Logistic Regression

Wenzhen Zhu, Sayantan Bhadra, Cancan Li, Charlie Wu, Chih Yun Pai

Washington University in St. Louis

CSE 543T Algorithms for Nonlinear Optimization

September 26, 2017

Overview

Section 1: Introduction

Binary Classification

- Key problem in supervised machine learning.
- Predict binary output from input variables, e.g.
  - Spam or not spam
  - Disease or no disease
  - Rain or no rain
  - Win or lose election etc.
- Simplest classifiers: linear classifiers, e.g. *perceptron*.

Linear classification

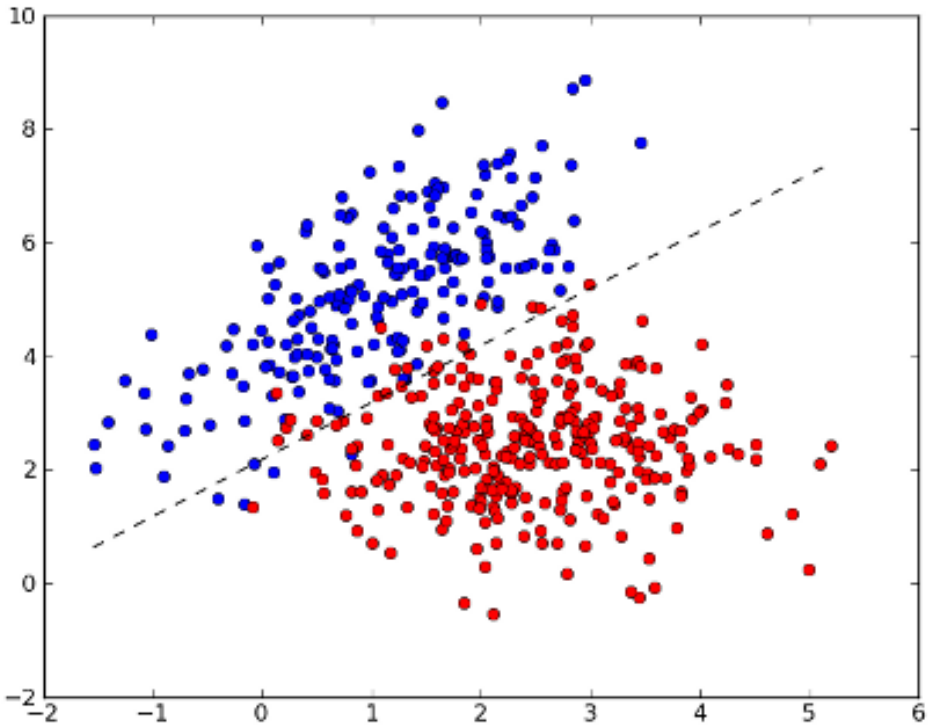


Figure: Example of linear classification

Linear Classification

- Output binary class (0 or 1) based on where a linear combination of the input feature vector lies with respect to the decision boundary:

$$y = \frac{1}{2}(1 + \text{sign}(\theta^T \mathbf{x})) \tag{1}$$

$\mathbf{x} \in \mathbb{R}^{d+1}, \theta \in \mathbb{R}^{d+1}; y \in \{0, 1\}$.

  - $\theta^T \mathbf{x} = 0 \implies$ decision boundary
  - $\theta^T \mathbf{x} > 0 \implies y = 1$
  - $\theta^T \mathbf{x} < 0 \implies y = 0$

- The decision boundary (i.e. θ) is learned from the training data.

Hard vs. Soft Threshold

- Exact binary prediction (*hard threshold*) may be impractical - does not reveal the confidence in the prediction.
- Express as *probability* of belonging to a class (*soft threshold*), e.g. probability of raining, probability of a heart disease etc.
- Instead of a strict binary value, the classifier output is a real number between 0 and 1 - interpreted as class probability.
- This leads to **logistic regression**.

Modeling Conditional Probabilities

- Determine a conditional probability distribution, or *likelihood* $P(Y|X)$ that best describes the observed data.
- Assuming $P(Y = 1|X = \mathbf{x}) = p(\mathbf{x}; \theta)$ for some function p parameterized by θ and with $i.i.d.$ observation variables, the conditional likelihood function $P(Y|X)$ is,

$$\prod_{i=1}^{n} P(Y = y_i|X = \mathbf{x}_i) = \prod_{i=1}^{n} p(\mathbf{x}_i; \theta)^{y_i}(1 - p(\mathbf{x}_i; \theta)^{1-y_i}) \quad (2)$$

- Estimate θ by *maximum likelihood estimation*.

Section 2: Logistic Regression Model

Fisher's Iris

The data set consists of 50 samples from each of three species of iris flowers. Four features were measured from each sample, they are the length and the width of sepal and petal. This data set is often used to demonstrate classification techniques and discriminant analysis.

(a) setosa

(b) versicolor

(c) virginica

Fisher's Iris

Four features were measured from each sample, they are the length and the width of sepal and petal.

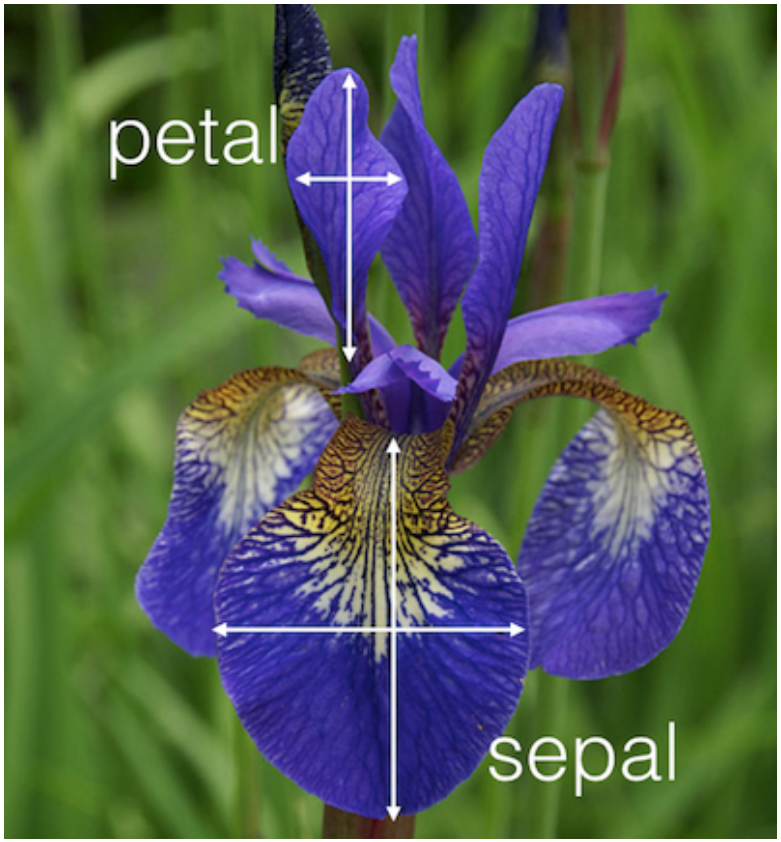
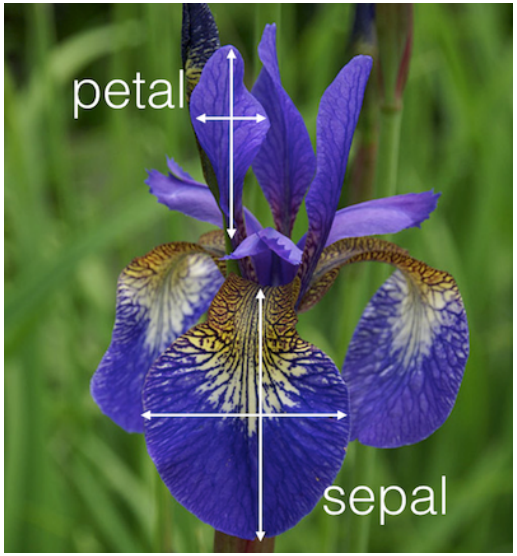


Figure: Features

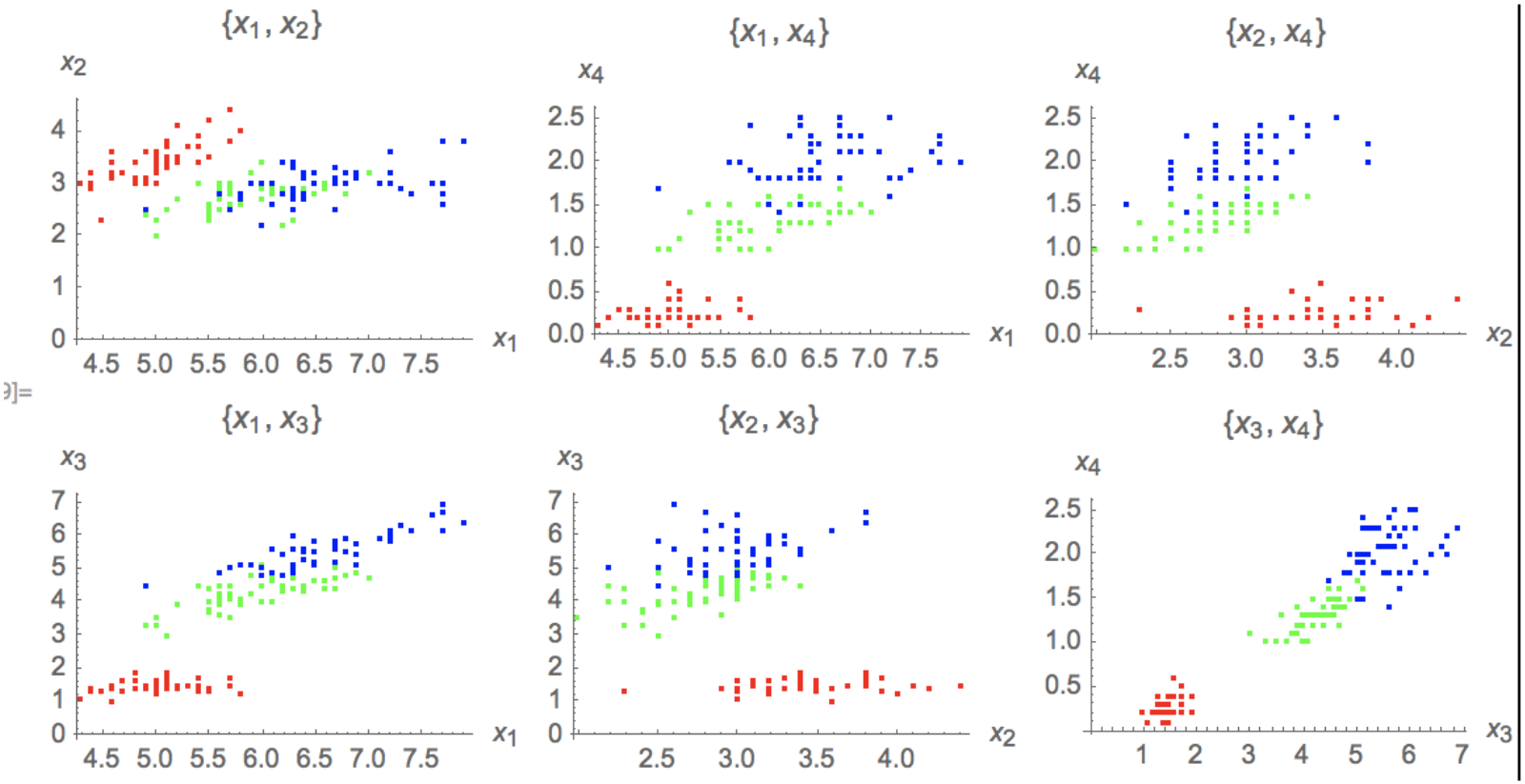
Sample Data

Sepal Length	Sepal Width	Petal Length	Petal Width	Species
5.8	4.0	1.2	0.2	setosa
6.4	2.8	5.6	2.1	virginica
6.7	3.1	5.6	2.4	virginica

Table: Fisher's Iris Dataset Sample



Visualization of Iris Dataset



Let's start from binary classification $y \in \{0, 1\}$

$y \in \{0, 1\}$ where $0 \rightarrow$ versicolor $1 \rightarrow$ virginica

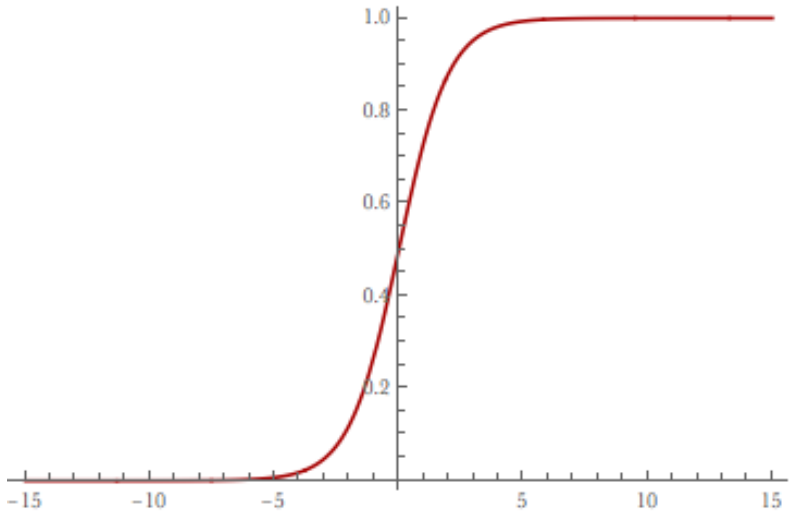
$\mathbf{x} = (x_1, x_2)$ *where* $x_1 =$ sepal width, $x_2 =$ petal width

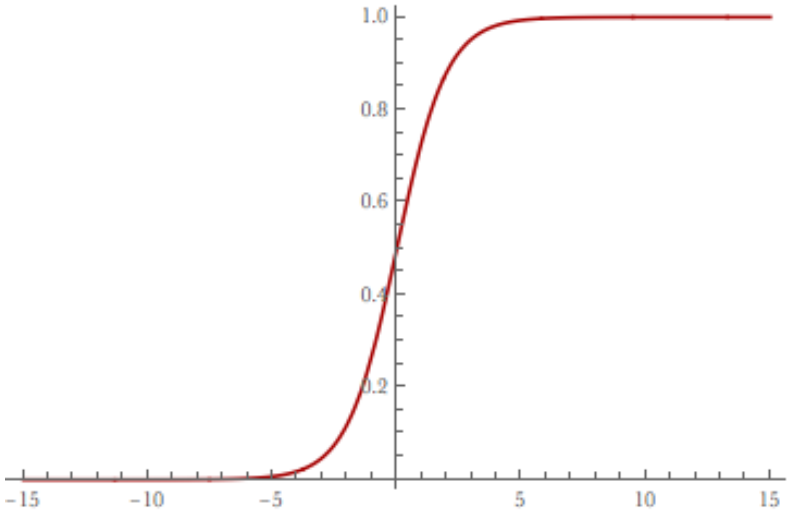
(a) versicolor

(b) virginica

Sigmoid Function

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





In this example, you can think

$$\boldsymbol{\theta} = (\theta_0, \theta_1) \; and \; \mathbf{x} = (1, x)$$

therefore,

$$p(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 \cdot x)}}$$

$$p(x; \boldsymbol{\theta}) \geq \frac{1}{2} \Rightarrow class1$$

$$p(x; \boldsymbol{\theta}) < \frac{1}{2} \Rightarrow class0$$

$$p(\mathbf{x}; \boldsymbol{\theta}) = P(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^\intercal \mathbf{x}}}$$

$$1 - p(\mathbf{x}; \boldsymbol{\theta}) = P(y = 0|\mathbf{x}) = \frac{e^{-\boldsymbol{\theta}^\intercal \mathbf{x}}}{1 + e^{-\boldsymbol{\theta}^\intercal \mathbf{x}}}$$

$$
\begin{aligned}
class1 &\Leftrightarrow p(\mathbf{x};\boldsymbol{\theta}) = \frac{1}{1+e^{-\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}}} > \frac{1}{2} \\
&\Leftrightarrow e^{-\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}} > 1 \\
&\Leftrightarrow \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} > 0
\end{aligned} \tag{3}
$$

$$
\begin{aligned}
class0 &\Leftrightarrow p(\mathbf{x};\boldsymbol{\theta}) = \frac{1}{1+e^{-\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}}} < \frac{1}{2} \\
&\Leftrightarrow e^{-\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}} < 1 \\
&\Leftrightarrow \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} < 0
\end{aligned} \tag{4}
$$

Conclusion

- Logistic regression gives us a **linear classifier**.
- $\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x} = 0$ is **decision boundary**.

3D Visualization

See a Mathematica Dynamic Visualization

Maximum Likelihood

We can assume that

$$P(y = 1|\mathbf{x}) = p(\mathbf{x}; \boldsymbol{\theta})$$

$$P(y = 0|\mathbf{x}) = 1 - p(\mathbf{x}; \boldsymbol{\theta})$$

The likelihood function $\mathcal{L}$, which quantifies how likely output is given input, is defined as follows:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= \prod_{i=1}^{n} P(Y = y_i|X = \mathbf{x}_i; \boldsymbol{\theta}) = \prod_{y_i=0} (1 - p(\mathbf{x}_i; \boldsymbol{\theta})) \prod_{y_i=1} (p(\mathbf{x}_i; \boldsymbol{\theta})) \\ &= \prod_{i=1}^{n} p(\mathbf{x}_i; \boldsymbol{\theta})^{y_i} (1 - p(\mathbf{x}_i; \boldsymbol{\theta}))^{(1-y_i)} \end{aligned} \tag{5}$$

Maximum Likelihood

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \prod_{i=1}^{n} P(Y = y_i | X = \mathbf{x}_i, \boldsymbol{\theta}) = \prod_{y_i=0} (1 - p(\mathbf{x}_i; \boldsymbol{\theta})) \prod_{y_i=1} (p(\mathbf{x}_i; \boldsymbol{\theta})) \\ &= \prod_{i=1}^{n} p(\mathbf{x}_i; \boldsymbol{\theta})^{y_i} (1 - p(\mathbf{x}_i; \boldsymbol{\theta}))^{(1-y_i)}\end{aligned} \tag{6}$$

$$\begin{aligned}\ell(\boldsymbol{\theta}) &= \log(\mathcal{L}(\boldsymbol{\theta})) = \sum_{i=1}^{n} y_i \log(p(\mathbf{x}_i; \boldsymbol{\theta}) + (1 - y_i) \log(1 - p(\mathbf{x}_i; \boldsymbol{\theta})) \\ &= \sum_{y_i=0} \log(1 - p(\mathbf{x}_i; \boldsymbol{\theta})) + \sum_{y_i=1} \log(p(\mathbf{x}_i; \boldsymbol{\theta}))\end{aligned} \tag{7}$$

Since $f(x) = log(x)$ is a monotonically increasing function, maximizing $\mathcal{L}(\boldsymbol{\theta})$ is equivalent as maximizing $\ell(\boldsymbol{\theta})$

Maximum Likelihood Estimation

Set derivatives equal to zero, then solve, we will find the maximum likelihood. Let $p(\mathbf{x}) = \sigma(\boldsymbol{\theta}^\intercal \mathbf{x}) = \frac{1}{1+e^{-\boldsymbol{\theta}^\intercal \mathbf{x}}}$

$$
\begin{aligned}
\ell'(\boldsymbol{\theta}) = \frac{\partial \ell}{\partial \boldsymbol{\theta}} &= \left(y \frac{1}{\sigma(\boldsymbol{\theta}^\intercal \mathbf{x})} - (1-y) \frac{1}{1-\sigma(\boldsymbol{\theta}^\intercal \mathbf{x})} \right) \frac{\partial}{\partial \boldsymbol{\theta}_j} \sigma(\boldsymbol{\theta}^\intercal \mathbf{x}) \\
&= \left(y \frac{1}{\sigma(\boldsymbol{\theta}^\intercal \mathbf{x})} - (1-y) \frac{1}{1-\sigma(\boldsymbol{\theta}^\intercal \mathbf{x})} \right) \sigma(\boldsymbol{\theta}^\intercal \mathbf{x})(1-\sigma(\boldsymbol{\theta}^\intercal \mathbf{x})) \frac{\partial}{\partial \boldsymbol{\theta}_j} \boldsymbol{\theta}^\intercal \mathbf{x} \\
&= \left(y(1-\sigma(\boldsymbol{\theta}^\intercal \mathbf{x}) - (1-y)\sigma(\boldsymbol{\theta}^\intercal \mathbf{x}) \right) \mathbf{x}_j \\
&= (y - p(\mathbf{x}))\mathbf{x}_j
\end{aligned}
\tag{8}
$$

Solve for $\boldsymbol{\theta}$ by setting $\ell'(\boldsymbol{\theta}) = 0$

Let's go back to our iris flower and take a look of a concrete example!

Logistic Regression with More Than Two Classes

If Y can take on k values, i.e. we have k classes, we can still use logistic regression. The predicted conditional probabilities will be

$$P(Y = c|X = \mathbf{x}) = \frac{\exp(\boldsymbol{\theta}^{c\,\mathsf{T}}\mathbf{x})}{\sum_j^k \exp(\boldsymbol{\theta}^{j\,\mathsf{T}}\mathbf{x})} \tag{9}$$

This is commonly known as Softmax function.

More topics about logistic regression..

The Math of March Madness

SundayReview

The Math of March Madness

Gray Matter
By JORDAN ELLENBERG MARCH 20, 2015

Olimpia Zagnoli

THE N.C.A.A. men's basketball tournament started Thursday, but for most Americans the real action began days before, as they pored over brackets, competing to make the most accurate predictions — for money, or just office glory. These days, when statistical algorithms can figure out what breakfast cereal you want based on your browser history, stats-minded hoops fans have thrown lots of complex analysis at the problem of picking winners.

But what's the best method? Last year a company called Kaggle, which offers predictive modeling services, held a competition in which more than 200 teams of data scientists threw the best statistics they could at the problem, in pursuit of a $15,000 top prize. It

The Math of March Madness

- *"Professors Lopez and Matthews didn't use any of the au courant methods in data science circles, either: no deep learning, no hierarchical clustering, no compressed sensing; just a good old model called* **logistic regression**, *which turns a number (like a point spread) into an estimated probability that team A will beat team B."*

The Math of March Madness

- Just 2 data sources used - the Las Vegas point spreads for the N.C.A.A. match-ups and a set of offensive and defensive efficiency ratings.
- Shows that if the data is simple, then logistic regression can be an efficient algorithm - compared to more complex methods which may easily overfit.

Relation with Cross-entropy Error Measure

- For two probability distributions $\{p, 1-p\}$ and $\{q, 1-q\}$ with binary outcomes, the *cross-entropy* (from information theory)

$$\begin{aligned} &= p \log \frac{1}{q} + (1-p) \log \frac{1}{1-q} \\ &= -p \log q - (1-p) \log(1-q) \end{aligned} \tag{10}$$

- Assuming that p is the "true" distribution and q is the approximate distribution for p, minimizing the cross-entropy is equivalent to minimizing the distance (KL-divergence) between the 2 distributions, i.e. make the approximate distribution as close to the "true" distribution as possible.

Relation with Cross-entropy Error Measure

- For the i^{th} training data,

$$\begin{aligned} p_i &= y_i \\ q_i &= \sigma(\theta^T \mathbf{x}_i) \end{aligned} \tag{11}$$

- The negative log likelihood for the i^{th} training data point may be expressed as

$$-\log P(Y = y_i | X = \mathbf{x_i}) = -y_i \log \sigma(\theta^T \mathbf{x}_i) - (1 - y_i) \log(1 - \sigma(\theta^T \mathbf{x}_i)) \tag{12}$$

- Same as cross-entropy measure.

Relation with Cross-entropy Error Measure

- Total cross-entropy error measure over all data points is equal to the negative log likelihood for observation Y given training data set $\mathbf{x}$ and θ, i.e. the cost function for the logistic regression model.
- Minimizing the cost function w.r.t. θ may thus be also interpreted as minimizing the distance between the "true" distribution (i.e. observation) and the sigmoid approximation.

Regularization

- **Bayesian view:** If the parameters are assumed to have a prior distribution, then we need to perform *maximum a posteriori* (MAP) estimation.
- Recall, Bayes' theorem:

$$posterior \propto likelihood \times prior \tag{13}$$

- If the parameters are assumed to have a prior distribution, then we need to perform *maximum a posteriori* (MAP) estimation.

Regularization

- Let $P(\theta) = \exp(-\frac{||\theta||^2}{2\sigma^2})$. Then

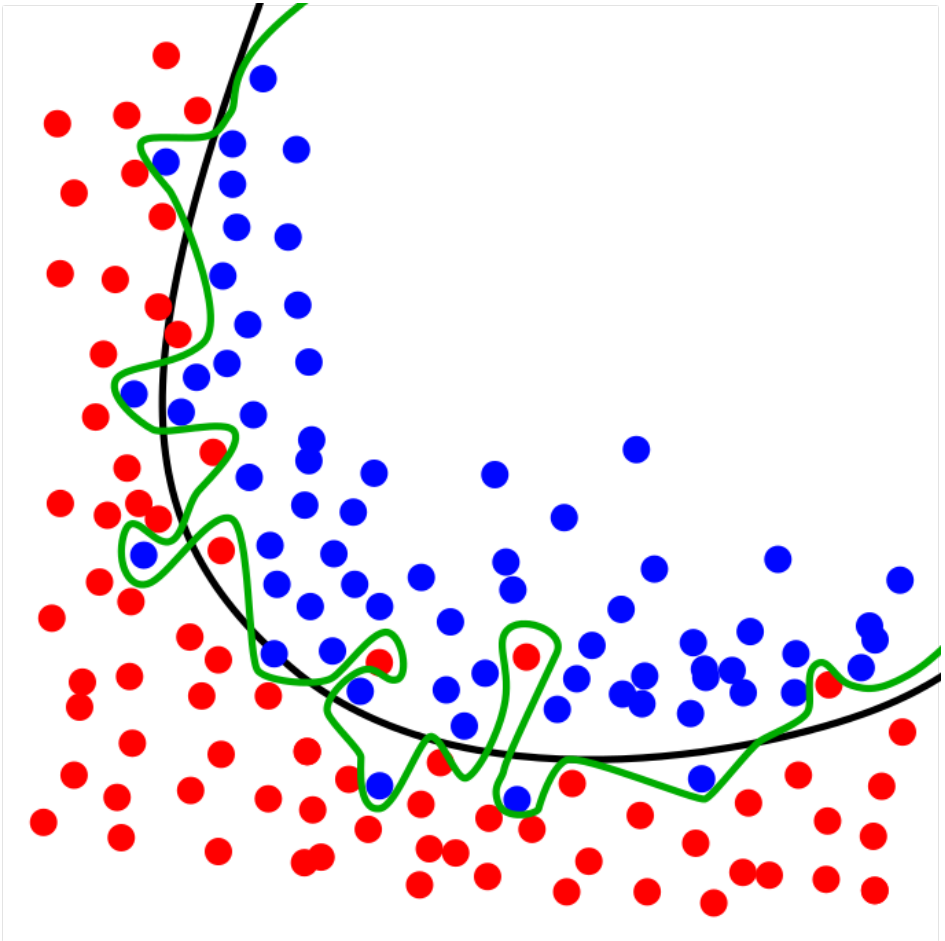
$$\begin{aligned}
\theta^* &= \text{argmax}(\prod_{i=1}^{n} P(y_i|\mathbf{x_i}, \theta)P(\theta)) \\
&= \text{argmin}(-\log(\prod_{i=1}^{n} P(y_i|\mathbf{x_i}, \theta)P(\theta))) \\
&= \text{argmin}(-\log(\prod_{i=1}^{n} P(y_i|\mathbf{x_i}, \theta) - \log P(\theta)) \\
&= \text{argmin}(-\log(\prod_{i=1}^{n} P(y_i|\mathbf{x_i}, \theta) + \frac{||\theta||^2}{2\sigma^2})
\end{aligned} \tag{14}$$

Regularization

- Replacing $\lambda = \frac{1}{2\sigma^2}$, the MAP estimator minimizes the negative log-likelihood added with an L2 regularization term.
- Adding a regularization term prevents overfitting by increasing training error but reducing generalization error. λ controls the extent of regularization.
- Reduces model complexity by enforcing constraints on parameters.
- Varying the prior function leads to different types of regularization, e.g. a Laplacian prior produces L1 regularization.

Regularization

- Illustration:



Regularization

- $$\min \quad J(\theta) + \lambda_C \theta^T \theta \tag{15}$$

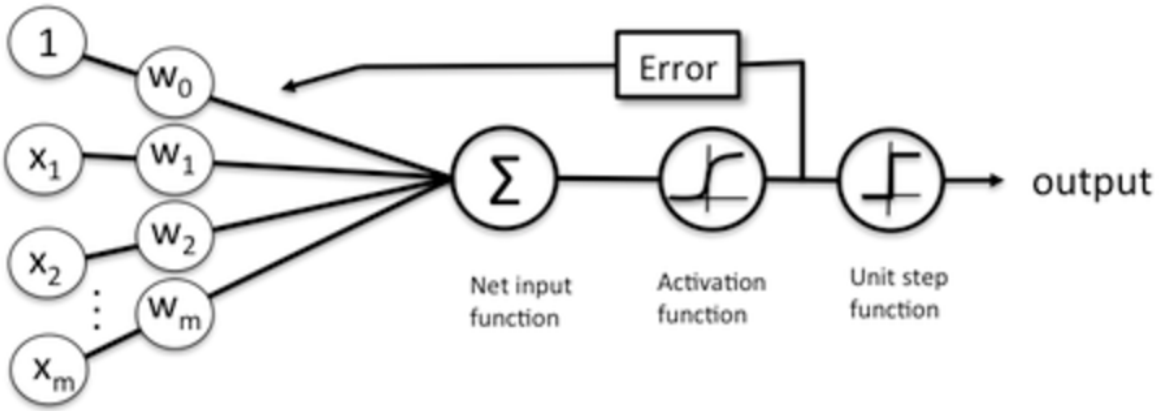
  where λ_C is the Lagrangian multiplier.
- Lagrangian form of constrained optimization problem:

$$\min \quad J(\theta) \tag{16}$$

$$\text{s.t.} \quad \theta^T \theta \leq C \tag{17}$$

Logistic Regression and Neural Networks

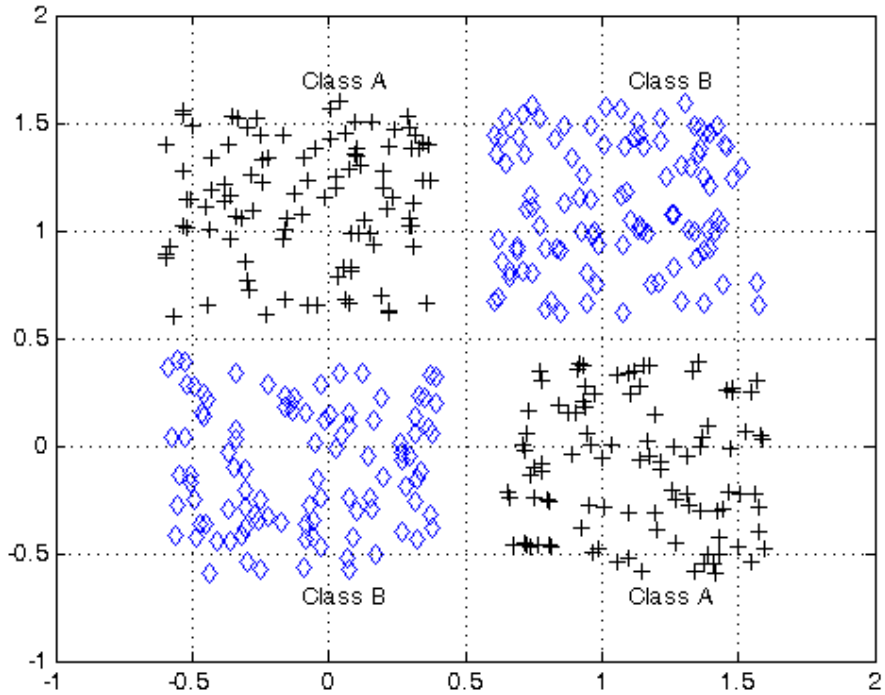
- Logistic regression may be thought of as a single-layer neural network with a sigmoid activation function.



Schematic of a logistic regression classifier.

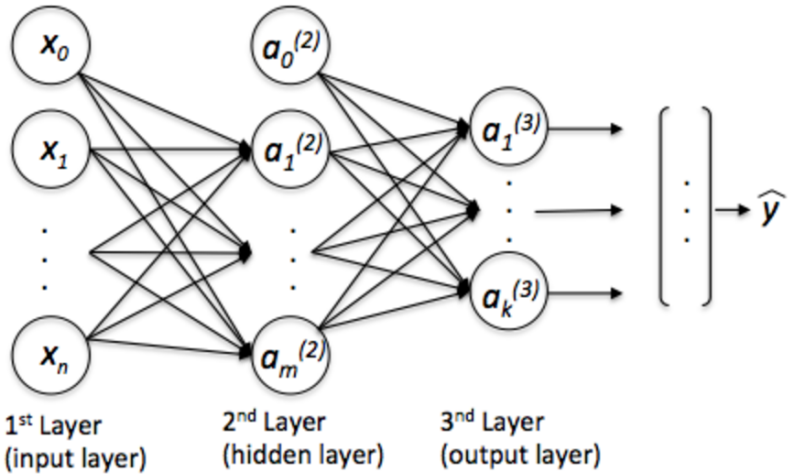
Logistic Regression and Neural Networks

- Neural networks may be visualized as a combination of linear classifiers for complex boundaries.
- Example: XOR function



Logistic Regression and Neural Networks

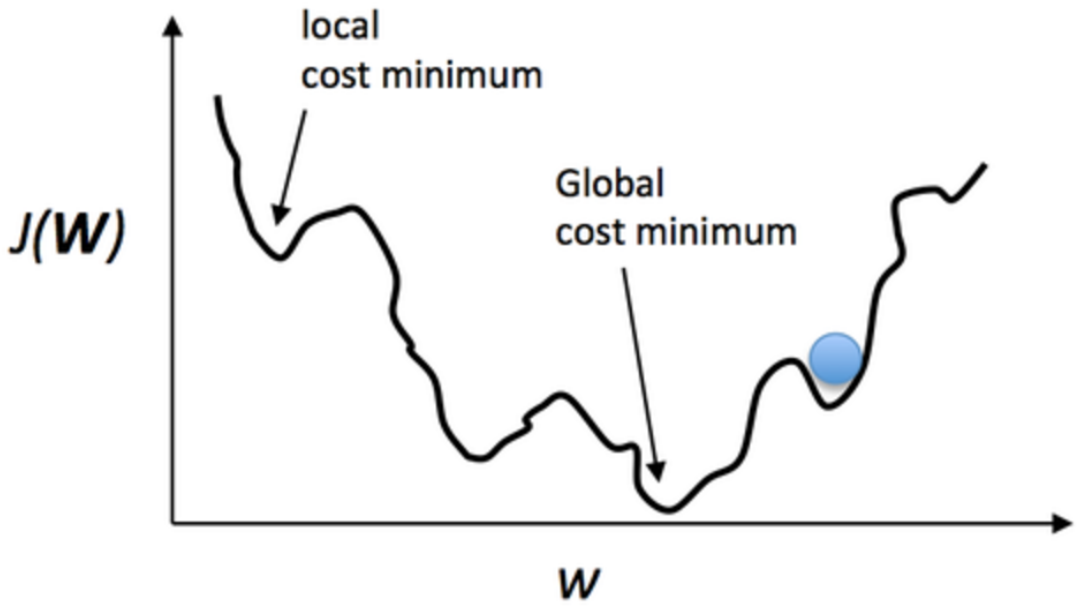
- Sigmoid activation function commonly used previously in hidden layers of neural networks along with *softmax* (multinomial logistic regression) at the final fully connected output layer.
- Nowadays, ReLU (**Re**ctified **L**inear **U**nit) activation function is most popular, but softmax is still commonly used at the final layer.



Schematic of a multi-layer perceptron.

Logistic Regression and Neural Networks

- Logistic regression cost function is convex, but cost function of neural networks is non-convex due to combination of multiple logistic regression layers.



Section 3: Optimization Methods

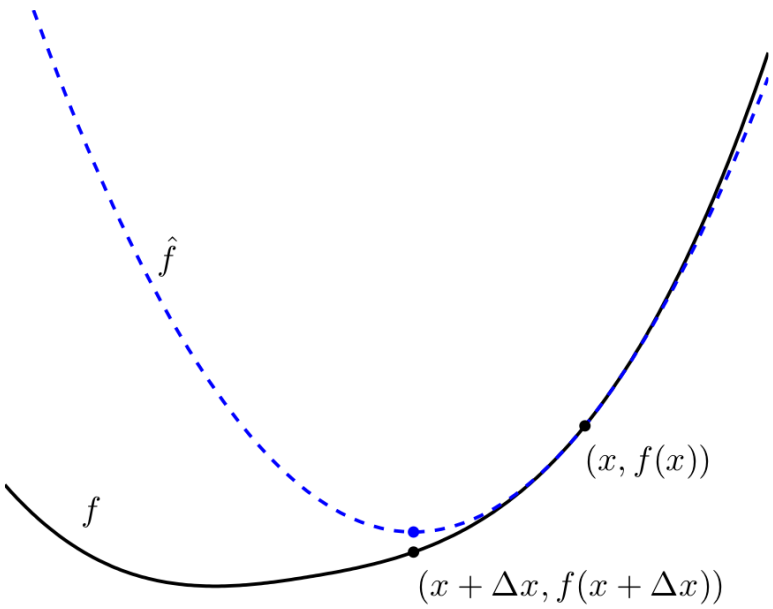
What Optimization Methods to Use?

- Cost function of Logistic Regression is convex

Newton's Method for Optimization

- Newton's Method in 1-D is used as an approximation tool to find roots of differentiable functions
  - Second order Taylor Expansion $\ell^T(\theta)$ of ℓ around θ_n

$$\theta_{n+1} = \theta_n - \frac{\ell'(\theta)}{\ell''(\theta)} \tag{18}$$



$\hat{f}$ is 2^{nd}-order approximation, f is true function.

Newton's Method in Higher Dimensions

- When l is a function of multiple arguments

$$\theta_{n+1} = \theta_n - H^{-1}(\theta_n)\delta\ell(\theta_n)$$

- Calculating the inverse of H is very time consuming

- Pros
  - Converge quadratically
  - Finds minima instead of general critical points
- Cons
  - Newton's Method uses second partial derivatives, which may be complex to calculate
  - Hessian matrix requires $O(n^3)$ operations to solve and $O(n^2)$ memory to store
  - Newton's Method may diverge away from the root if the initial point is chosen poorly

Newton's Method for Optimization

- Because of the problems that Newton's Method has, We used Gradient Descent to minimize our cost function

Gradient Descent

- Having an estimate of $\delta\ell$, gradient of a function ℓ
- Decrease ℓ by a quantity along the direction of $\delta\ell$
  - Direction
  - Step size
- https://www.youtube.com/watch?v=GCvWD9zIF-s

Quasi-Newton Method

- Used when the problem size is big and the Hessian matrix is dense
- Keeps "a rolling estimate" of $H(x)$ to solve this problem

Newton's Method as Iteratively Re-weighted Least Squares

- Recall, logistic regression model is a Bernoulli distribution with parameter $\sigma(\theta^T \mathbf{x})$:

$$\begin{aligned} P(Y = y|X = \mathbf{x}, \theta) &= \sigma(\theta^T \mathbf{x})^y (1 - \sigma(\theta^T \mathbf{x}))^{1-y} \\ &= \text{Ber}(y; \sigma(\theta^T \mathbf{x})) \end{aligned} \tag{19}$$

- Again, logistic regression may be formulated as linear regression that best fits the *log-odds*:

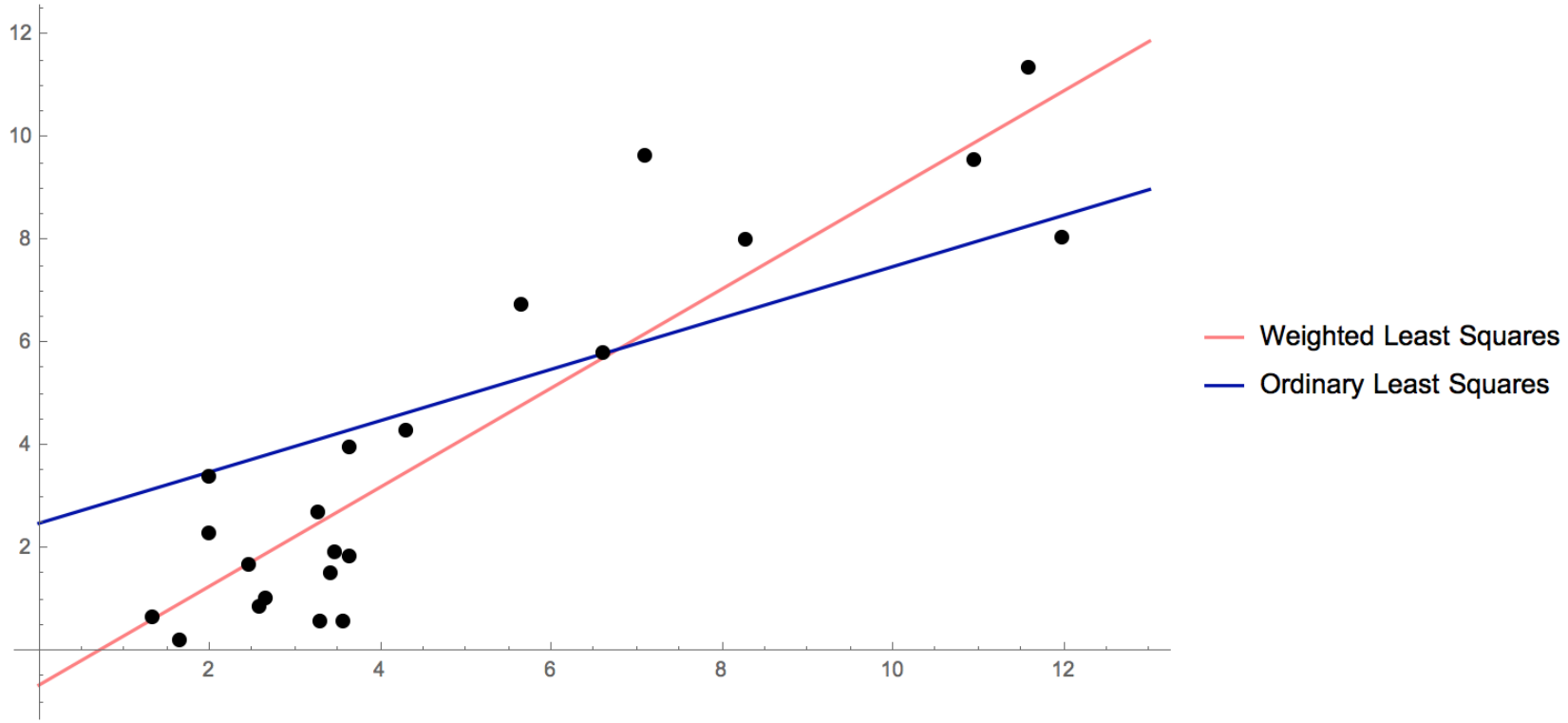
$$\begin{aligned} \log \frac{P(Y = 1|X = \mathbf{x}, \theta)}{P(Y = 0|X = \mathbf{x}, \theta)} &= \log \frac{\sigma(\theta^T \mathbf{x})}{1 - \sigma(\theta^T \mathbf{x})} \\ &= \theta^T \mathbf{x} \end{aligned} \tag{20}$$

Newton's Method as Iteratively Re-weighted Least Squares

- Linear regression is heavily affected by outliers - all observation variables may not have same precision of measurement (or *variance*).
- *Weighted least squares* associates amount of *uncertainty* at every point as a weight - large weights on points with *high precision* or *low variance* and vice-versa.
- **Idea:** Multiply squared distance at each observation point by inverse of variance.

Newton's Method as Iteratively Re-weighted Least Squares

- Illustration:



Newton's Method as Iteratively Re-weighted Least Squares

- Loss function for weighted least squares regression (without regularization):

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \frac{1}{\text{Var}(y_i)} (y_i - \theta^T x_i)^2 \tag{21}$$

- Least squares solution ($\text{Var}(y_i) = \text{const.}$):

$$\theta^* = (X^T X)^{-1} X^T Y \tag{22}$$

where $X = [x_1, ..., x_n]^T$, $Y = [y_1, ..., y_n]^T$.

Newton's Method as Iteratively Re-weighted Least Squares

- *Weighted* least squares solution:

$$\theta^* = (X^T W X)^{-1} X^T W Y \tag{23}$$

with $n \times n$ diagonal matrix $W_{ii} = \dfrac{1}{\text{Var}(y_i)}$

- Variance of a Bernoulli distribution with parameter p is $p(1-p)$. Hence, for logistic regression model,

$$W_{ii} = \frac{1}{\sigma(\theta^T x_i)(1 - \sigma(\theta^T x_i))} \tag{24}$$

Newton's Method as Iteratively Re-weighted Least Squares

- It can be shown that *in expectation*, a step update by Newton's method on the logistic regression objective function is equivalent to reaching the weighted least-squares solution.
- **Key reason**: Expected second derivative of the Bernoulli log-likelihood function is inversely proportional to the variance of the Bernoulli distribution (proof not shown).
- However, not a single-step solution, as the weights themselves are functions of θ and the objective function is not quadratic.

Section 4: Generalized Linear Models and Generalized Additive Models

Generalized Linear Models (GLM)

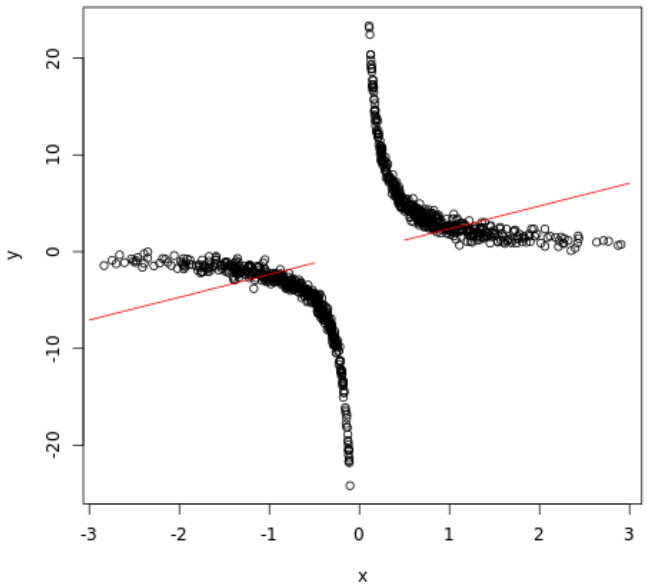
- Generalized Linear Models are models in which conditional distribution of the response falls in some parametric family (exponential family), and the parameters are set by the linear predictor.
- Link function determines the relationship between the parameters and the linear predictor.
- Usually $g(E(\mathbf{Y})) = g(\mu) = X\theta_1 + \theta_0$, where g is the link function.

Generalized Linear Models (GLM)

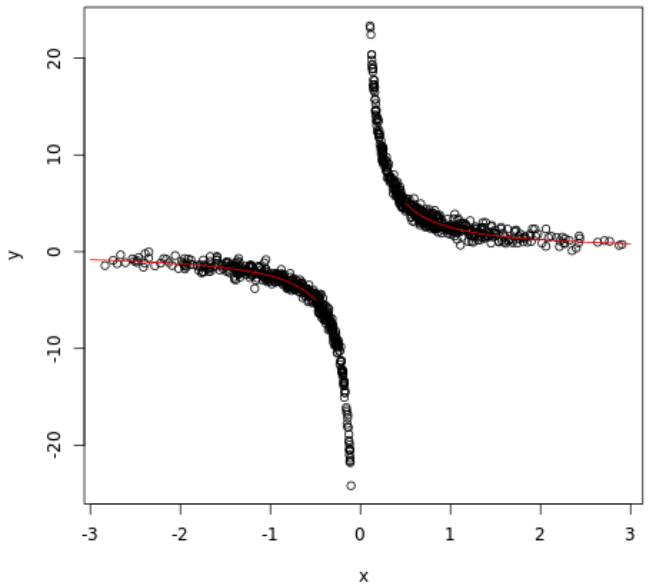
- Both least-squares regression and logistic regression are part of generalized linear models.
- Least-squares regression assumes gaussian distribution with mean predicted by the linear predictor. (identity link)
- Logistic regression assumes binomial distribution, with n equal to the number of data points with given x, and p given by the logistic function. $(p = \frac{1}{1 + e^{-(\theta_0 + x \cdot \theta_1)}})$

GLM vs LM

Data is generated with the the equation $y \sim \mathcal{N}(\mu, 0.5)$, $\mu = \frac{1}{0.4x}$.



(a) Line fitted with least regression



(b) Line fitted with GLM

Generalized Additive Models (GAM)

- Unlike Generalized Linear Models, instead of making the transformed mean a linear function of the inputs, we can make it an additive function of the inputs
- $g(E(\mathbf{Y})) = g(\mu) = \alpha + f_1(x_1) + f_2(x_2) + ... + f_n(x_n)$

Model Checking

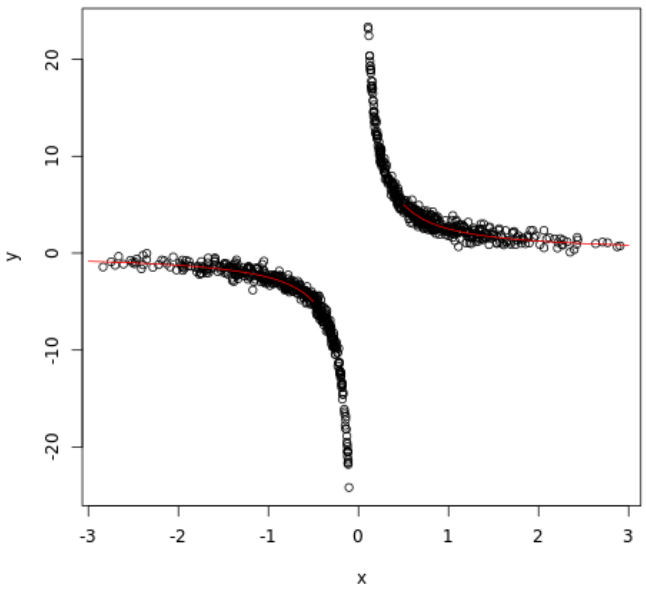
- When our GLM is wrong, then no matter how much data we use, the prediction of our model is systematically wrong. However, since GAM can be nonparametric, with enough data, GAM will fit better than GLM. We can therefore use this property to design a procedure that detect if our GLM is wrong.

Model Checking

MSE: Mean Squared Error
MSE_p: Parametric mean squared error
MSE_{np}: Nonparametric

1. Fit a GAM and a GLM to the same data. Compute MSE_p and MSE_{np}. Compute $t = MSE_p - MSE_{np}$.
2. Simulate from fitted GLM, refit both models to the simulated data. Repeat many times. compute $\hat{MSE}_p$ and $\hat{MSE}_{np}$. Compute $\hat{t} = \hat{MSE}_p - \hat{MSE}_{np}$.
3. p-value: $\dfrac{1 + \#\{\hat{t} > t\}}{1 + \#\hat{t}}$

Model Checking: Example



- By performing model checking on our glm model with inverse link function, we get p-value of 0.72. This is reassuring, since the data is indeed generated from the inverse of a linear function.

Section 5: Applications

Section 5: Applications

Quick Review

What we get so far...

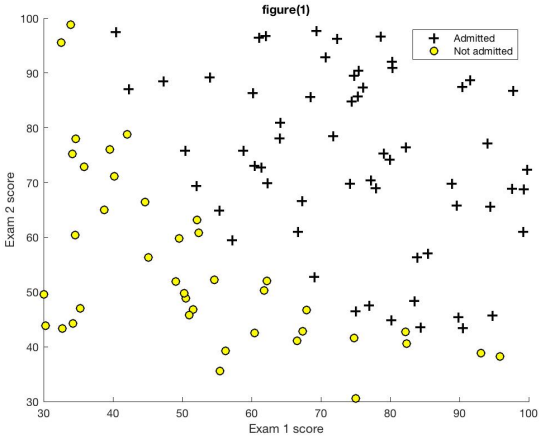
Logistic Regression hypothesis: $h_\theta(x) = \dfrac{1}{1 + e^{-\theta^T x}}$

Optimization: Gradient Descent / Newton Method

Regularization

Logistic Regression as Binary Classifier

Problem: Predict whether a student gets admitted into a university, given two historical exams data.



Labels: y=1: Admitted / y=0: Not admitted

Logistic model: $h_\theta(x) = \dfrac{1}{1 + e^{-\theta^T x}}$

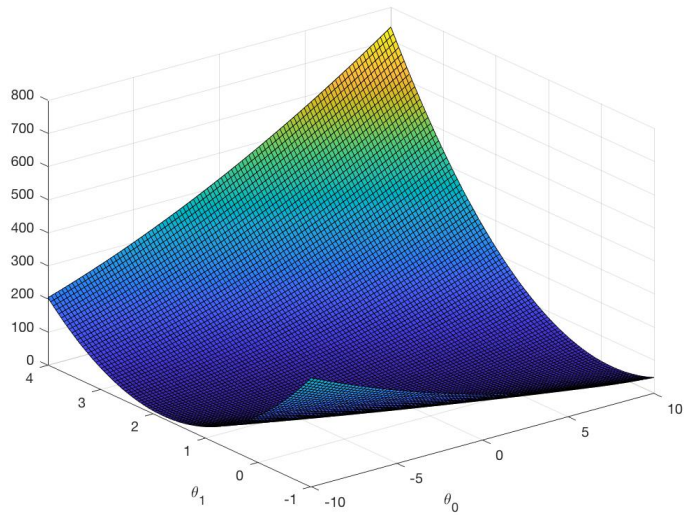
Logistic Regression as Binary Classifier

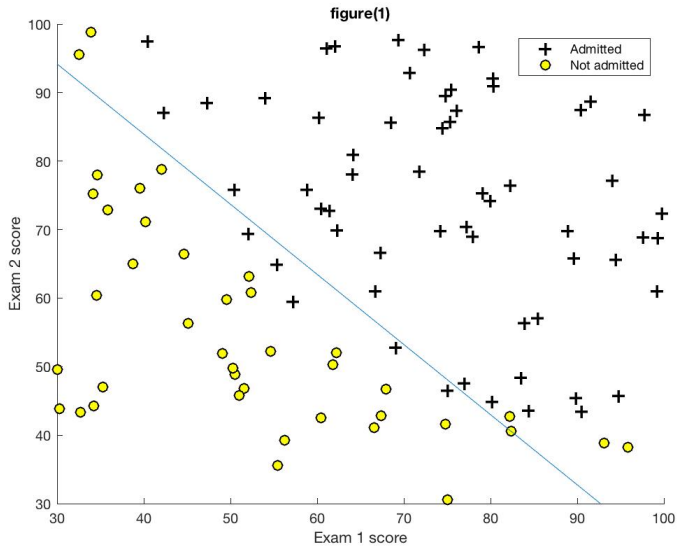
Cost function without regularization:

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}[-y^{(i)} * log(h_\theta(x^{(i)})) - (1 - y^{(i)}) * log(1 - h_\theta(x^{(i)})]$$

Optimized by Gradient Descent

Training Accuracy: 89%





Logistic Regression as Binary Classifier with Regularization

Problem: Predict whether a product of Microchip Manufacturing is malfunction, given 2 kind of tests score. Labels: y=1 pass / y=0 fail.

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

Cost function with regularization:

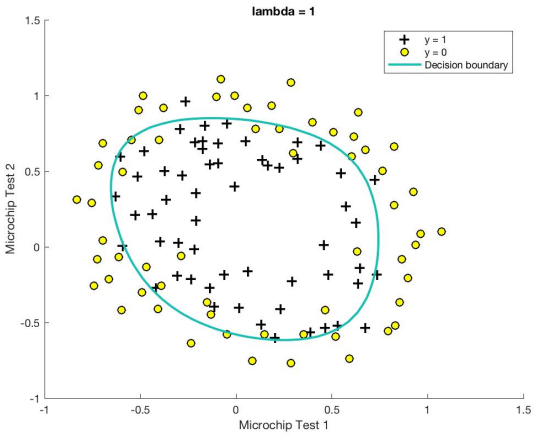
$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}[-y^{(i)} * log(h_\theta(x^{(i)})) - (1-y^{(i)}) * log(1-h_\theta(x^{(i)})] + \frac{\lambda}{2n}\sum_{j=1}^{d}\theta_j^2$$

Feature Transformation: $Z(x) = [1, x_1, x_2, x_1^2, x_2^2, x_1x_2, ..., x_1x_2^5, x_2^6]^T$

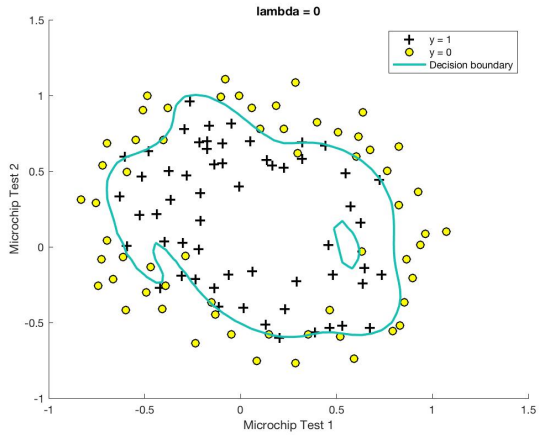
Optimized by Gradient Descent

Logistic Regression as Binary Classifier with Regularization

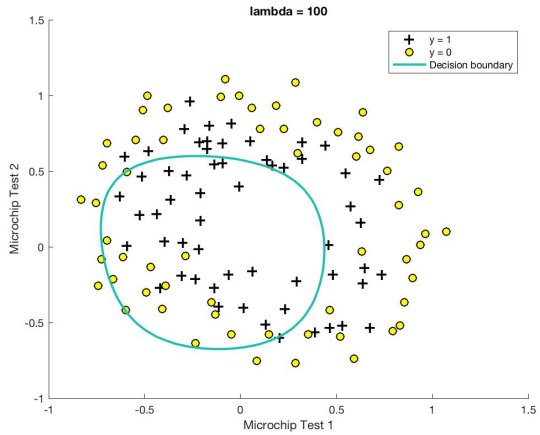
Training Accuracy:



(a) $\lambda = 1$:
accuracy = 83%
Good fitting



(b) $\lambda = 0$:
accuracy = 87%
Over-fitting



(c) $\lambda = 100$:
accuracy = 61%
Under-fitting

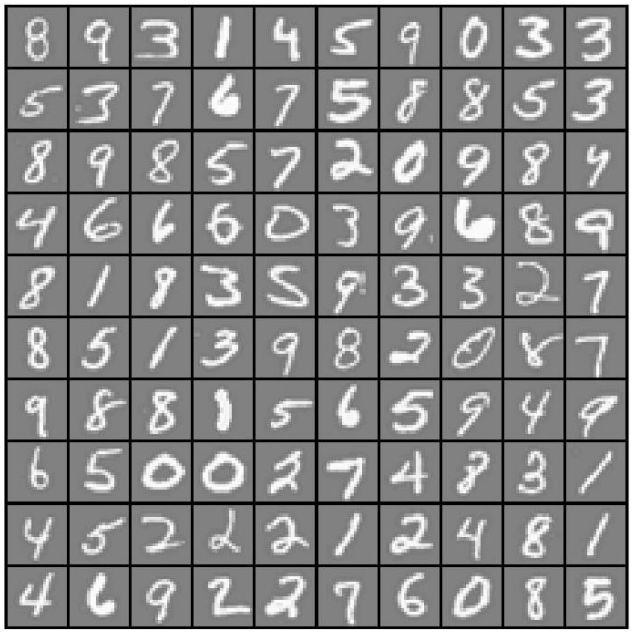
Handwritten Digit Recognition

Problem: Recognize handwritten digits (from 0-9)

Dataset: 5000 20 pixel by 20 pixel greyscale images, 5000 labels

X: 5,000 $\times$400 matrix

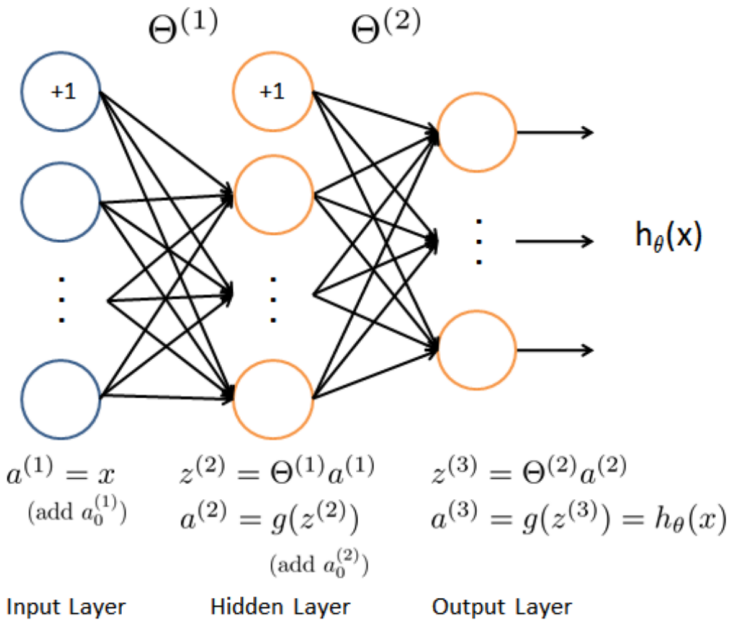
Y: 5,000 $\times$1 vector $\in 0, 1, ..., 9$



Handwritten Digit Recognition - Model

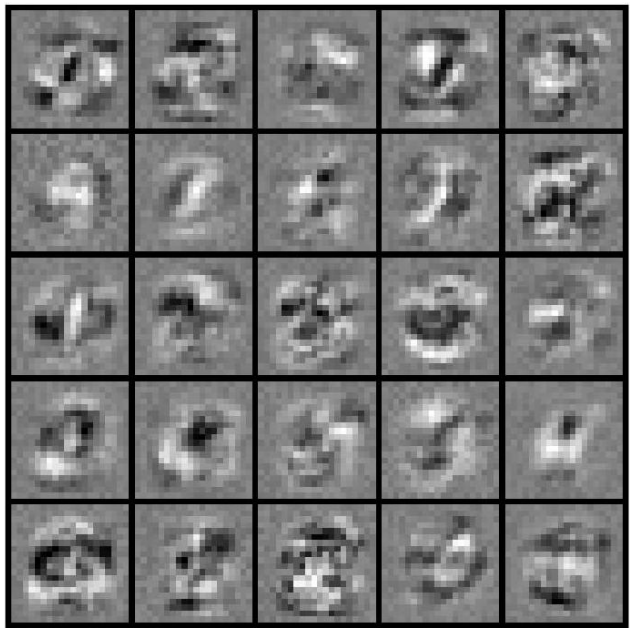
Model:

- Logistic Regression hypothesis: $h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$
- One vs All multi-class classification: $h_\theta^{(c)} = P(y = c|x; \theta), c = 1, ..., 10$
- Prediction: $\max_c h_\theta^{(c)}(x)$
- Neural Network (one hidden layer, 25 units), Optimization: Back-propagation with Gradient Descent



Handwritten Digit Recognition - Result

- Hidden layer units visualization



- Training Accuracy:
  97.52% without regularization, $\lambda = 0$
  95.36% with regularization, $\lambda = 1$

References

- *Advanced Data Analysis from an Elementary Point of View* by C. Shalizi.
- *Learning from Data* by Mostafa, Magdon-Ismail, Lin.
- *Elements of Statistical Learning* by Hastie, Tibshirani.
- *Pattern Recognition and Machine Learning* by Christopher M. Bishop.
- *A comparison of numerical optimizers for logistic regression* by Thomas P. Minka, October 22, 2003
- CS 229 Machine Learning (Stanford) course lectures by Prof. Andrew Ng.
- CSE 517A Machine Learning (WUSTL) course lectures by Prof. Marion Neumann.

Thank You!

Q&A