Supervised Learning

Linear Regression

Ira A. Fulton Schools of Engineering
Arizona State University

Objective

Objective

Define the set-up of Supervised Learning

Objective

Discuss basic regression models

Supervised Learning

- **The set-up: the given training data consist of <sample, label> pairs, or (x, y); the objective of learning is to figure out a way to predict label y for any new sample x.**

- **Consider two types of problems:**
  - **Regression**: y continuous
  - **Classification**: y is discrete, e.g., class labels.

The Task of Regression

- **Given: A training set of *n* samples <$x^{(i)}$, $y^{(i)}$ > where $y^{(i)}$ is a continuous "label" (or target value) for $x^{(i)}$**

- **To learn a model for predicting y for any new sample x.**

- **A simple model is linear regression: modeling the relation between y and x via a linear function.**

$$y \approx w_0 + w_1 x_1 + \ ... + w_d x_d = \mathbf{w}^t\mathbf{x}$$

(Note: **x** is *augmented* by adding a dimension of constant 1 to the original sample.)

Linear Regression

- We can introduce an error term to capture the residual $\mathbf{y} = \mathbf{w}^t\mathbf{x} + e$

- Applying this to all *n* samples, we have: $\mathbf{y} = \mathbf{X}\,\mathbf{w} + \mathbf{e}$

$$\begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{pmatrix} \qquad \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} & \cdots & x_d^{(n)} \end{bmatrix} \qquad \begin{pmatrix} e^{(1)} \\ e^{(2)} \\ \vdots \\ e^{(n)} \end{pmatrix}$$

- *Learning* in this case is to figure out a good w.

Linear Regression (cont'd)

- Find an optimal w by minimizing the squared error

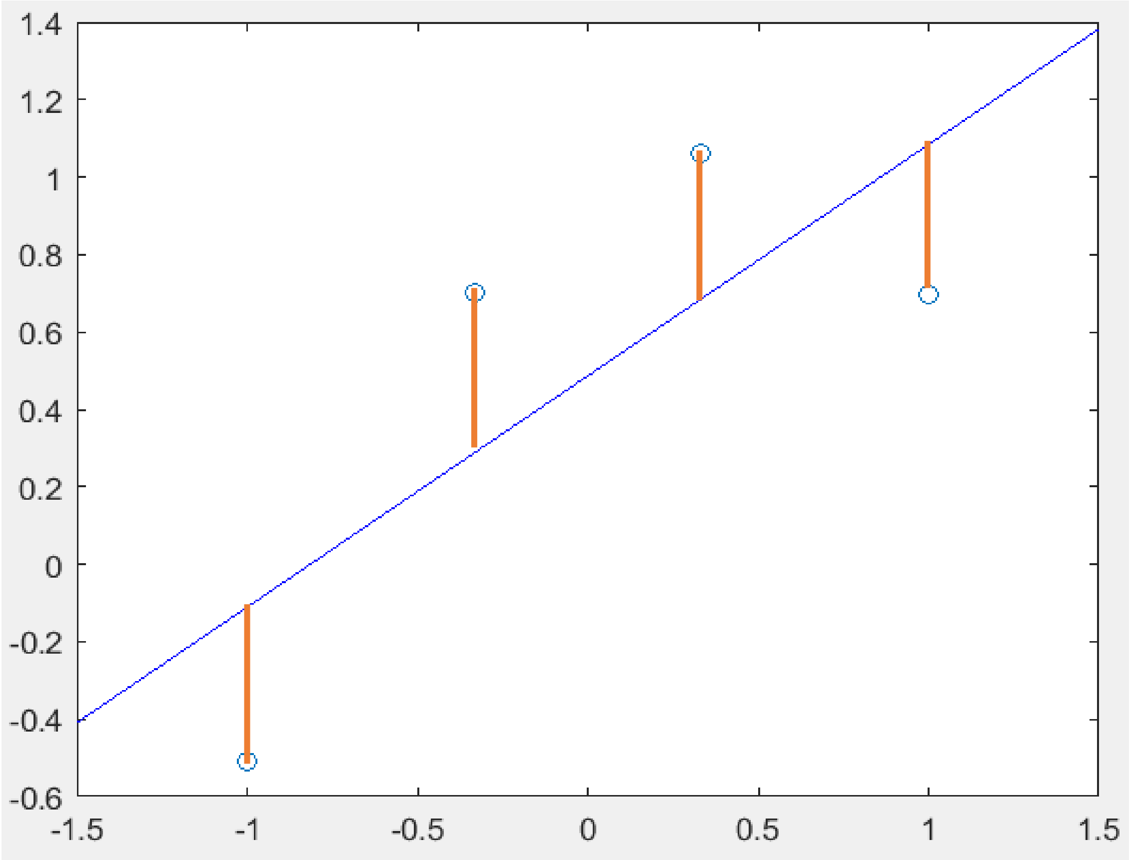
$$||\mathrm{e}||^2 = ||\mathrm{y} - X\,\mathrm{w}||^2$$

- The solution can be found to be:

$$\hat{\mathrm{w}} = (X^t X)^{-1} X^t \mathrm{y}$$

- In practice, some iterative approaches may be used (e.g., gradient descent search).

A simple example



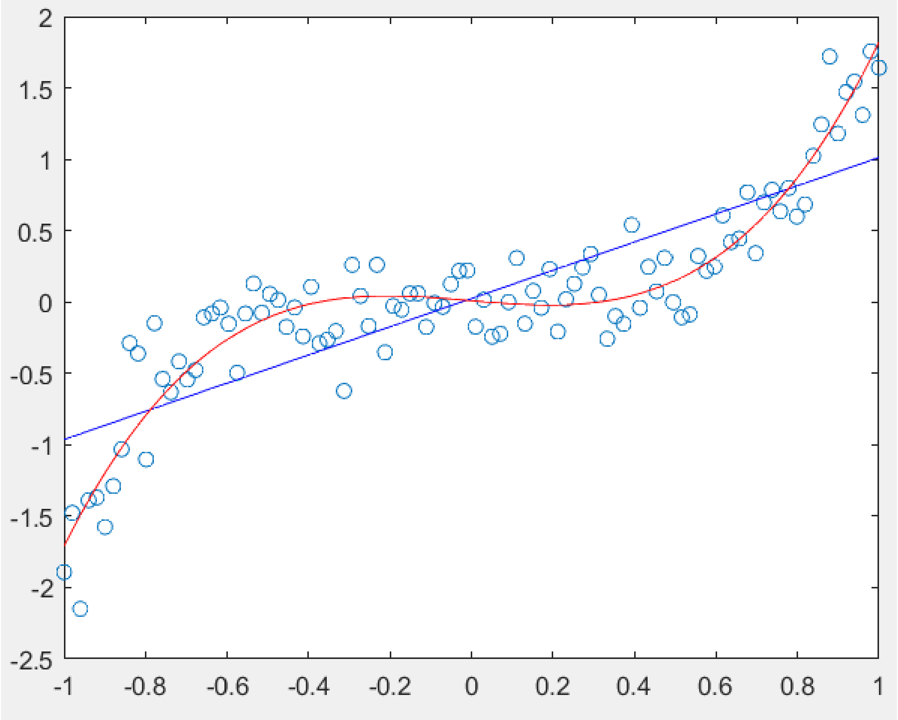
Generalizing Linear Regression

Introducing some basis functions $\phi_j(\mathbf{x})$:

$$y = w_0 + w_1\phi_1(\mathbf{x}) + \ \ldots + w_{M-1}\phi_{M-1}(\mathbf{x})$$

Compare:

- Blue: Linear Regression
- Red: With $\phi_j(x) = x^j$



Regularized Least Squares

- **E.g., use a new error function:** $E_D(\boldsymbol{w}) + \lambda E_W(\boldsymbol{w})$
  - λ is the regularization coefficient
  - $E_D(\mathbf{w})$ is the data-dependent error
  - $E_{\mathbf{W}}(\mathbf{w})$ is the $regularization\ term$, e.g., $\boldsymbol{E}_{\mathbf{W}}(\mathrm{w}) = ||\mathbf{w}||^{\boldsymbol{q}}$

- **Help to alleviate overfitting.**

Supervised Learning

Density Estimation in Supervised Learning

Ira A. Fulton Schools of Engineering
Arizona State University

Objective

Objective

Illustrate classification in Supervised Learning

Objective

Discuss basic density estimation techniques

Supervised Learning

- **The set-up: the given training data consist of <sample, label> pairs, or (x, y); the objective of learning is to figure out a way to predict label y for any new sample x.**

- **Consider two types of problems:**
  - **Regression**: y continuous
  - **Classification**: y is discrete, e.g., class labels.

Examples of Image Classification

- The MNIST training images of hand-written digits

- The Extended Yale B Face Images

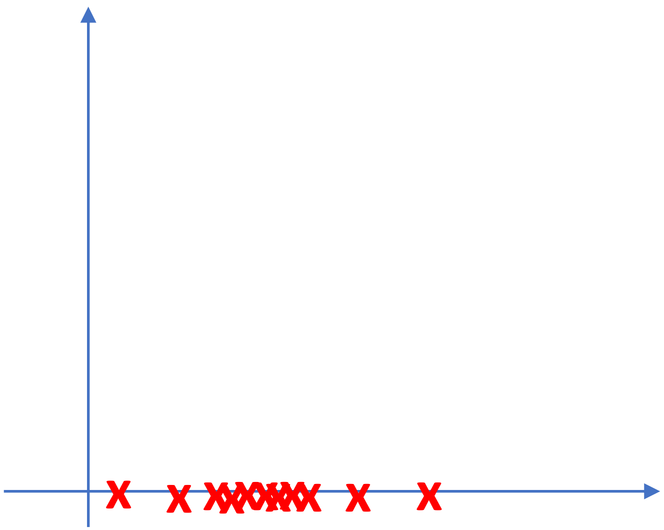
How do we model the training images?

- **Parametric: each class of images (the feature vectors) may be modeled by a density function $p_{\boldsymbol{\theta}}(x)$ with parameter $\boldsymbol{\theta}$.**
  - To emphasize the density is for images from class/label *y*, we may write $p_{\boldsymbol{\theta}}(\mathbf{x}|y)$.
  - We may also use the notation $p(\mathbf{x}|\boldsymbol{\theta})$, if the discussion is true for any y.

  ➔ How to estimate $\boldsymbol{\theta}$ from the training images?

- **Note: We may also consider non-parametric approaches.**

MLE for Density Estimation (1/3)



- **Given some training data; Assuming a parametric model *p*(x|θ); What specific θ will fit/explain the data best?**
  - E.g., Consider a simple 1-D normal density with only a parameter μ (assuming the variance is known)
- **Given a sample x_i, $p(x_i \mid \mu)$ gives an indication of how likely x_i is from $p(x_i \mid \mu)$**

➔the concept of the likelihood function.

MLE for Density Estimation (2/3)

- **The likelihood function: the density function $p(\mathrm{x}|\theta)$ evaluated at the given data sample x_i, and viewed as a function of the parameter θ.**
  - Assessing how likely the parameter $\boldsymbol{\theta}$ (defining the corresponding $p(\mathbf{x}|\boldsymbol{\theta})$) gives arise to the sample $\mathbf{x}_i$.
  - We often use $L(\boldsymbol{\theta})$ to denote the likelihood function, and $l(\boldsymbol{\theta}) = \log(L(\boldsymbol{\theta}))$ is called the log-likelihood.
- **Maximum Likelihood Estimation (MLE): Finding the parameter that maximizes the likelihood function**

$$\widehat{\boldsymbol{\theta}} = \mathrm{argmax}_{\boldsymbol{\theta}}\, p(\mathbf{x}|\boldsymbol{\theta})$$

MLE for Density Estimation (3/3)

- How to consider *all* the given samples $D=\{x_i, i=1,\ldots,n\}$?

- The concept of i.i.d. samples: the samples are assumed to be *independent* and *identically distributed*

- So, the data likelihood is given by

$$L(\boldsymbol{\theta}) = P(D|\boldsymbol{\theta}) =$$

MLE Example 1

- **Tossing a coin for *n* times, observing n_1 times for head.**
  - Estimate the probability θ for head
- **The likelihood function is:**

$$L(\theta) = P(D|\theta) = \theta^{n_1}(1-\theta)^{n-n_1}$$

MLE Example 1 (cont'd)

- We want to find what θ maximizes this likelihood, or equivalently, the log-likelihood

$$l(\theta) = \log P(D|\theta) = \log(\theta^{n_1}(1-\theta)^{n-n_1})$$
$$= \cdots$$

- Take the derivative and set to 0:

$$\frac{d}{d\theta} l(\theta) = 0$$

- This will give us:

$$\hat{\theta} = \frac{n_1}{n}$$

MLE Example 2

- **Given n i.i.d. samples $\{x_i\}$ from the 1-D normal distribution $N(\mu, \sigma^2)$, find the MLE for μ and σ^2**

- **The likelihood function is:**

$$L(\mu,\sigma) = p(D|\mu,\sigma) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \prod_{i=1}^{n} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- **The log-likelihood is:**

$$\begin{aligned} l(\mu,\sigma) &= \log P(D|\mu,\sigma) \\ &= \log\left(\left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \prod_{i=1}^{n} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}\right) \\ &= -n\log\left(\sigma\sqrt{2\pi}\right) - \sum_{i=1}^{n} \frac{(x_i-\mu)^2}{2\sigma^2} \end{aligned}$$

MLE Example 2 (cont'd)

- **The MLE solution for μ**

$$\hat{\mu} = \text{argmax}_{\mu} l(\mu, \sigma)$$
$$= \text{argmax}_{\mu}\{-n \log\left(\sigma\sqrt{2\pi}\right) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2}\}$$

- **Set the derivative to 0:**

$$\frac{\partial}{\partial \mu} l(\mu, \sigma) = 0$$

- **The solution is:**

$$\hat{\mu} = \frac{\sum_{i=1}^{n} x_i}{n}$$

MLE Example 2 (cont'd)

- The MLE solution for μ

$$\hat{\sigma} = \text{argmax}_{\sigma} l(\mu, \sigma)$$
$$= \text{argmax}_{\sigma}\{-n \log\left(\sigma\sqrt{2\pi}\right) - \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2}\}$$

- Set the derivative to 0:

$$\frac{\partial}{\partial\sigma} l(\mu, \sigma) = 0$$

- The solution is:

$$\widehat{\sigma^2} = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}$$

Supervised Learning

Generative vs Discriminative Models in Supervised Learning

Ira A. Fulton Schools of Engineering
Arizona State University

Objective

Objective

Differentiate between generative and discriminative models of supervised learning

Objective

Discuss challenges in Bayesian learning

Supervised Learning

- **The set-up: the given training data consist of <sample, label> pairs, or (x, y); the objective of learning is to figure out a way to predict label y for any new sample x.**
  - E.g., Given n pairs <$\mathbf{x}^{(i)}$, $y^{(i)}$ >, i=1, …, n; $\mathbf{x}^{(i)}$: i-th sample represented as d-dimensional vectors; $y^{(i)}$: corresponding labels.
- **Equivalently, to find $P(y|x)$**

Two Types of Models

Generative Model

- $P(y|x) \propto P(y)\, p(x|y)$'
- ➔ To learn $P(y)$ and $p(x|y)$.

Discriminative Model

- Directly learn $P(y|x)$
- No assumption made on $p(x|y)$

Two Types of Models

Generative Model

- $P(y|x) \propto P(y)\, p(x|y)$’
- ➔ To learn $P(y)$ and $p(x|y)$.
- ➔ Bayesian learning, Bayes classifiers.
- Example: Naïve Bayes Classifier

Discriminative Model

- Directly learn $P(y|x)$
- No assumption made on $p(x|y)$
- Example: Logistic Regression

Practical Difficulty of Bayesian Learning

Consider doing Bayesian learning without making simplifying assumptions.

- Given n training pairs <$\mathbf{x}^{(i)}$, $y^{(i)}$ >, i=1, …, n. Each $\mathbf{x}^{(i)}$ is d-dimensional.
- We need to learn $P(y)$ and $p(\mathbf{x}|y)$

➔ $p(\mathbf{x}|y)$ can be very difficult to estimate:

➔ Consider a very simple case: binary features, and y is also binary. How many probabilities do we need to estimate?

Supervised Learning

Naïve Bayes Classifier

Ira A. Fulton Schools of Engineering
Arizona State University

Objective

Objective

Implement the fundamental learning algorithm Naive Bayes

Naïve Bayesian Classifier

- **The "naive" *conditional independence* assumption: each feature is (conditionally) independent of every other feature, given the label, i.e., $p(x_i | \{x_j \text{ for any } j \neq i\}, y) = p(x_i | y)$**

- **How does this assumption simplify the problem?**
  - Consider the previous example again: d-dimensional binary features, and y is also binary.
  - How many probabilities do we need to estimate now?

$$p(\mathbf{x} | y) = p(x_1, x_2, \ldots, x_d | y) = \ldots$$

Naïve Bayesian Classifier (cont'd)

- The naïve Bayes classifier: the predicted label is given by

$$\hat{y} = \operatorname*{argmax}_{y} P(y) \prod_{i=1}^{d} p(x_i|y)$$

- "Parameters" of the classifier:
  - $P(y)$
  - $p(x_i|y)$ for all i, y

Naïve Bayesian Classifier (cont'd)

- **E.g., estimating the "parameters" of the classifier**
  - $P(y)$ & $p(x_i|y)$ for all i, y -

for the following familiar example

Discrete Feature Example

- $x = <x_1, x_2, \ldots, x_d>$ where each x_i can take only a finite number of values from $\{v_1, v_2, \ldots, v_m\}$:
- In this case, the "parameters" of the classifier are
  - $P(y)$
  - $P(x_i = v_k | y)$, for all i, k, and y
- Given: A training set of n labelled samples $<x^{(i)}, y^{(i)}>$, $i=1, \ldots, n$

  ➔ How to estimate the model parameters?

Discrete Feature Example (cont'd)

Given: A training set of *n* labelled samples <$x^{(i)}$, $y^{(i)}$ >, *i*=1, …, *n*

➔ How to estimate the model parameters?

$P(y) =$

$P(x_i = v_k | y) =$

These are in fact the MLE solutions for the corresponding parameters.

Supervised Learning

Logistic Regression

Ira A. Fulton Schools of Engineering
Arizona State University

Objective

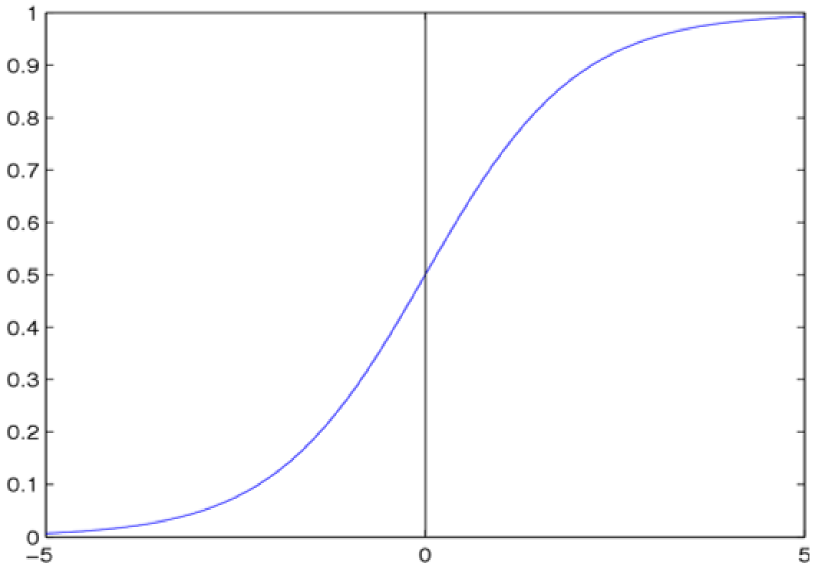
Objective

Implement the fundamental learning algorithm Logistic Regression

Discriminative Model: Example

- **Again, we are given a training set of *n* labelled samples <$x^{(i)}$, $y^{(i)}$ >**
- **Why not directly model/learn *P*(y|x)?**
  - Discriminative model
- **Further assume *P*(y|x) takes the form of a logistic sigmoid function**

→ Logistic Regression



Logistic Regression

- **Logistic regression: use the logistic function for modeling *P*(y|x), considering only the case of y $\in \{0,1\}$**

$$P(y=0|\mathbf{x}) = \frac{1}{1+\exp\left(w_0 + \sum_{i=1}^{d} w_i x_i\right)}$$

$$P(y=1|\mathbf{x}) = \frac{\exp\left(w_0 + \sum_{i=1}^{d} w_i x_i\right)}{1+\exp\left(w_0 + \sum_{i=1}^{d} w_i x_i\right)}$$

- **The *logistic function***

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

Logistic Regression → Linear Classifier

Given a sample x, we classify it as 0 (i.e., predicting y=0) if

$$P(\mathrm{y}=0|\mathbf{x}) \geq P(\mathrm{y}=1|\mathbf{x})$$

➔ This is a linear classifier.

The Parameters of the Model

- What are the model parameters in logistic regression?
- Given a parameter w, we have P(y|x) =

$$[\sigma(w^t x)]^y [1-\sigma(w^t x)]^{1-y}$$

- Suppose we have two different sets of parameters, $w^{(1)}$ and $w^{(2)}$, whichever giving a larger P(y|x) should be a better parameter.

The Conditional Likelihood

- Given n training samples, $<x^{(i)}, y^{(i)}>$, $i=1,\dots,n$, how can we use them to estimate the parameters?

➔ For a given w, the probability of getting all those $y^{(1)}, y^{(2)} \dots, y^{(n)}$ from the corresponding data $x^{(i)}$, $i=1,\dots,n$, is

$$P[y^{(1)}, y^{(2)}, \dots, y^{(n)} \mid x^{(1)}, x^{(2)}, \dots, x^{(n)}; w] = \prod_{i=1}^{n} P(y^{(i)} \mid x^{(i)}; w)$$

$$= \prod_{i=1}^{n} \left[\sigma(w^t x^{(i)})\right]^{y^{(i)}} \left[1 - \sigma(w^t x^{(i)})\right]^{1-y^{(i)}}$$

➔ Call this L(w), the (conditional) likelihood.

The Conditional Log Likelihood

$$\ell(w) = \log \mathcal{L}(w) = \log \prod_{i=1}^{n} (\cdots)$$

$$= \sum_{i=1}^{n} \log \left[\sigma(w^t x^{(i)})^{y^{(i)}} (1 - \sigma(w^t x^{(i)}))^{1-y^{(i)}} \right]$$

$$= \sum_{i=1}^{n} \left[\log\left(\sigma(w^t x^{(i)})^{y^{(i)}}\right) + \log\left((1 - \sigma(w^t x^{(i)}))^{1-y^{(i)}}\right) \right]$$

Maximizing Conditional Log Likelihood

Optimal parameters

$$\mathbf{w}^* = \mathrm{argmax}_{\mathbf{w}} l(\mathbf{w})$$
$$= \mathrm{argmax}_{\mathbf{w}} \sum_{i=1}^{n} [y^{(i)} \mathbf{w}^t \mathbf{x}^{(i)} - \log\left(1 + \exp\left(\mathbf{w}^t \mathbf{x}^{(i)}\right)\right)]$$

We cannot really solve for w* analytically (no closed-form solution)

- We can use a commonly-used optimization technique, gradient descent/ascent, to find a solution.

Finding the Gradient of $l(w)$

Recall: $\frac{\partial (w^t x)}{\partial w} = x$, $\left(\frac{\partial \log f(x)}{\partial x} = \frac{1}{f(x)} \frac{\partial f(x)}{\partial x} \right.$

$\frac{\partial e^x}{\partial x} = e^x$

$$\nabla_w l(w) = \nabla_w \left[\sum_{i=1}^{n} \left(y^{(i)} w^t x^{(i)} - \log\left(1 + e^{w^t x^{(i)}}\right) \right) \right],$$

$$= \sum_{i=1}^{n} \left[y^{(i)} x^{(i)} - \frac{e^{w^t x^{(i)}} \cdot x^{(i)}}{1 + e^{w^t x^{(i)}}} \right]$$

(Setting this to 0 cannot really give us a closed-form solution for w.

So we will do gradient ascent.)

Gradient Ascent Algorithm

<u>The algorithm</u>

Iterate until converge

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \eta \nabla_{\mathbf{w}^{(k)}} l(\mathbf{w})$$

$\eta > 0$ is a constant called the learning rate.