\theta)$:

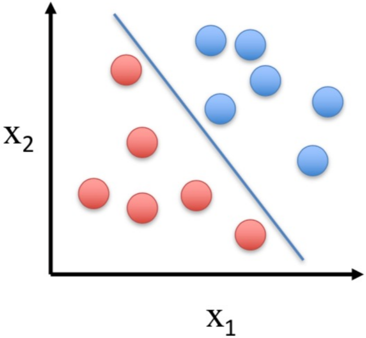
Repeat $\{$

$$\theta_j := \theta_j - \alpha\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \; -\lambda\theta_j$$

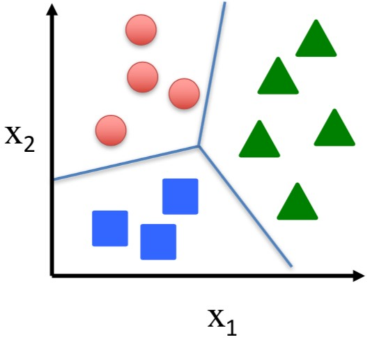
$\}$

Multi-class classification

Binary classification:



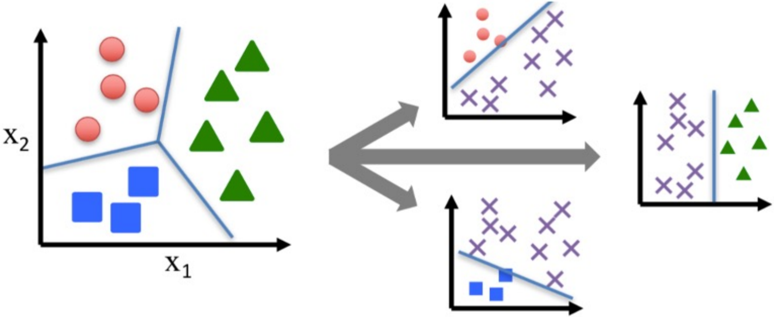
Multi-class classification:



Disease diagnosis: healthy / cold / flu / pneumonia

Object classification: desk / chair / monitor / bookcase

One-vs-all (one-vs rest)

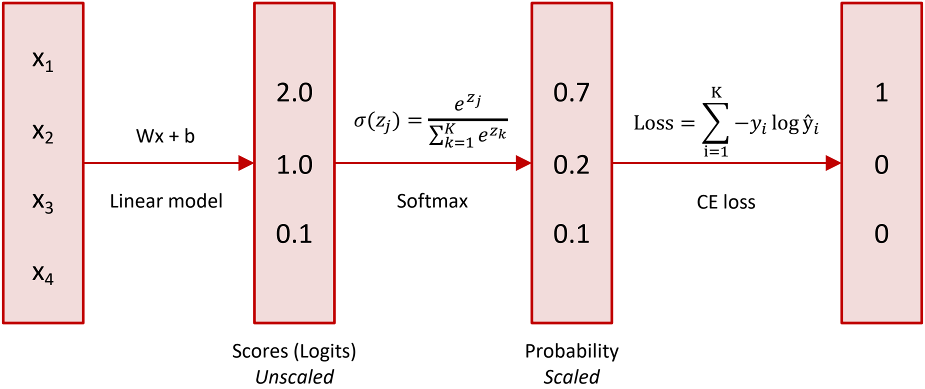


Take the max-probability class among all logistic regression classifiers.

Extra reading: **softmax regression**.

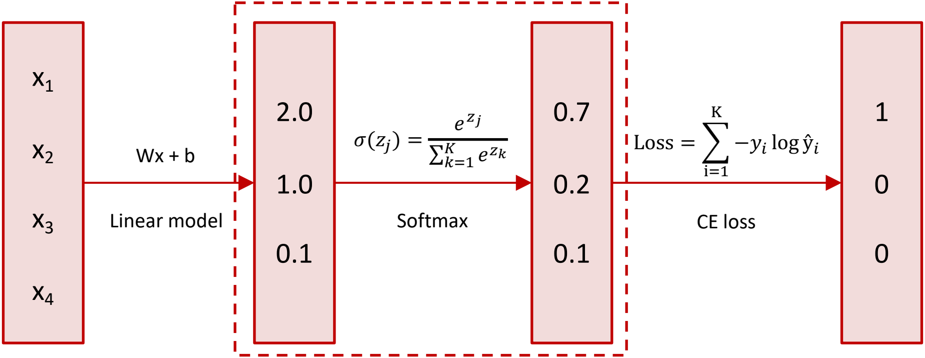
Softmax regression

Softmax regression (or multinomial logistic regression) is a generalization of logistic regression to the case where we want to handle multiple classes.



Softmax regression

Softmax regression (or multinomial logistic regression) is a generalization of logistic regression to the case where we want to handle multiple classes.



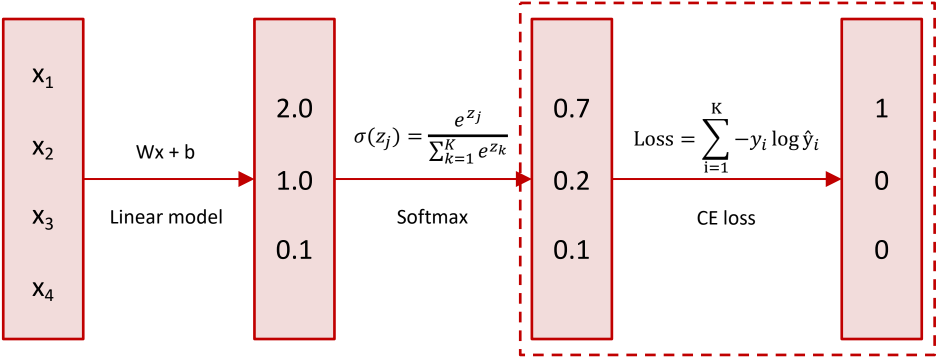
$$\sigma(2.0) = \frac{e^{2.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = 0.7$$

$$\sigma(0.1) = \frac{e^{0.1}}{e^{2.0} + e^{1.0} + e^{0.1}} = 0.1$$

$$\sigma(1.0) = \frac{e^{1.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = 0.2$$

Softmax regression

Softmax regression (or multinomial logistic regression) is a generalization of logistic regression to the case where we want to handle multiple classes.



$$\text{Loss} = -1 \times log_2 0.7 - 0 \times log_2 0.2 - 0 \times log_2 0.1 = 0.51$$

Evaluation metrics

General method: calculate the **difference** between ground-truth labels and model predictions.

Example: testing 165 emails in a spam/non-spam classification problem.

	Prediction YES	Prediction NO
Actual YES	100	5
Actual NO	10	50

Evaluation metrics

Example: testing 165 emails in a spam/non-spam classification problem.

	Prediction YES	Prediction NO
Actual YES	100	5
Actual NO	10	50

- Precision = 100/(100+10) ~ 91%: how many predicted items are relevant.
- Recall = 100/(100+5) ~ 95%: how many relevant items are predicted.

Evaluation metrics

Example: testing 165 emails in a spam/non-spam classification problem.

	Prediction YES	**Prediction NO**
Actual YES	True Positive TP	False Negative FN
Actual NO	False Positive FP	True Negative TN

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

$$Recall = \frac{TP}{TP + FN}$$

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

Summary

Binary Classification

Decision Boundary

Logistic Regression

- Sigmoid function
- Cost Function
- Optimization
- Regularization

Multi-class (Multinomial Classification)

- One-vs-all
- Softmax regression

Evaluation metrics

Q&A

Thank you