Linear and Logistic regression

CS21206: Foundations of AI and ML

Abir Das & Ayan Chaudhury

IIT Kharagpur

Jan 30, Feb 05, 2026

Agenda

- Understand regression and classification with linear models.
- Understand Gradient Descent and its few variants.
- Using MLE to understand linear regression.
- Using logistic function for binary classification and estimating logistic regression parameters.

Resources

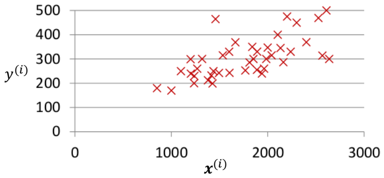
- The Elements of Statistical Learning by T Hastie, R Tibshirani, J Friedman. [Link] [Chapter 3 and 4]
- Artificial Intelligence: A Modern Approach by S Russell and P Norvig. [Link] [Chapter 19]

Linear Regression

- § In a regression problem we want to find the relation between some input variables $\mathbf{x}$ and output variables y, where $\mathbf{x} \in \mathbb{R}^d$ and $y \in \mathbb{R}$.
- § Inputs are also often referred to as covariates, predictors and features; while outputs are known as variates, targets and labels.
- § Examples of such input-output pairs can be
  - ▶ {Outside temperature, People inside classroom, target room temperature | Energy requirement}
  - ▶ {Size, Number of Bedrooms, Number of Floors, Age of the Home | Price}

Linear Regression

- § In a regression problem we want to find the relation between some input variables $\mathbf{x}$ and output variables y, where $\mathbf{x} \in \mathbb{R}^d$ and $y \in \mathbb{R}$.
- § Inputs are also often referred to as covariates, predictors and features; while outputs are known as variates, targets and labels.
- § Examples of such input-output pairs can be
  - ▶ {Outside temperature, People inside classroom, target room temperature | Energy requirement}
  - ▶ {Size, Number of Bedrooms, Number of Floors, Age of the Home | Price}
- § We have a set of N observations of y as $\{y^{(1)}, y^{(2)}, \cdots, y^{(N)}\}$ and the corresponding input variables $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(N)}\}$.



Linear Regression

§ The input and output variables are assumed to be related via a relation, known as hypothesis. $\widehat{y} = h_{\boldsymbol{\theta}}(\mathbf{x})$, where $\boldsymbol{\theta}$ is the parameter vector.

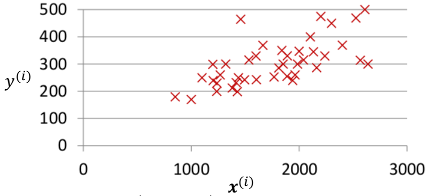
§ The goal is to predict the output variable $\widehat{y^*} = f(\mathbf{x}^*)$ for an arbitrary value of the input variable $\mathbf{x}^*$.

§ However, let us start with scalar inputs (x) and scalar outputs (y).

Univariate Linear Regression

- **hypothesis**: $h_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x$.
- **Cost Function**: Sum of squared errors.

$$J(\theta_0, \theta_1) = \frac{1}{2N} \sum_{i=1}^{N} \left(h_{\boldsymbol{\theta}}(x^{(i)}) - y^{(i)}\right)^2$$

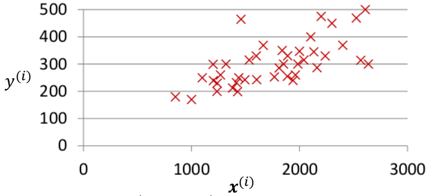


- **Optimization objective**: find model parameters (θ_0, θ_1) that will minimize the sum of squared errors.

Univariate Linear Regression

- hypothesis: $h_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x$.
- Cost Function: Sum of squared errors.

$$J(\theta_0, \theta_1) = \frac{1}{2N}\sum_{i=1}^{N}\left(h_{\boldsymbol{\theta}}(x^{(i)}) - y^{(i)}\right)^2$$



- Optimization objective: find model parameters (θ_0, θ_1) that will minimize the sum of squared errors.
- Machine learning involves optimization (minimization/maximization) in many contexts.
- The goal is to find parameters $\boldsymbol{\theta}$ of a hypothesis that significantly reduce a cost function or objective function $J(\boldsymbol{\theta})$.
- Its easy to spend a semester on optimization. Thus, we will very briefly scratch the surface to the level we need it here.

Machine Learning and Optimization

§ In learning we care about some performance measure P (*e.g.*, image classification accuracy, language translation accuracy *etc.*) on test set, but we minimize a different cost function $J(\boldsymbol{\theta})$ on training set, with the hope that doing so will improve P.

§ This is in contrast to pure optimization where minimizing $J(\boldsymbol{\theta})$ is a goal in itself.

Machine Learning and Optimization

- § In learning we care about some performance measure P (*e.g.*, image classification accuracy, language translation accuracy *etc.*) on test set, but we minimize a different cost function $J(\boldsymbol{\theta})$ on training set, with the hope that doing so will improve P.
- § This is in contrast to pure optimization where minimizing $J(\boldsymbol{\theta})$ is a goal in itself.
- § Gradient based optimization is the most popular way for training Machine Learning Models.

Machine Learning and Optimization

- In learning we care about some performance measure P (*e.g.*, image classification accuracy, language translation accuracy *etc.*) on test set, but we minimize a different cost function $J(\boldsymbol{\theta})$ on training set, with the hope that doing so will improve P.
- This is in contrast to pure optimization where minimizing $J(\boldsymbol{\theta})$ is a goal in itself.
- Gradient based optimization is the most popular way for training Machine Learning Models.
- But before going there, a quick brush-up of Vector/Matrix Calculus.

Vector/Matrix Calculus

§ **Gradient:** If $f(\mathbf{x}) \in \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^n$, then the gradient of $f(\mathbf{x})$ is defined as:

$$\nabla_{\mathbf{x}} f \triangleq \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}^T$$

Vector/Matrix Calculus

§ **Gradient:** If $f(\mathbf{x}) \in \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^n$, then the gradient of $f(\mathbf{x})$ is defined as:

$$\nabla_{\mathbf{x}} f \triangleq \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}^T$$

§ **Jacobian Matrix:** If $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})]^T \in \mathbb{R}^m$ and $\mathbf{x} \in \mathbb{R}^n$, then the Jacobian matrix of $\mathbf{f}(\mathbf{x})$ w.r.t. $\mathbf{x}$ is:

$$\nabla_{\mathbf{x}} \mathbf{f} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}_{m \times n}$$

Vector/Matrix Calculus

§ If $f(\mathbf{x}) \in \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^n$, then the **Hessian matrix** of $f(\mathbf{x})$ w.r.t. $\mathbf{x}$ is defined as:

$$\nabla^2_{\mathbf{x}} f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}_{n \times n}$$

§ Note: The Hessian captures the second-order curvature of the function.

Standard Results: Linear & Quadratic Forms

Here are some standard derivative results you will use frequently:

- **Derivative of Dot Product:**

$$\frac{\partial}{\partial \mathbf{x}}(\mathbf{b}^T\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{b}) = \mathbf{b}$$

- **Derivative of Linear Transformation:**

$$\frac{\partial}{\partial \mathbf{x}}(\mathbf{A}\mathbf{x}) = \mathbf{A}$$

- **Derivative of Quadratic Form:**

$$\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{A}\mathbf{x}) = (\mathbf{A} + \mathbf{A}^T)\mathbf{x}$$

- **Special Case (Symmetric A):**

$$\text{If } \mathbf{A} = \mathbf{A}^T, \quad \frac{\partial}{\partial \mathbf{x}}(\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}) = \mathbf{A}\mathbf{x}$$

Standard Results: Product and Chain Rules

Product Rule: For $\mathbf{u} \in \mathbb{R}^m, \mathbf{v} \in \mathbb{R}^m, \mathbf{x} \in \mathbb{R}^n$:

$$\left[\frac{\partial(\mathbf{u}^T\mathbf{v})}{\partial \mathbf{x}}\right]^T_{1\times n} = [\mathbf{u}^T]_{1\times m}\left[\frac{\partial \mathbf{v}}{\partial \mathbf{x}}\right]_{m\times n} + [\mathbf{v}^T]_{1\times m}\left[\frac{\partial \mathbf{u}}{\partial \mathbf{x}}\right]_{m\times n}$$

Chain rule - Scalar Function of Scalar Function: If $F(f(\mathbf{x})) \in \mathbb{R}$, $f(\mathbf{x}) \in \mathbb{R}$, and $\mathbf{x} \in \mathbb{R}^n$:

$$\left[\frac{\partial F}{\partial \mathbf{x}}\right]_{n\times 1} = \left[\frac{\partial f}{\partial \mathbf{x}}\right]_{n\times 1}\left[\frac{\partial F}{\partial f}\right]_{1\times 1}$$

Chain rule - Scalar Function of Vector Function: If $F(\mathbf{f}(\mathbf{x})) \in \mathbb{R}$, $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^m$, and $\mathbf{x} \in \mathbb{R}^n$:

$$\left[\frac{\partial F}{\partial \mathbf{x}}\right]_{n\times 1} = \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right]^T_{n\times m}\left[\frac{\partial F}{\partial \mathbf{f}}\right]_{m\times 1}$$

This vector chain rule is the foundation of Backpropagation!

Derivatives of Norms

These results are crucial for understanding Regularization (L2 norm) and Matrix Factorization.

Squared L2 Norm (Vector)

$$\frac{\partial}{\partial \mathbf{x}} ||\mathbf{x}||_2^2 = \frac{\partial}{\partial \mathbf{x}} (\mathbf{x}^T \mathbf{x}) = 2\mathbf{x}$$

Squared Frobenius Norm (Matrix)

$$\frac{\partial}{\partial \mathbf{X}} ||\mathbf{X}||_F^2 = \frac{\partial}{\partial \mathbf{X}} \text{tr}(\mathbf{X}\mathbf{X}^T) = 2\mathbf{X}$$

Finding Optimum

§ Optimization Problem Statement:

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \quad \text{subject to } \boldsymbol{\theta} \in \boldsymbol{\Theta}$$

§ We are after that $\boldsymbol{\theta}$ which gives the minimum value of the cost function $J(\boldsymbol{\theta})$.

Finding Optimum

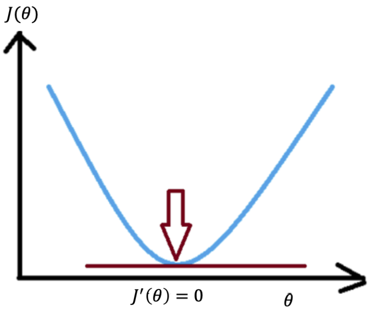
- Optimization Problem Statement:

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \quad \text{subject to } \boldsymbol{\theta} \in \boldsymbol{\Theta}$$

- We are after that $\boldsymbol{\theta}$ which gives the minimum value of the cost function $J(\boldsymbol{\theta})$.
- For scalar θ, the condition boils down to $\frac{\partial J}{\partial \theta} = 0$.
- For higher dimensional $\boldsymbol{\theta}$, the condition boils down to,

$$\nabla_{\boldsymbol{\theta}} J = \mathbf{0}$$

$$\begin{bmatrix} \frac{\partial J}{\partial \theta_1} & \frac{\partial J}{\partial \theta_2} & \cdots & \frac{\partial J}{\partial \theta_n} \end{bmatrix}^T = \mathbf{0}$$



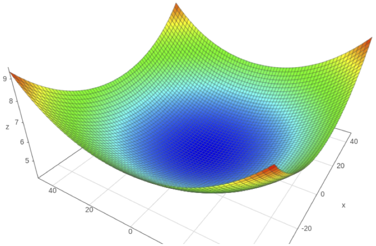
One Way to Find Minima – Gradient Descent

- This is helpful but not always useful.
- For example $J(\boldsymbol{\theta}) = \log \sum_{i=1}^{m} e^{(\mathbf{a}_i^T\boldsymbol{\theta}+b_i)}$ is a convex function with clear minima.
- But finding analytical solution is not easy.

$$\nabla_{\boldsymbol{\theta}} J = \mathbf{0}$$

$$\frac{1}{\sum_{i=1}^{m} e^{(\mathbf{a}_i^T\boldsymbol{\theta}+b_i)}} \sum_{i=1}^{m} e^{(\mathbf{a}_i^T\boldsymbol{\theta}+b_i)}\mathbf{a}_i = \mathbf{0}$$

$$\sum_{i=1}^{m} e^{(\mathbf{a}_i^T\boldsymbol{\theta}+b_i)}\mathbf{a}_i = \mathbf{0}$$



- So, a numerical iterative solution is sought for.

One Way to Find Minima – Gradient Descent

- Start with an initial guess $\boldsymbol{\theta}^0$.
- Repeatedly update $\boldsymbol{\theta}$ by taking a small step: $\boldsymbol{\theta}^k = \boldsymbol{\theta}^{k-1} + \eta\Delta\boldsymbol{\theta} \quad \cdots$ (a)
- so that $J(\boldsymbol{\theta})$ gets smaller with each update: $J(\boldsymbol{\theta}^k) \leq J(\boldsymbol{\theta}^{(k-1)}) \quad \cdots$ (b)

$$\begin{aligned} J(\boldsymbol{\theta}^k) &= J(\boldsymbol{\theta}^{k-1} + \eta\Delta\boldsymbol{\theta}) \\ &= J(\boldsymbol{\theta}^{k-1}) + \eta(\Delta\boldsymbol{\theta})^T J'(\boldsymbol{\theta}^{k-1}) + h.o.t. \text{ [Using Taylor series expansion]} \\ &\approx J(\boldsymbol{\theta}^{k-1}) + \eta(\Delta\boldsymbol{\theta})^T J'(\boldsymbol{\theta}^{k-1}) \text{ [neglecting h.o.t.]} \quad \cdots \text{(c)} \end{aligned}$$

- Combining (b) and (c), $\eta(\Delta\boldsymbol{\theta})^T J'(\boldsymbol{\theta}^{k-1}) \leq 0$ *i.e.*, $(\Delta\boldsymbol{\theta})^T J'(\boldsymbol{\theta}^{k-1}) \leq 0$, as η is positive.

One Way to Find Minima – Gradient Descent

§ Start with an initial guess $\boldsymbol{\theta}^0$.

§ Repeatedly update $\boldsymbol{\theta}$ by taking a small step: $\boldsymbol{\theta}^k = \boldsymbol{\theta}^{k-1} + \eta\Delta\boldsymbol{\theta} \quad \cdots$ (a)

§ so that $J(\boldsymbol{\theta})$ gets smaller with each update: $J(\boldsymbol{\theta}^k) \leq J(\boldsymbol{\theta}^{(k-1)}) \quad \cdots$ (b)

$$\begin{aligned} J(\boldsymbol{\theta}^k) &= J(\boldsymbol{\theta}^{k-1} + \eta\Delta\boldsymbol{\theta}) \\ &= J(\boldsymbol{\theta}^{k-1}) + \eta(\Delta\boldsymbol{\theta})^T J'(\boldsymbol{\theta}^{k-1}) + h.o.t. \text{ [Using Taylor series expansion]} \\ &\approx J(\boldsymbol{\theta}^{k-1}) + \eta(\Delta\boldsymbol{\theta})^T J'(\boldsymbol{\theta}^{k-1}) \text{ [neglecting h.o.t.]} \quad \cdots\text{(c)} \end{aligned}$$

§ Combining (b) and (c), $\eta(\Delta\boldsymbol{\theta})^T J'(\boldsymbol{\theta}^{k-1}) \leq 0$ *i.e.*, $(\Delta\boldsymbol{\theta})^T J'(\boldsymbol{\theta}^{k-1}) \leq 0$, as η is positive.

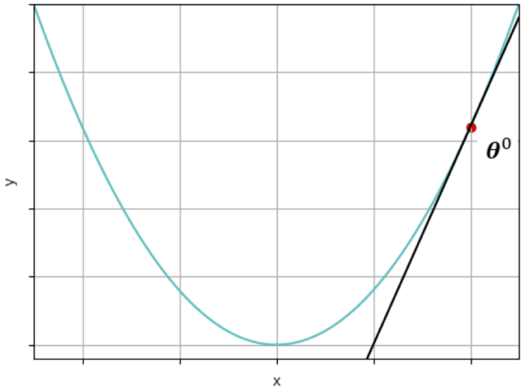
§ So, for $\boldsymbol{\theta}$ to minimize $J(\boldsymbol{\theta})$, *i.e.*, to satisfy (b) we have to choose some $\Delta\boldsymbol{\theta}$ that gives a negative dot product when multiplied with $J'(\boldsymbol{\theta}^{k-1})$.

§ Then why not choose, $\Delta\boldsymbol{\theta} = -J'(\boldsymbol{\theta}^{k-1})$.

§ Then, $(\Delta\boldsymbol{\theta})^T J'(\boldsymbol{\theta}^{k-1}) = -||J'(\boldsymbol{\theta}^{k-1})||_2^2$ surely is a negative quantity and satisfies the condition.

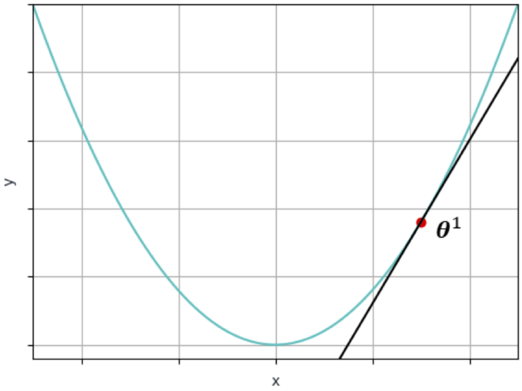
Gradient Descent

- Start with an initial guess $\boldsymbol{\theta}^0$.
- Repeatedly update $\boldsymbol{\theta}$ by taking a small step: $\boldsymbol{\theta}^k = \boldsymbol{\theta}^{k-1} - \eta J'(\boldsymbol{\theta}^{k-1})$, until convergence [$J'(\boldsymbol{\theta}^{k-1})$ is very small].



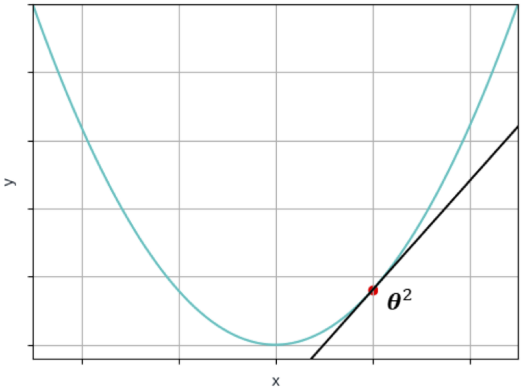
Gradient Descent

- Start with an initial guess $\boldsymbol{\theta}^0$.
- Repeatedly update $\boldsymbol{\theta}$ by taking a small step: $\boldsymbol{\theta}^k = \boldsymbol{\theta}^{k-1} - \eta J'(\boldsymbol{\theta}^{k-1})$, until convergence [$J'(\boldsymbol{\theta}^{k-1})$ is very small].



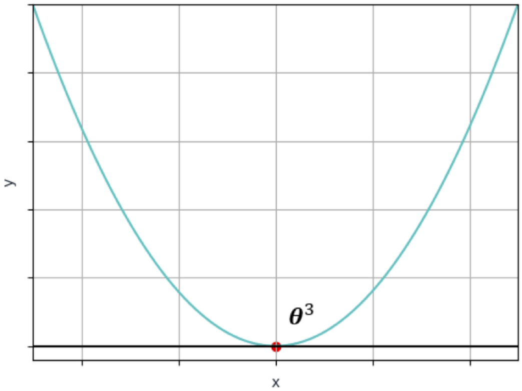
Gradient Descent

- Start with an initial guess $\boldsymbol{\theta}^0$.
- Repeatedly update $\boldsymbol{\theta}$ by taking a small step: $\boldsymbol{\theta}^k = \boldsymbol{\theta}^{k-1} - \eta J'(\boldsymbol{\theta}^{k-1})$, until convergence [$J'(\boldsymbol{\theta}^{k-1})$ is very small].

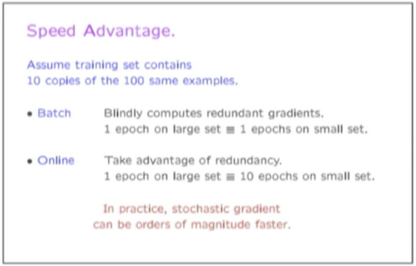


Gradient Descent

- Start with an initial guess $\boldsymbol{\theta}^0$.
- Repeatedly update $\boldsymbol{\theta}$ by taking a small step: $\boldsymbol{\theta}^k = \boldsymbol{\theta}^{k-1} - \eta J'(\boldsymbol{\theta}^{k-1})$, until convergence [$J'(\boldsymbol{\theta}^{k-1})$ is very small].



But Imagine This


Speed Advantage.

Assume training set contains
10 copies of the 100 same examples.

- Batch Blindly computes redundant gradients.
  1 epoch on large set ≡ 1 epochs on small set.
- Online Take advantage of redundancy.
  1 epoch on large set ≡ 10 epochs on small set.

In practice, stochastic gradient
can be orders of magnitude faster.

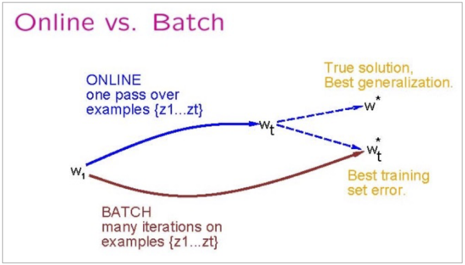
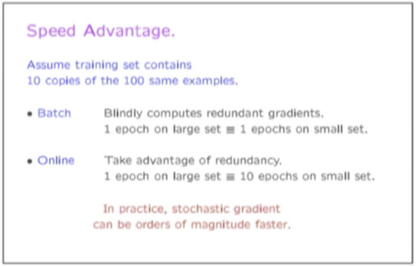


Figure credit: *Olivier Bousquet's talk at NeurIPS 2018 after winning Test of Time Award for NIPS 2007 paper:*
"The Trade Offs of Large Scale Learning" by Leon Bottou and Olivier Bousquet.

But Imagine This



Speed Advantage.

Assume training set contains
10 copies of the 100 same examples.

- Batch Blindly computes redundant gradients.
  1 epoch on large set ≡ 1 epochs on small set.
- Online Take advantage of redundancy.
  1 epoch on large set ≡ 10 epochs on small set.

In practice, stochastic gradient
can be orders of magnitude faster.

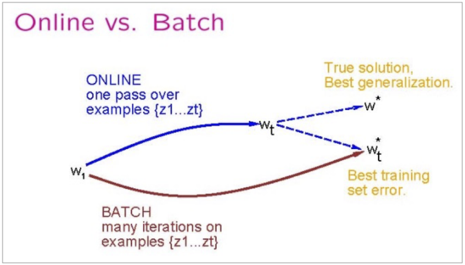


Figure credit: *Olivier Bousquet's talk at NeurIPS 2018 after winning Test of Time Award for NIPS 2007 paper: "The Trade Offs of Large Scale Learning" by Leon Bottou and Olivier Bousquet.*

- Optimization algorithms that use the entire training set to compute the gradient are called batch or deterministic gradient methods. Ones that use a single training example for that task are called stochastic or online gradient methods
- Most of the algorithms we use for machine learning fall somewhere in between!
- These are called minibatch or minibatch stochastic methods.

Batch, Stochastic and Mini-batch Stochastic Gradient Descent

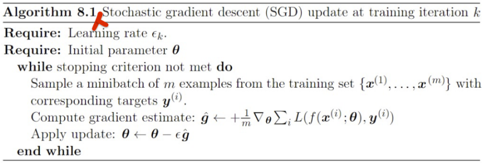
Algorithm 1 Batch Gradient Descent at Iteration k

```
Require: Learning rate $\epsilon_k$
Require: Initial Parameter $\theta$
1: while stopping criteria not met do
2:     Compute gradient estimate over $N$ examples:
3:     $\hat{\mathbf{g}} \leftarrow +\frac{1}{N}\nabla_\theta \sum_i L(f(\mathbf{x}^{(i)};\theta),\mathbf{y}^{(i)})$
4:     Apply Update: $\theta \leftarrow \theta - \epsilon\hat{\mathbf{g}}$
5: end while
```

Algorithm 2 Stochastic Gradient Descent at Iteration k

```
Require: Learning rate $\epsilon_k$
Require: Initial Parameter $\theta$
1: while stopping criteria not met do
2:     Sample example $(\mathbf{x}^{(i)},\mathbf{y}^{(i)})$ from training set
3:     Compute gradient estimate:
4:     $\hat{\mathbf{g}} \leftarrow +\nabla_\theta L(f(\mathbf{x}^{(i)};\theta),\mathbf{y}^{(i)})$
5:     Apply Update: $\theta \leftarrow \theta - \epsilon\hat{\mathbf{g}}$
6: end while
```

Mini-batch



Algorithm 8.1 Stochastic gradient descent (SGD) update at training iteration k

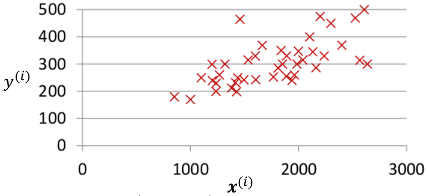
```
Require: Learning rate $\epsilon_k$.
Require: Initial parameter $\boldsymbol{\theta}$
  while stopping criterion not met do
    Sample a minibatch of $m$ examples from the training set $\{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow +\frac{1}{m}\nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)};\boldsymbol{\theta}),\boldsymbol{y}^{(i)})$
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon\hat{\boldsymbol{g}}$
  end while
```

Figure credit: *Shubhendu Trivedi et al., Goodfellow et al.*

Coming Back to Univariate Linear Regression

- hypothesis: $h_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x$.
- Cost Function: Sum of squared errors.

$$J(\theta_0, \theta_1) = \frac{1}{2N} \sum_{i=1}^{N} \left(h_{\boldsymbol{\theta}}(x^{(i)}) - y^{(i)}\right)^2$$

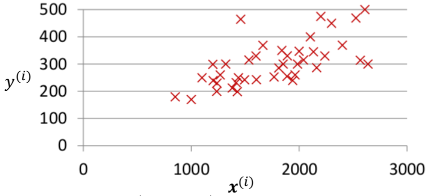


- Optimization objective: find model parameters (θ_0, θ_1) that will minimize the sum of squared errors.

Coming Back to Univariate Linear Regression

- hypothesis: $h_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x$.
- Cost Function: Sum of squared errors.

$$J(\theta_0, \theta_1) = \frac{1}{2N}\sum_{i=1}^{N}\left(h_{\boldsymbol{\theta}}(x^{(i)}) - y^{(i)}\right)^2$$



- Optimization objective: find model parameters (θ_0, θ_1) that will minimize the sum of squared errors.
- Gradient of the cost function w.r.t. θ_0:

$$\frac{J(\theta_0, \theta_1)}{\theta_0} = \frac{1}{N}\sum_{i=1}^{N}\left(h_{\boldsymbol{\theta}}(x^{(i)}) - y^{(i)}\right)$$

- Gradient of the cost function w.r.t. θ_1:

$$\frac{J(\theta_0, \theta_1)}{\theta_1} = \frac{1}{N}\sum_{i=1}^{N}\left(h_{\boldsymbol{\theta}}(x^{(i)}) - y^{(i)}\right)x^{(i)}$$

- Apply your favorite gradient based optimization algorithm.

Univariate Linear Regression

§ These being linear equations of $\boldsymbol{\theta}$, have a unique closed form solution too.

$$\theta_1 = \frac{N\sum_{i=1}^{N} y^{(i)}x^{(i)} - \big(\sum_{i=1}^{N} x^{(i)}\big)\big(\sum_{i=1}^{N} y^{(i)}\big)}{N\sum_{i=1}^{N}\big(x^{(i)}\big)^2 - \big(\sum_{i=1}^{N} x^{(i)}\big)^2}$$

$$\theta_0 = \frac{1}{N}\Big\{\sum_{i=1}^{N} y^{(i)} - \theta_1 \sum_{i=1}^{N} x^{(i)}\Big\}$$

Multivariate Linear Regression

- We can easily extend to multivariate linear regression problems, where $\mathbf{x} \in \mathbb{R}^d$
- hypothesis: $h_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d$. For convenience of notation, define $x_0 = 1$.
- Thus h is simply the dot product of the parameters and the input vector.

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x}$$

- Cost Function: Sum of squared errors.

$$J(\boldsymbol{\theta}) = J(\theta_0, \theta_1, \cdots, \theta_d) = \frac{1}{2N} \sum_{i=1}^{N} \left(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}\right)^2 \tag{1}$$

- We will use the following to write the cost function in a compact matrix vector notation

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} = \mathbf{x}^T \boldsymbol{\theta}$$

Multivariate Linear Regression

$$\begin{bmatrix} \widehat{y}^{(1)} \\ \widehat{y}^{(2)} \\ \vdots \\ \widehat{y}^{(N)} \end{bmatrix} = \begin{bmatrix} h_\theta(\mathbf{x^{(1)}}) \\ h_\theta(\mathbf{x^{(2)}}) \\ \vdots \\ h_\theta(\mathbf{x^{(N)}}) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_0^{(1)} & \mathbf{x}_1^{(1)} & \mathbf{x}_2^{(1)} & \cdots & \mathbf{x}_d^{(1)} \\ \mathbf{x}_0^{(2)} & \mathbf{x}_1^{(2)} & \mathbf{x}_2^{(2)} & \cdots & \mathbf{x}_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \\ \mathbf{x}_0^{(N)} & \mathbf{x}_1^{(N)} & \mathbf{x}_2^{(N)} & \cdots & \mathbf{x}_d^{(N)} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix} \quad (2)$$

$$\widehat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta}$$

Here, $\mathbf{X}$ is a $N \times (d+1)$ matrix with each row an input vector. $\widehat{\mathbf{y}}$ is a N length vector of the outputs in the training set.

Multivariate Linear Regression

§ Eqn. (1), gives,

$$J(\boldsymbol{\theta}) = \frac{1}{2N}\sum_{i=1}^{N}\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)} - y^{(i)}\right)^2 = \frac{1}{2N}\sum_{i=1}^{N}\left(\widehat{y}^{(i)} - y^{(i)}\right)^2 \tag{3}$$

$$= \frac{1}{2N}||\widehat{\mathbf{y}} - \mathbf{y}||_2^2 = \frac{1}{2N}\left(\widehat{\mathbf{y}} - \mathbf{y}\right)^T\left(\widehat{\mathbf{y}} - \mathbf{y}\right)$$

$$= \frac{1}{2N}\left(\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\right)^T\left(\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\right) = \frac{1}{2N}\left\{\boldsymbol{\theta}^T\left(\mathbf{X}^T\mathbf{X}\right)\boldsymbol{\theta} - \boldsymbol{\theta}^T\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\boldsymbol{\theta} + \mathbf{y}^T\mathbf{y}\right\}$$

$$= \frac{1}{2N}\left\{\boldsymbol{\theta}^T\left(\mathbf{X}^T\mathbf{X}\right)\boldsymbol{\theta} - \left(\mathbf{X}^T\mathbf{y}\right)^T\boldsymbol{\theta} - \left(\mathbf{X}^T\mathbf{y}\right)^T\boldsymbol{\theta} + \mathbf{y}^T\mathbf{y}\right\}$$

$$= \frac{1}{2N}\left\{\boldsymbol{\theta}^T\left(\mathbf{X}^T\mathbf{X}\right)\boldsymbol{\theta} - 2\left(\mathbf{X}^T\mathbf{y}\right)^T\boldsymbol{\theta} + \mathbf{y}^T\mathbf{y}\right\}$$

Multivariate Linear Regression

§ Equating the gradient of the cost function to 0,

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{2N}\left\{2\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} - 2\mathbf{X}^T\mathbf{y} + 0\right\} = 0 \\ \mathbf{X}^T\mathbf{X}\boldsymbol{\theta} - \mathbf{X}^T\mathbf{y} = 0 \\ \boldsymbol{\theta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \qquad (4)\end{aligned}$$

Multivariate Linear Regression

§ Equating the gradient of the cost function to 0,

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \frac{1}{2N}\{2\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} - 2\mathbf{X}^T\mathbf{y} + 0\} = 0 \\ \mathbf{X}^T\mathbf{X}\boldsymbol{\theta} - \mathbf{X}^T\mathbf{y} &= 0 \\ \boldsymbol{\theta} &= \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \end{aligned} \tag{4}$$

§ This gives a closed form solution, but another option is to use iterative solution (just like the univariate case).

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

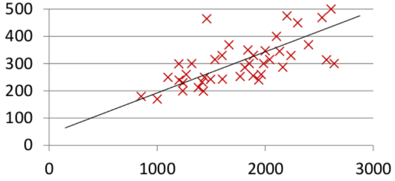
Multivariate Linear Regression

§ Iterative Gradient Descent needs to perform many iterations and need to choose a stepsize parameter judiciously. But it works equally well even if the number of features (d) is large.

§ For the least square solution, there is no need to choose the step size parameter or no need to iterate. But, evaluating $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ can be slow if d is large.

Linear Regression as Maximum Likelihood Estimation

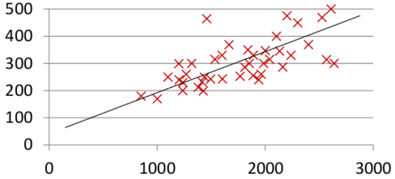
- So far we tried to fit a "straightline" ("hyperplane" to be more precise) for linear regression problem.
- This is, in a sense, a "constrained" way of looking at the problem. Datapoints may not be perfectly fit to the hyperplane, but "how uncertain" they are from the hyperplane is never considered.



Linear Regression as Maximum Likelihood Estimation

§ So far we tried to fit a "straightline" ("hyperplane" to be more precise) for linear regression problem.

§ This is, in a sense, a "constrained" way of looking at the problem. Datapoints may not be perfectly fit to the hyperplane, but "how uncertain" they are from the hyperplane is never considered.



§ An alternate view considers the following.

- $y^{(i)}$ are generated from the $\mathbf{x}^{(i)}$ following a underlying hyperplane.
- But we don't get to "see" the generated data. Instead we "see" a noisy version of the $y^{(i)}$'s.
- Maximum likelihood (or in general, probabilistic estimation) models this uncertainty in determining the data generating function.

Linear Regression as Maximum Likelihood Estimation

§ Thus data are assumed to be generated as follows.

$$y^{(i)} = h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) + \epsilon^{(i)}$$

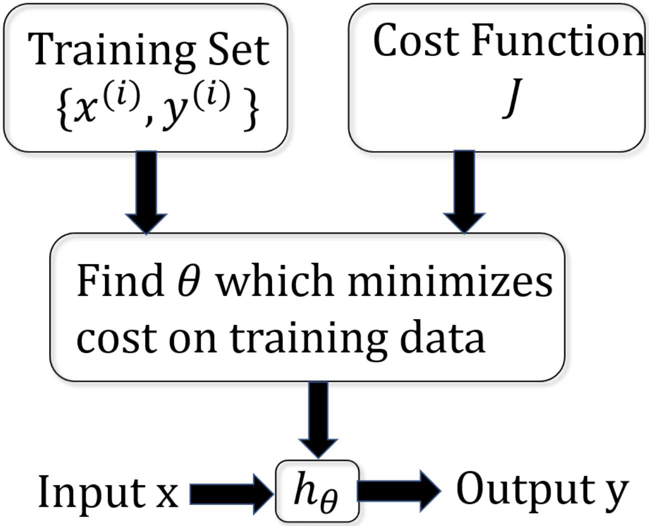
where $\epsilon^{(i)}$ is an additive noise following some probability distribution.

§ So, $\left(\mathbf{x}^{(i)}, y^{(i)}\right)$'s form a joint distribution.

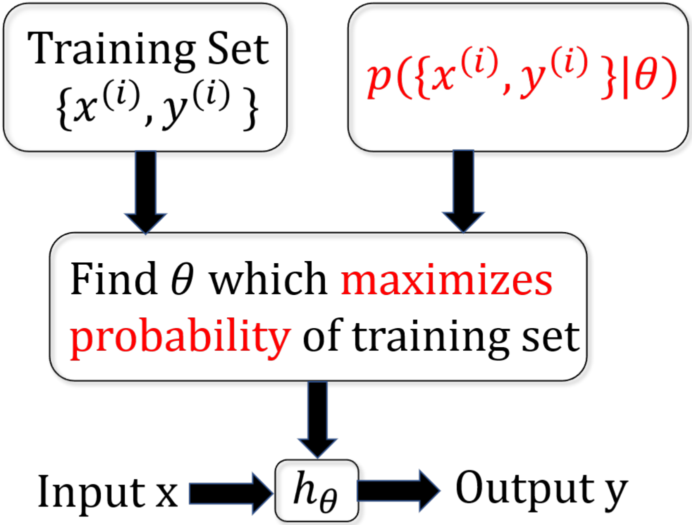
§ The idea is to assume a probability distribution on the noise and the probability distribution is parameterised by some additional parameters (*e.g.*, Gaussian with 0 mean and covariance σ^2).

§ Then find the parameters (both $\boldsymbol{\theta}$ and σ^2) that is "most likely" to generate the data.

Recall: Cost Function



Alternate View: "Maximum Likelihood"


Training Set
$\{x^{(i)}, y^{(i)}\}$
$p(\{x^{(i)}, y^{(i)}\}|\theta)$
Find θ which maximizes probability of training set
Input x
h_θ
Output y

Maximum Likelihood for Linear Regression

- Let us assume that the noise is Gaussian distributed with mean 0 and variance σ^2
$$y^{(i)} = h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) + \epsilon^{(i)} = \boldsymbol{\theta}^T\mathbf{x}^{(i)} + \epsilon^{(i)}$$
- Noise $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$ and thus $y^{(i)} \sim \mathcal{N}(\boldsymbol{\theta}^T\mathbf{x}^{(i)}, \sigma^2)$.

Maximum Likelihood for Linear Regression

- Let us assume that the noise is Gaussian distributed with mean 0 and variance σ^2
$$y^{(i)} = h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) + \epsilon^{(i)} = \boldsymbol{\theta}^T\mathbf{x}^{(i)} + \epsilon^{(i)}$$
- Noise $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$ and thus $y^{(i)} \sim \mathcal{N}(\boldsymbol{\theta}^T\mathbf{x}^{(i)}, \sigma^2)$.
- Let us compute the likelihood.

$$\begin{aligned} p(\mathbf{y}|\mathbf{X};\boldsymbol{\theta},\sigma^2) &= \prod_{i=1}^{N} p(y^{(i)}|\mathbf{x}^{(i)};\boldsymbol{\theta},\sigma^2) \\ &= \prod_{i=1}^{N} (2\pi\sigma^2)^{-\frac{1}{2}}\, e^{-\frac{1}{2\sigma^2}\left(y^{(i)}-\boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)^2} \\ &= (2\pi\sigma^2)^{-\frac{N}{2}}\, e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(y^{(i)}-\boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)^2} \\ &= (2\pi\sigma^2)^{-\frac{N}{2}}\, e^{-\frac{1}{2\sigma^2}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\right)^T\left(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\right)} \end{aligned} \tag{5}$$

Maximum Likelihood for Linear Regression

§ So we have got the likelihood as,

$$p(\mathbf{y}|\mathbf{X};\boldsymbol{\theta},\sigma^2) = (2\pi\sigma^2)^{-\frac{N}{2}}\, e^{-\frac{1}{2\sigma^2}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\right)^T\left(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\right)}$$

§ The log likelihood is

$$l(\boldsymbol{\theta},\sigma^2) = -\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\right)^T\left(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\right)$$

Maximum Likelihood for Linear Regression

- So we have got the likelihood as,

$$p(\mathbf{y}|\mathbf{X};\boldsymbol{\theta},\sigma^2) = (2\pi\sigma^2)^{-\frac{N}{2}}\, e^{-\frac{1}{2\sigma^2}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\right)^T\left(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\right)}$$

- The log likelihood is

$$l(\boldsymbol{\theta},\sigma^2) = -\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\right)^T\left(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\right)$$

- Maximizing the likelihood w.r.t. $\boldsymbol{\theta}$ means maximizing $-\left(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\right)^T\left(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\right)$ which in turn means minimizing $\left(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\right)^T\left(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\right)$.
- Note the similarity with what we did earlier.
- Thus linear regression can be equivalently viewed as minimizing error sum of squares as well as maximum likelihood estimation under zero mean Gaussian noise assumption.

Maximum Likelihood for Linear Regression

§ In a similar manner, the maximum likelihood estimate of σ^2 can also be calculated.

§ Remember, the log likelihood can be written as,

$$l(\boldsymbol{\theta}, \sigma^2) = -\frac{N}{2}\log(2\pi) - \frac{N}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T x^{(i)}\right)^2$$

§ Now, we take the partial derivative with respect to σ^2 (let's treat $v = \sigma^2$ as our variable for simplicity).

$$\frac{\partial l}{\partial v} = -\frac{N}{2}\frac{\partial}{\partial v}\log(v) - \frac{1}{2}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T x^{(i)}\right)^2 \frac{\partial}{\partial v}(v^{-1})$$

§ Using standard derivatives $\frac{d}{dv}\ln(v) = \frac{1}{v}$ and $\frac{d}{dv}v^{-1} = -v^{-2}$:

$$\frac{\partial l}{\partial \sigma^2} = -\frac{N}{2\sigma^2} + \frac{1}{2(\sigma^2)^2}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T x^{(i)}\right)^2$$

Maximum Likelihood for Linear Regression

§ Now, we set the derivative to zero to find the optimum.

$$-\frac{N}{2\sigma^2} + \frac{1}{2(\sigma^2)^2}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T x^{(i)}\right)^2 = 0$$

$$-N\sigma^2 + \sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T x^{(i)}\right)^2 = 0$$

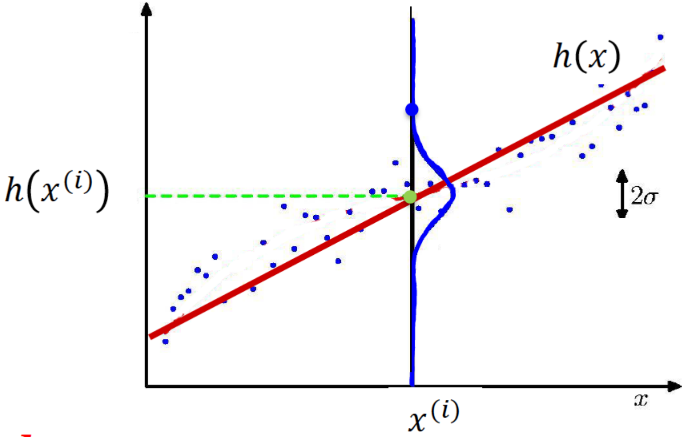
§ Rearranging terms gives us the MLE estimator:

$$\sigma^2_{MLE} = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T_{MLE} x^{(i)}\right)^2$$

Interpretation

The MLE estimate for the variance is simply the **average squared residual** (Mean Squared Error) of the model on the training data.

Maximum Likelihood for Linear Regression



Recall: Ordinary Least Squares (OLS)

In standard Linear Regression, we minimize the Sum of Squared Errors:

$$J(\boldsymbol{\theta}) = ||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}||_2^2$$

The closed-form solution (MLE) is:

$$\widehat{\boldsymbol{\theta}}_{MLE} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Recall: Ordinary Least Squares (OLS)

In standard Linear Regression, we minimize the Sum of Squared Errors:

$$J(\boldsymbol{\theta}) = ||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}||_2^2$$

The closed-form solution (MLE) is:

$$\widehat{\boldsymbol{\theta}}_{MLE} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Potential Problem

This requires inverting $\mathbf{X}^T\mathbf{X}$.

- What if $\mathbf{X}^T\mathbf{X}$ is ill-conditioned?
- What if features are highly correlated (multicollinearity)?
- inverting the matrix is numerically unstable

The Fix: Ridge Regression

Idea: Add a small positive element $(\delta^2 > 0)$ to the diagonal of $\mathbf{X}^T\mathbf{X}$.

$$\mathbf{X}^T\mathbf{X} \quad \longrightarrow \quad (\mathbf{X}^T\mathbf{X} + \delta^2\mathbf{I})$$

This ensures the matrix is always Full Rank (invertible) and well-conditioned.

The Fix: Ridge Regression

Idea: Add a small positive element ($\delta^2 > 0$) to the diagonal of $\mathbf{X}^T\mathbf{X}$.

$$\mathbf{X}^T\mathbf{X} \quad \longrightarrow \quad (\mathbf{X}^T\mathbf{X} + \delta^2\mathbf{I})$$

This ensures the matrix is always Full Rank (invertible) and well-conditioned.

Ridge Regression Estimate

$$\hat{\boldsymbol{\theta}}_R = (\mathbf{X}^T\mathbf{X} + \delta^2\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

Implied Cost Function: It turns out (via calculus) that this is solution to the following **regularized quadratic cost function**.

$$J(\boldsymbol{\theta}) = \underbrace{||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}||_2^2}_{\text{Error}} \quad + \quad \underbrace{\delta^2||\boldsymbol{\theta}||_2^2}_{\text{Penalty/Regularizer}}$$

A Deeper Question

We fixed the numerical problem by adding $\delta^2\mathbf{I}$.

Is this just a mathematical hack?

- Does minimizing $||\boldsymbol{\theta}||^2$ have a statistical meaning?
- Why squared norm? Why not absolute value?

Let's look at this from a **Bayesian Probability** *perspective.*

Bayesian Linear Regression: The Setup

Let's incorporate a **Prior Belief** about our weights $\boldsymbol{\theta}$.

1. The Likelihood (Data)

Assume standard Gaussian noise (Like we did earlier). In Eqn. (5), we saw:

$$p(\mathbf{y}|\mathbf{X};\boldsymbol{\theta},\sigma^2) = (2\pi\sigma^2)^{-\frac{N}{2}}\, e^{-\frac{1}{2\sigma^2}\left(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\right)^T\left(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}\right)}$$

2. The Prior (Belief)

We assume weights are likely small (centered at 0). Let's use a **Gaussian Prior**:

$$\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \tau^2\mathbf{I})$$

The PDF is given by:

$$p(\boldsymbol{\theta}) = (2\pi\tau^2)^{-\frac{D}{2}}\, e^{-\frac{1}{2\tau^2}\boldsymbol{\theta}^T\boldsymbol{\theta}}$$

Deriving the MAP Estimate

The MAP estimate maximizes the log-posterior:

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\max_{\boldsymbol{\theta}} \left[\ln p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) + \ln p(\boldsymbol{\theta})\right]$$

Expanding the terms:

$$\begin{aligned} \ln p(\mathbf{y}|\dots) &= -\frac{N}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) \\ \ln p(\boldsymbol{\theta}) &= -\frac{D}{2}\ln(2\pi\tau^2) - \frac{1}{2\tau^2}\boldsymbol{\theta}^T\boldsymbol{\theta} \end{aligned}$$

Dropping terms that are constant *w.r.t.* $\boldsymbol{\theta}$:

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\max_{\boldsymbol{\theta}} \left[-\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) - \frac{1}{2\tau^2}\boldsymbol{\theta}^T\boldsymbol{\theta}\right]$$

Deriving the MAP Estimate

From previous slide:

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\max_{\boldsymbol{\theta}} \left[-\frac{1}{2\sigma^2} ||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}||_2^2 - \frac{1}{2\tau^2} ||\boldsymbol{\theta}||_2^2 \right]$$

Converting maximization to minimization, (multiply by $-2\sigma^2$):

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\min_{\boldsymbol{\theta}} \left[||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}||_2^2 + \frac{2\sigma^2}{2\tau^2} ||\boldsymbol{\theta}||_2^2 \right]$$

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\min_{\boldsymbol{\theta}} \left[\underbrace{||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}||_2^2}_{\text{SSE}} + \underbrace{\frac{\sigma^2}{\tau^2}}_{\delta^2} \underbrace{||\boldsymbol{\theta}||_2^2}_{\text{Regularizer}} \right]$$

Conclusion

MAP with Gaussian Prior is identical to Ridge Regression with $\delta^2 = \frac{\sigma^2}{\tau^2}$.

Deriving the MAP Estimate

From previous slide:

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\max_{\boldsymbol{\theta}} \left[-\frac{1}{2\sigma^2} ||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}||_2^2 - \frac{1}{2\tau^2} ||\boldsymbol{\theta}||_2^2 \right]$$

Converting maximization to minimization, (multiply by $-2\sigma^2$):

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\min_{\boldsymbol{\theta}} \left[||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}||_2^2 + \frac{2\sigma^2}{2\tau^2} ||\boldsymbol{\theta}||_2^2 \right]$$

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\min_{\boldsymbol{\theta}} \left[\underbrace{||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}||_2^2}_{\text{SSE}} + \underbrace{\frac{\sigma^2}{\tau^2}}_{\delta^2} \underbrace{||\boldsymbol{\theta}||_2^2}_{\text{Regularizer}} \right]$$

Conclusion

MAP with Gaussian Prior is identical to Ridge Regression with $\delta^2 = \frac{\sigma^2}{\tau^2}$.

Deriving the MAP Estimate

From previous slide:

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\max_{\boldsymbol{\theta}} \left[-\frac{1}{2\sigma^2} ||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}||_2^2 - \frac{1}{2\tau^2} ||\boldsymbol{\theta}||_2^2 \right]$$

Converting maximization to minimization, (multiply by $-2\sigma^2$):

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\min_{\boldsymbol{\theta}} \left[||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}||_2^2 + \frac{2\sigma^2}{2\tau^2} ||\boldsymbol{\theta}||_2^2 \right]$$

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\min_{\boldsymbol{\theta}} \left[\underbrace{||\mathbf{y} - \mathbf{X}\boldsymbol{\theta}||_2^2}_{\text{SSE}} + \underbrace{\frac{\sigma^2}{\tau^2}}_{\delta^2} \underbrace{||\boldsymbol{\theta}||_2^2}_{\text{Regularizer}} \right]$$

Conclusion

MAP with Gaussian Prior is identical to Ridge Regression with $\delta^2 = \frac{\sigma^2}{\tau^2}$.

Deriving the Ridge Estimator

We found the Objective Function:

$$J(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \delta^2\boldsymbol{\theta}^T\boldsymbol{\theta}$$

To find the optimal $\boldsymbol{\theta}$, we take the gradient $\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta})$ and set it to 0.

Step 1: Expand the terms

$$\begin{aligned} J(\boldsymbol{\theta}) &= (\mathbf{y}^T - \boldsymbol{\theta}^T\mathbf{X}^T)(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \delta^2\boldsymbol{\theta}^T\boldsymbol{\theta} \\ &= \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\boldsymbol{\theta} - \boldsymbol{\theta}^T\mathbf{X}^T\mathbf{y} + \boldsymbol{\theta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} + \delta^2\boldsymbol{\theta}^T\boldsymbol{\theta} \end{aligned}$$

Note that $\mathbf{y}^T\mathbf{X}\boldsymbol{\theta}$ is a scalar, so it equals its transpose $\boldsymbol{\theta}^T\mathbf{X}^T\mathbf{y}$.

$$J(\boldsymbol{\theta}) = \mathbf{y}^T\mathbf{y} - 2\boldsymbol{\theta}^T\mathbf{X}^T\mathbf{y} + \boldsymbol{\theta}^T(\mathbf{X}^T\mathbf{X})\boldsymbol{\theta} + \delta^2\boldsymbol{\theta}^T\boldsymbol{\theta}$$

Deriving the Ridge Estimator

Step 2: Take the Gradient
Using matrix calculus rules $\nabla_{\boldsymbol{\theta}}(\boldsymbol{\theta}^T\mathbf{A}\boldsymbol{\theta}) = 2\mathbf{A}\boldsymbol{\theta}$ (for symmetric $\mathbf{A}$) and $\nabla_{\boldsymbol{\theta}}(\mathbf{b}^T\boldsymbol{\theta}) = \mathbf{b}$,

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = 0 - 2\mathbf{X}^T\mathbf{y} + 2(\mathbf{X}^T\mathbf{X})\boldsymbol{\theta} + 2\delta^2\boldsymbol{\theta}$$

Deriving the Ridge Estimator

Step 2: Take the Gradient

Using matrix calculus rules $\nabla_{\boldsymbol{\theta}}(\boldsymbol{\theta}^T\mathbf{A}\boldsymbol{\theta}) = 2\mathbf{A}\boldsymbol{\theta}$ (for symmetric $\mathbf{A}$) and $\nabla_{\boldsymbol{\theta}}(\mathbf{b}^T\boldsymbol{\theta}) = \mathbf{b}$,

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = 0 - 2\mathbf{X}^T\mathbf{y} + 2(\mathbf{X}^T\mathbf{X})\boldsymbol{\theta} + 2\delta^2\boldsymbol{\theta}$$

Step 3: Set Gradient to Zero

$$-2\mathbf{X}^T\mathbf{y} + 2(\mathbf{X}^T\mathbf{X})\boldsymbol{\theta} + 2\delta^2\mathbf{I}\boldsymbol{\theta} = 0$$

$$(\mathbf{X}^T\mathbf{X} + \delta^2\mathbf{I})\boldsymbol{\theta} = \mathbf{X}^T\mathbf{y}$$

Step 4: Solve for θ

$$\hat{\boldsymbol{\theta}}_{Ridge} = (\mathbf{X}^T\mathbf{X} + \delta^2\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

This matches the "stabilized" inverse we saw earlier!

Changing the Prior: Ridge vs. Lasso

What if we used a **Laplacian Prior** ($P(\boldsymbol{\theta}) \propto e^{-|\boldsymbol{\theta}|}$) instead?

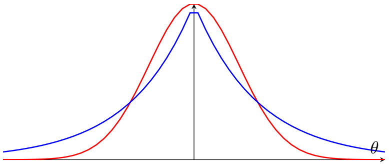
Ridge (L2)

- Prior: Gaussian
- Penalty: $\delta^2||\boldsymbol{\theta}||_2^2$
- Solution: $\boldsymbol{\theta}$ is small, but non-zero.
- **Use case:** Handling multicollinearity (Matrix Conditioning).

Lasso (L1)

- Prior: Laplacian
- Penalty: $\delta^2||\boldsymbol{\theta}||_1$
- Solution: Sparse (many $\theta_i = 0$).
- **Use case:** Feature Selection.

Gaussian (Red) vs Laplacian (Blue)

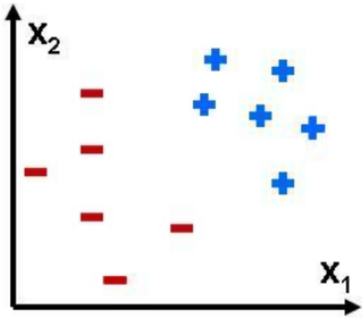


Summary: The Equivalence Table

Method	**Algebraic View**	**Probabilistic View**
OLS	Minimize SSE	MLE (Gaussian Noise)
Ridge	Stabilize Inverse $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})$	MAP (Gaussian Prior)
Lasso	Constraint $\|\|\boldsymbol{\theta}\|\|_1 \leq t$	MAP (Laplacian Prior)

Classification

- $y \in \{0,1\}$, where $0:$ "Negative class" (*e.g.*, benign tumor), $1:$ "Positive class" (*e.g.*, malignant tumor)
- Some more examples:
  - Email: Spam/ Not Spam?
  - Video: Viral/Not Viral?
  - Tremor: Earthquake/Nuclear explosion?



Linear classifiers with hard threshold

- Linear functions can be used to do classification as well as regression.
- For example,

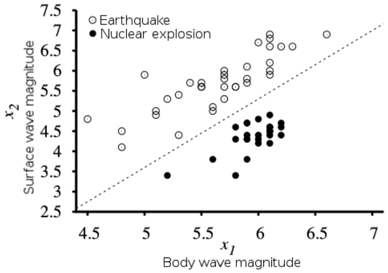


Figure credit: *AIMA: Russell, Norvig*

- A **decision boundary** is a line (or a surface, in higher dimensions) that separates the two classes.
- A linear function gives rise to a **linear separator** and the data that results in such a separator are called **linearly separable**.

Linear Classifier with Hard Threshold

§ The linear separator in the associated fig is given by,

$$x_2 = 1.7x_1 - 4.9$$

$$\implies -4.9 + 1.7x_1 - x_2 = 0$$

$$\implies [-4.9, 1.7, -1] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = 0$$

$$\boldsymbol{\theta}^T \mathbf{x} = 0$$

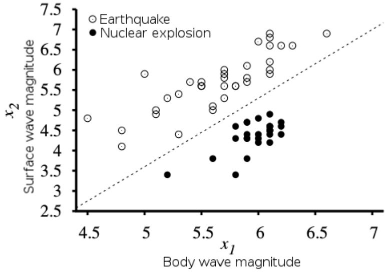


Figure credit: *AIMA: Russell, Norvig*

Linear Classifier with Hard Threshold

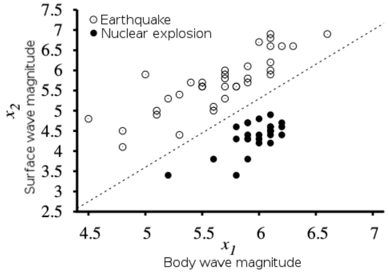


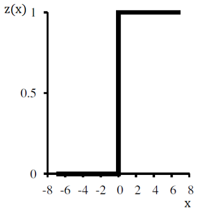
Figure credit: *AIMA: Russell, Norvig*

- The explosions $(y = 1)$ are to the right of this line with higher values of x_1 and lower values of x_2. So, they are points for which $\boldsymbol{\theta}^T\mathbf{x} \geq 0$
- Similarly earthquakes $(y = 0)$ are to the left of this line. So, they are points for which $\boldsymbol{\theta}^T\mathbf{x} < 0$
- The classification rule is then,

$$y(\mathbf{x}) = \begin{cases} 1 & \text{if } \boldsymbol{\theta}^T\mathbf{x} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

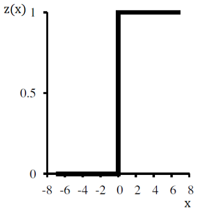
Linear classifiers with hard threshold

- Alternatively, we can think y as the result of passing the linear function $\boldsymbol{\theta}^T\mathbf{x}$ through a threshold function.



Linear classifiers with hard threshold

§ Alternatively, we can think y as the result of passing the linear function $\boldsymbol{\theta}^T\mathbf{x}$ through a threshold function.



§ To get the linear separator we have find the θ which minimizes classification error on the training set.

§ For regression problems, we found θ in both closed form and by gradient descent. But both approaches required us to compute the gradient.

§ This is not possible for the above threshold function as the gradient is undefined when the *value* at $x - axis = 0$ and 0 elsewhere.

Linear classifiers with hard threshold

- Perceptron Rule - This algorithm does not compute the gradient to find θ.

Linear classifiers with hard threshold

- Perceptron Rule - This algorithm does not compute the gradient to find θ.
- Perceptron Learning Rule can find a linear separator $\boxed{\text{given the data is linearly separable}}$.
- For data that are not linearly separable, the Perceptron algorithm fails.

Linear classifiers with hard threshold

- Perceptron Rule - This algorithm does not compute the gradient to find θ.
- Perceptron Learning Rule can find a linear separator $\boxed{\text{given the data is linearly separable}}$.
- For data that are not linearly separable, the Perceptron algorithm fails.
- So, we need to go for a gradient based optimization approach
- Thus, we need to approximate the hard threshold function with something smooth.

$$\sigma(u) = \frac{1}{1+e^{-u}}$$
$$y = \sigma(h_\theta(x)) = \sigma(\boldsymbol{\theta}^T\mathbf{x})$$

- Notice that the output is a number between 0 and 1, so it can be interpreted as a probability value belonging to Class 1.
- This is called a logistic regression classifier. The gradient computation is tedious but straight forward.

Maximum Likelihood Estimation of Logistic Regression

- Mathematically, the probability that an example belongs to class 1 is $P(y^{(i)} = 1|\mathbf{x}^{(i)}; \boldsymbol{\theta}) = \sigma\big(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\big)$
- Similarly, $P(y^{(i)} = 0|\mathbf{x}^{(i)}; \boldsymbol{\theta}) = 1 - \sigma\big(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\big)$
- Thus, $P(y^{(i)}|\mathbf{x}^{(i)}; \boldsymbol{\theta}) = \Big(\sigma\big(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\big)\Big)^{y^{(i)}}\Big(1 - \sigma\big(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\big)\Big)^{(1-y^{(i)})}$

Maximum Likelihood Estimation of Logistic Regression

- Mathematically, the probability that an example belongs to class 1 is $P(y^{(i)} = 1|\mathbf{x}^{(i)}; \boldsymbol{\theta}) = \sigma\big(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\big)$
- Similarly, $P(y^{(i)} = 0|\mathbf{x}^{(i)}; \boldsymbol{\theta}) = 1 - \sigma\big(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\big)$
- Thus, $P(y^{(i)}|\mathbf{x}^{(i)}; \boldsymbol{\theta}) = \Big(\sigma\big(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\big)\Big)^{y^{(i)}} \Big(1 - \sigma\big(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\big)\Big)^{(1-y^{(i)})}$
- The joint probability of all the labels $\prod\limits_{i=1}^{N} \Big(\sigma\big(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\big)\Big)^{y^{(i)}} \Big(1 - \sigma\big(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\big)\Big)^{(1-y^{(i)})}$

Maximum Likelihood Estimation of Logistic Regression

- Mathematically, the probability that an example belongs to class 1 is $P(y^{(i)} = 1|\mathbf{x}^{(i)}; \boldsymbol{\theta}) = \sigma\big(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\big)$
- Similarly, $P(y^{(i)} = 0|\mathbf{x}^{(i)}; \boldsymbol{\theta}) = 1 - \sigma\big(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\big)$
- Thus, $P(y^{(i)}|\mathbf{x}^{(i)}; \boldsymbol{\theta}) = \Big(\sigma\big(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\big)\Big)^{y^{(i)}} \Big(1 - \sigma\big(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\big)\Big)^{(1-y^{(i)})}$
- The joint probability of all the labels $\prod\limits_{i=1}^{N} \Big(\sigma\big(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\big)\Big)^{y^{(i)}} \Big(1 - \sigma\big(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\big)\Big)^{(1-y^{(i)})}$
- So the log likelihood for logistic regression is given by,

$$l(\boldsymbol{\theta}) = \sum_{i=1}^{N} y^{(i)} \log \sigma\big(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\big) + (1 - y^{(i)}) \log \big(1 - \sigma\big(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\big)\big)$$

Maximum Likelihood Estimation of Logistic Regression

§ Derivative of log likelihood w.r.t. one component of $\boldsymbol{\theta}$,

$$\begin{aligned}
\frac{\partial l(\boldsymbol{\theta})}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j} \sum_{i=1}^{N} y^{(i)} \log \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right) + (1 - y^{(i)}) \log\left(1 - \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)\right) \\
&= \sum_{i=1}^{N} \left[\frac{y^{(i)}}{\sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)} - \frac{1 - y^{(i)}}{1 - \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)}\right] \frac{\partial}{\partial \theta_j} \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right) \\
&= \sum_{i=1}^{N} \left[\frac{y^{(i)}}{\sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)} - \frac{1 - y^{(i)}}{1 - \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)}\right] \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right) \left(1 - \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)\right) \mathbf{x}_j^{(i)} \\
&= \sum_{i=1}^{N} \left[\frac{y^{(i)} - \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)}{\sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)\left(1 - \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)\right)}\right] \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right) \left(1 - \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)\right) \mathbf{x}_j^{(i)} \\
&= \sum_{i=1}^{N} \left[y^{(i)} - \sigma\left(\boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)\right] \mathbf{x}_j^{(i)} \qquad (6)
\end{aligned}$$

§ This is used in an iterative gradient ascent loop.