Contents

Machine Learning

June 14, 2016

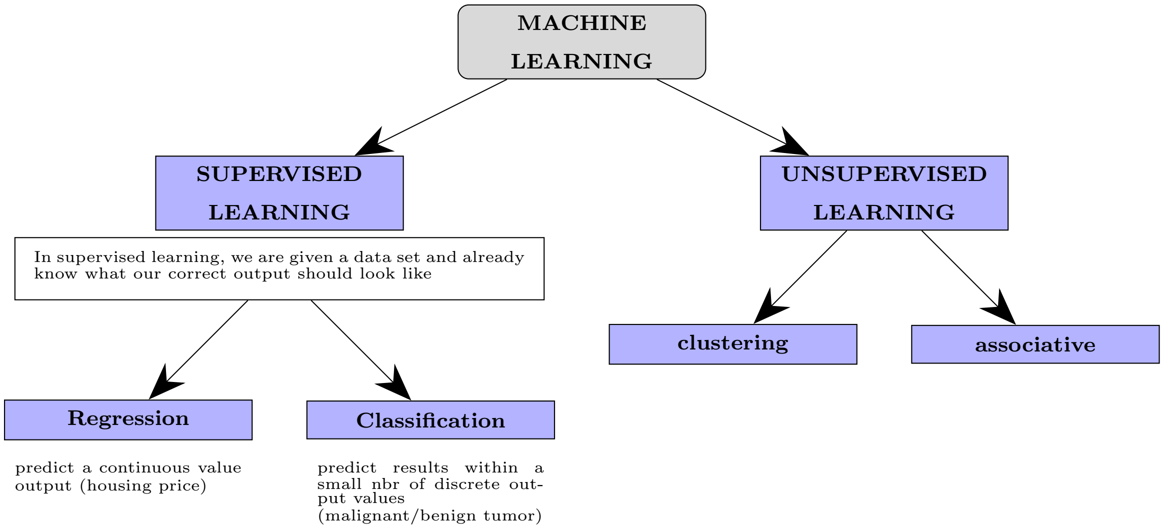
Part I

Week 1 and 2

0.1 Introduction / Definition

0.1.1 What is Machine Learning?

Machine Learning: algorithms for **inferring** unknowns from knowns.



0.1.2 Supervised Learning

- **a regression problem: predict housing price**

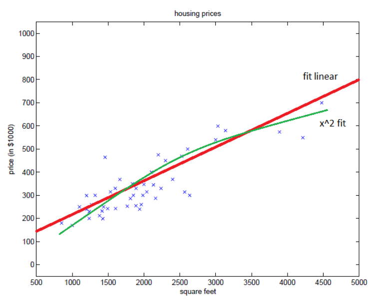


Figure 1: The right answer in this example is the price corresponding to the house size.

- **a classification problem: predict if tumor is malignant or benign.**

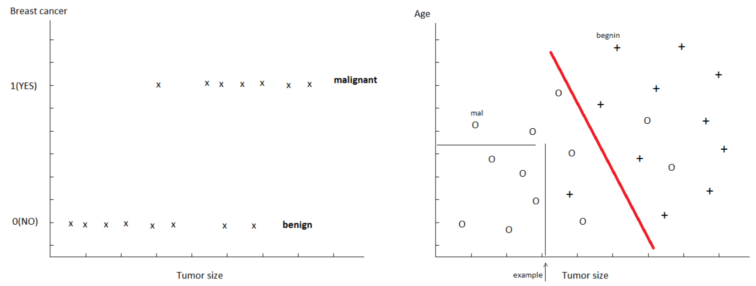


Figure 2: In classification problems, one can have more than 2 values for the output: for example may be 3 types of cancer (0=benign, 1=malignant, 2=type2, 3=type3). The graph on the right shows an alternative way of plotting the data when using several features / attributes. The learning algorithm may decide to separate the malign and benign with a straight line, hence the example of with shown age and tumor size would have a high probability to be a malign tumor

0.1.3 Supervised vs. Unsupervised Learning

- in **supervised** learning, each **example is labeled**.
  - regression
  - classification
- in **unsupervised learning**, there is **no label**: the algorithm might look for some structures in the data, and might separate the data in to 2 clusters (clustering algorithm). Examples where unsupervised Learning algorithm is used: organize computing clusters, social network analysis, market segmentations.
  - clustering
  - density estimation
  - Dimensionality reduction

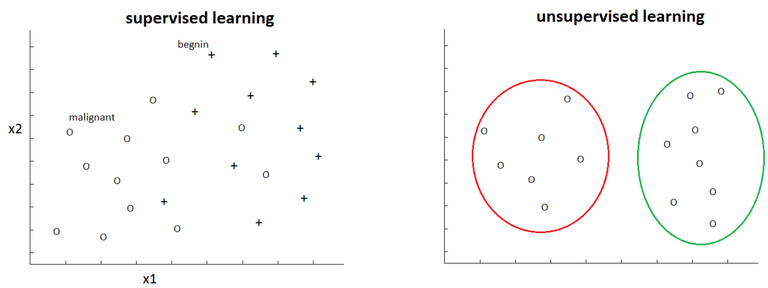


Figure 3: Illustration of supervised and unsupervised learning

There are some variations:

- Semi-supervised learning: mix of labeled and unlabeled dataset
- Active learning
- Decision Theory
- Reinforced learning

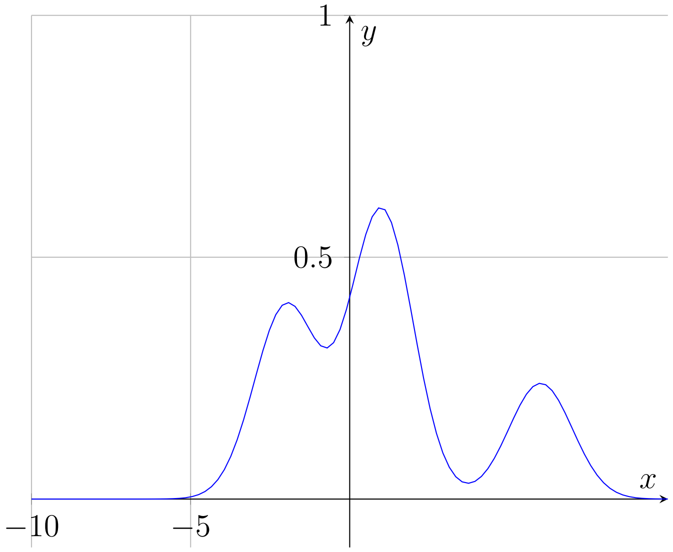


Figure 4: This is an example of density estimation. The x axis has a distribution of data point centered at -2, 1 and 6

0.2 Generative vs. Discriminative models

0.2.1 Discriminative models

Discriminative approach would model the probability $p(y|x)$ i.e probability of y given x.

0.2.2 Generative model

Generative approach models the joint probability distribution i.e $p(x, y)$ (probability of x and y).

$$p(x, y) = f(x|y)p(y) \tag{1}$$

where $f(x|y)$ is the density of x given y, and $p(y)$ ist the probability of y.

0.3 Supervised Learning: Linear regression with 1 variable/feature (aka univariable linear regression)

The Goal is to predict a unique output from a single input value.

1. The hypothesis function $h_\theta(x)$ (or h_θ):

$$h_\theta(x) = \theta_0 + \theta_1 x, \tag{2}$$

   where θ's are the parameters

2. The cost function (or squared error function or Mean Squared Error): measure the accuracy of our hypothesis function.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2, \tag{3}$$

where $\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$ is the difference between the predicted value and the actual value, and m is the size of the training set (data set). $x^{(i)}$ and $y^{(i)}$ are values of examples in the given training set.

3. Gradient Descent method: to automatically improve hypothesis function.

Repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

simultaneous update of all θ's: $j = 0$ and $j = 1$.
}

---

where α is called the learning rate.
The goal is to minimize the cost function i.e get θ_0 and θ_1 value for which the difference between predicted and real value is the smallest.

Gradient descent for linear regression:
Using definition of h_θ and $J(\theta_0, \theta_1)$:

Repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \times x^{(i)}$$

simultaneous update for θ_0 and θ_1
}

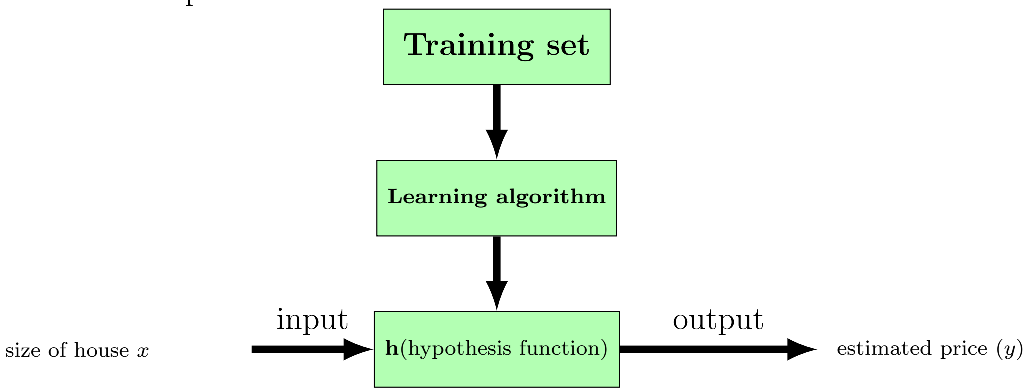
0.3.1 Application to the house pricing example

Provided training set:

	size in ft^2 (x)	Price \$\$ (y)
$(x,y) \rightarrow$	2104 $\leftarrow x^{(1)}$	460 $\leftarrow y^{(1)}$
	1600 $\leftarrow x^{(2)}$	330 $\leftarrow y^{(2)}$
	2400 $\leftarrow x^{(3)}$	369 $\leftarrow y^{(3)}$
	1416 $\leftarrow x^{(4)}$	232 $\leftarrow y^{(4)}$
	3000 $\leftarrow x^{(5)}$	540 $\leftarrow y^{(5)}$
	.	.
	.	.
	.... $\leftarrow x^{(m)}$	... $\leftarrow y^{(m)}$

- x = input variables / features

- m is the total number of training examples
- y = output variable / target variable
- (x, y) = one training example
- $(x^{(i)}, y^{(i)}) = i^{\text{th}}$ training example
- Picture of the process:



In this example, we take a linear representation of h (i.e linear regression with one variable or **univariante linear regression**): $h_\theta(x) = \theta_0 + \theta_1 x$

- Result

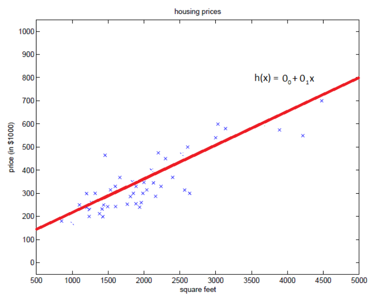


Figure 5: Result: the red line is the best fit from Gradient descent.

0.4 Multiple features

0.4.1 Hypothesis function for n features

For 1 feature, the hypothesis was:

$$
\begin{aligned}
h_\theta(x) &= \theta_0 + \theta x_1 \\
&= \theta_0 x_0 + \theta x_1
\end{aligned}
$$

where $x_0 = 1$ by convention.

Let's now consider the case of 4 features:

x_0	size in ft^2 (x_1)	Nbr of Bdr (x_2)	Nbr of floors (x_3)	Age home (x_4)	Price $$ (y)
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178
1	.	.			
.	.	.			
1	$(x_1^{(m)})$	$(x_2^{(m)})$	$(x_3^{(m)})$	$(x_4^{(m)})$	$(y^{(m)})$

- n is the number of features (4)
- $x_j^{(i)}$ = value of features j in i^{th} training example
- $x^{(2)} = \begin{bmatrix} 1 \\ 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$ is a vector matrix of size 4 ($\mathbb{R}^{4\times 1}$), showing the feature values of training example 2.

With $x_0 = 1$, the hypothesis for n features can take the general form (multivariate linear regression):

$$\begin{aligned} h_\theta(x) &= \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + ...\theta_n x_n \\ &= \sum_{j=0}^{n} \theta_j x_j \\ &= \theta^{\mathrm{T}} X \text{vectorized form} \end{aligned} \tag{4}$$

where:

- $X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \\ 1 & . & . & . & . \\ . & . & . & . & . \\ 1 & (x_1^{(m)}) & (x_2^{(m)}) & (x_3^{(m)}) & (x_4^{(m)}) \end{bmatrix} \in \mathbb{R}^{m\times(n+1)}$
- $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ . \\ . \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n\times 1}$

0.4.2 The Cost function $J(\theta)$

$$\begin{aligned} J(\theta_0, \theta_1, \theta_2, ..., \theta_n) &= J(\theta) \\ &= \frac{1}{2} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \end{aligned}$$

0.4.3 Gradient descent algorithm

Repeat until convergence {

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

(update all θ_j simultaneously: $j = 0$ to n). }

This is called "'batch Gradient Descent"': each step of gradient descent uses **all** the training examples (1 through m). This method is not appropriate (too heavy) when dealing with very high number of features (1+ million). An alternative is "'**Stochastic Gradient Descent**"':

Repeat until convergence { for $i = 1$ to m, {

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

(for every j). }
}

In this algorithm, we repeatedly run through the training set, and each time we encounter a training example, we update the parameters according to the gradient of the error with respect to that single training example only. Often stochastic gradient descent gets θ "'close"' to the minimum much faster than batch gradient descent (note however that it may never converge to the minimum, and θ will keep oscillating around the minimum of $J(\theta)$).
Repeat until convergence {

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

(update all θ_j simultaneously: $j = 0$ to n). }

0.5 Gradient descent in practice

In practice, we stop the iteration when $\frac{\partial J}{\partial \theta} < \epsilon$, where ϵ is a tolerance set by the user.

0.5.1 Feature scaling: how to converge faster

Place all the features at same scale:

1. method 1: features range $\Rightarrow -1 \leq x_j \geq 1$
   eg: x_1 =size (0-2000) and x_2 = Nbr of bdrs (1-5). Use x_1 = (size / 2000) x_2 = (bdr / 5)

2. method 2: mean normalization $\Rightarrow -0.5 \leq x_j \geq 0.5$
   Replace x_j by $(x_j - \mu_j)/s_j$ where μ_j is the mean of x_j^i for all i's (i.e over all training examples), and s_j is the range, either (max-min) or stdev.

0.5.2 Learning rate α: how to converge faster

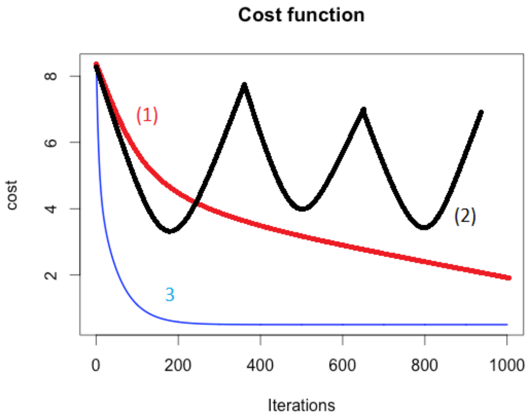


Figure 6: If gradient descent works, $J(\theta)$ must decrease after each iteration (3). If α is too small, Gradient Descent will slowly converge (1). But if α too large, Gradient descent woudl not converge ((case 2) or $J(\theta)$ increases)

A good approach is to try a series of α values: 0.001, 0.01, 0.1, 1 and choose α that gives fastest converging gradient descent. One can also try to reduce the step size as the number of iterations increases:

$$\alpha = \frac{\alpha_0}{t} \tag{5}$$

where t is the iteration

0.5.3 Polynomial regression

- Model exple 1: $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$
  How to convert it in to hypothesis: use x_1 =size, $x_2 = (\text{size})^2$, $x_3 = (\text{size})^3$. then, $h_\theta(x)$ becomes:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

- Model exple 2: $\theta_0 + \theta_1 x + \theta_2 \sqrt(x)$
  How to convert it in to hypothesis: use x_1 =size, $x_2 = (\text{size})^0.5$, then, $h_\theta(x)$ becomes:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

0.6 Normal Equations

Normal equations provides an analytical solution to $\partial J(\theta)/\partial\theta = 0$.

$$\nabla_\theta J = \begin{bmatrix} \partial J/\partial\theta_0 \\ \partial J/\partial\theta_1 \\ . \\ . \\ . \\ \partial J/\partial\theta_n \end{bmatrix} \in \mathbb{R}^{(n+1)}$$

Rewrite gradient descent:

$$\underbrace{\theta}_{\in \mathbb{R}^{(n+1)}} := \theta - \alpha \underbrace{\nabla_\theta J}_{\in \mathbb{R}^{(n+1)}}$$

Design matrix X contains all the inputs from training set:

$$X = \begin{bmatrix} --(x^{(1)})^{\mathrm{T}}-- \\ --(x^{(2)})^{\mathrm{T}}-- \\ -- \\ . \\ . \\ --(x^{(m)})^{\mathrm{T}}-- \end{bmatrix} = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & x_2^{(1)} & \ldots\ldots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & x_2^{(2)} & \ldots\ldots & x_n^{(2)} \\ . & . & . & \ldots\ldots & \\ . & . & . & \ldots\ldots & \\ . & . & . & \ldots\ldots & \\ x_0^{(m)} & x_1^{(m)} & x_2^{(m)} & \ldots\ldots & x_n^{(m)} \end{bmatrix}$$

is a $(m \times (n+1))$ matrix with $X^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ . \\ . \\ . \\ x_n^{(i)} \end{bmatrix}$

$$X \times \theta = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & x_2^{(1)} & \ldots\ldots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & x_2^{(2)} & \ldots\ldots & x_n^{(2)} \\ . & . & . & \ldots\ldots & \\ . & . & . & \ldots\ldots & \\ . & . & . & \ldots\ldots & \\ x_0^{(m)} & x_1^{(m)} & x_2^{(m)} & \ldots\ldots & x_n^{(m)} \end{bmatrix} \times \begin{bmatrix} \theta_0 \\ \theta_1 \\ . \\ . \\ . \\ \theta_n \end{bmatrix} = \begin{bmatrix} (x^{(1)})^{\mathrm{T}}\ \theta \\ (x^{(2)})^{\mathrm{T}}\ \theta \\ (x^{(3)})^{\mathrm{T}}\ \theta \\ \ldots \\ \ldots \\ \ldots \\ (x^{(m)})^{\mathrm{T}}\ \theta \end{bmatrix} = \begin{bmatrix} h_\theta(x^{(1)}) \\ h_\theta(x^{(2)}) \\ h_\theta(x^{(3)}) \\ \ldots \\ \ldots \\ \ldots \\ h_\theta(x^{(m)}) \end{bmatrix}$$

$$\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \ldots \\ \ldots \\ \ldots \\ y^{(m)} \end{bmatrix} \in \mathbb{R}^{(m \times 1)}$$

and :

$$X\theta - \vec{y} = \begin{bmatrix} h(x^{(1)}) - y^{(1)} \\ h(x^{(2)}) - y^{(2)} \\ h(x^{(3)}) - y^{(3)} \\ \ldots \\ \ldots \\ \ldots \\ h(x^{(m)}) - y^{(m)} \end{bmatrix}$$

For a vector z, $z^{\mathrm{T}}.z = \sum z_i^2$, hence:

$$\frac{1}{2}\left(X\theta - y\right)^{\mathrm{T}}\left(X\theta - y\right) = \frac{1}{2}\sum_{i=1}^{m}\left(h(x^i) - y^i\right)^2 = J(\theta)$$

Minimize $J(\theta)$ for every j:

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

$J(\theta) \in \mathbb{R}$ so $\mathrm{tr}(J(\theta)) = J(\theta) \Rightarrow \nabla_\theta(J(\theta)) = \nabla_\theta \mathrm{tr}(J(\theta))$.

$$
\begin{aligned}
\frac{1}{2}\nabla_\theta \mathrm{tr}\left(\theta^{\mathrm{T}}X^{\mathrm{T}}X\theta - \theta^{\mathrm{T}}X^{\mathrm{T}}y - y^{\mathrm{T}}X\theta + y^{\mathrm{T}}y\right) &= 0 \\
\frac{1}{2}\nabla_\theta \mathrm{tr}\left(\theta^{\mathrm{T}}X^{\mathrm{T}}X\theta\right) - \frac{1}{2}\nabla_\theta \mathrm{tr}\left(\theta^{\mathrm{T}}X^{\mathrm{T}}y\right) - \frac{1}{2}\nabla_\theta \mathrm{tr}\left(y^{\mathrm{T}}X\theta\right) + \frac{1}{2}\nabla_\theta \mathrm{tr}\left(y^{\mathrm{T}}y\right) &= 0
\end{aligned}
$$

$y^{\mathrm{T}}y$ does not depend on θ so $\nabla_\theta y^{\mathrm{T}}y = 0$:

$$
\frac{1}{2}\nabla_\theta \mathrm{tr}\left(\theta^{\mathrm{T}}X^{\mathrm{T}}X\theta\right) - \frac{1}{2}\nabla_\theta \mathrm{tr}\left(\theta^{\mathrm{T}}X^{\mathrm{T}}y\right) - \frac{1}{2}\nabla_\theta \mathrm{tr}\left(y^{\mathrm{T}}X\theta\right) = 0
$$

Note that $\theta^{\mathrm{T}}X^{\mathrm{T}}y \in \mathbb{R}$ (If $z \in \mathbb{R}$, then $z^{\mathrm{T}} = z$), so: $(\theta^{\mathrm{T}}X^{\mathrm{T}}y)^{\mathrm{T}} = y^{\mathrm{T}}X\theta$

$$
\frac{1}{2}\nabla_\theta \mathrm{tr}\left(\theta^{\mathrm{T}}X^{\mathrm{T}}X\theta\right) - \nabla_\theta \mathrm{tr}\left(y^{\mathrm{T}}X\theta\right) = 0
$$

By the property of permutation: $\theta^{\mathrm{T}}X^{\mathrm{T}}X\theta = \theta\theta^{\mathrm{T}}X^{\mathrm{T}}X$, and:

$$
\nabla_\theta \mathrm{tr}(\theta\theta^{\mathrm{T}}X^{\mathrm{T}}X) = \nabla_\theta \mathrm{tr}(\underbrace{\theta}_{A}\underbrace{I}_{B}\underbrace{\theta^{\mathrm{T}}}_{A^{\mathrm{T}}}\underbrace{X^{\mathrm{T}}X}_{C}) = \underbrace{X^{\mathrm{T}}X}_{C}\underbrace{\theta}_{A}\underbrace{I}_{B} + \underbrace{X^{\mathrm{T}}X}_{C^{\mathrm{T}}}\underbrace{\theta}_{A}\underbrace{I}_{B^{\mathrm{T}}}
$$

$$
\nabla_\theta \mathrm{tr}\,\underbrace{y^{\mathrm{T}}X}_{B}\underbrace{\theta}_{A} = \underbrace{X^{\mathrm{T}}y}_{B^{\mathrm{T}}}
$$

Hence:

$$
\nabla_\theta J(\theta) = \frac{1}{2}\left(X^{\mathrm{T}}X\theta + X^{\mathrm{T}}X\theta\ right) - X^{\mathrm{T}}y\right) = 0
$$

Finally:

$$
\begin{aligned}
X^{\mathrm{T}}X\theta &= X^{\mathrm{T}}y = 0 \\
\theta &= \left(X^{\mathrm{T}}X\right)^{-1}X^{\mathrm{T}}y
\end{aligned}
$$

Pros and Cons of Gradient Descent and Normal Equations:

Gradient Descent	Normal Equations
Need to choose α Need to use many iterations works well even with large Nbrs of features	No need to choose α No iteration slow for very large nbrs of features Need to compute $(XX^{\mathrm{T}})^{-1}$: cost of inverting matrix is $\mathrm{O}(n^3)$ for a $n \times n$ matrix

Appendices

Appendix A

Linear Algebra Review

A.1 1-indexed versus 0-indexed vector matrix

$$Y(1-\text{indexed}) = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \text{and} Y(0-\text{indexed}) = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix},$$

A.2 Matrix

A.2.1 Matrix size

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \\ j & k & l \end{bmatrix},$$

is a 4(rows) $\times$ 3(cols) matrix ($\mathbb{R}^{3\times2}$ matrix). Length = # Rows $\times$ #Cols
A_{ij} corresponds to value at row i^{th} and col j^{th}

$$A = \begin{bmatrix} a \\ d \\ g \\ j \end{bmatrix},$$

is a **vector** matrix ($n \times 1$)

A.2.2 Matrix Operation

Matrix addition

$$A + B = \begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} w & x \\ y & z \end{bmatrix} = \begin{bmatrix} (a+w) & (b+x) \\ (c+y) & (d+z) \end{bmatrix},$$

Matrix scalar multiplication

$$A \times x = x \times A = x \times \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} (a \times x) & (b \times x) \\ (c \times x) & (d \times x) \end{bmatrix},$$

Multiplication of 2 matrix

$A \times B$ requires that the # of rows of A be equal to # of cols of B.
Matrix $\times$ a vector = a vector $(n \times m)$ matrix $\times$ $(m \times 1)$ vector = $n \times 1$ vector

$$A \times B = \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} (ax+by) \\ (cx+dy) \\ (ex+fy) \end{bmatrix}$$

Matrix multiplication properties:

- Not commutative: $A \times B \neq B \times A$
- associative : $A \times B \times C = (A \times B) \times C = A \times (B \times C)$

Matrix Identity

$A \times I = A$, where I is a matrix identity:

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix Inverse

Inverse of A is denoted A^{-1}, and :

$$A \times A^{-1} = I$$

A non-square matrix does not have an inverse.

Transpose matrix A^{T}

$$A = \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix}$$

$$A^{\mathrm{T}} = \begin{bmatrix} a & c & e \\ b & d & f \end{bmatrix}$$

so $A_{ij} = A^T_{ji}$

A.2.3 Derivative $\nabla_A f(A)$

For a function f: $\mathbb{R}^{(m \times n)} \to \mathbb{R}$ mapping from m-by-n matrix to the real numbers, we define the derivative of f with respect to A to be:

$$\nabla_A f(A) = \begin{bmatrix} \partial f/\partial A_{11} & \partial f/\partial A_{12} & ...... & \partial f/\partial A_{1n} \\ .. & .. & .. & .. \\ .. & .. & .. & .. \\ .. & .. & .. & .. \\ \partial f/\partial A_{m1} & \partial f/\partial A_{m2} & ...... & \partial f/\partial A_{mn} \end{bmatrix}$$

so $\nabla_A f(A)$ is itself a $\mathbb{R}^{(m\times n)}$ whose (i,j) elements is $\partial f/\partial A_{ij}$.

For example: $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ is a 2×2 matrix, and $f : \mathbb{R}^{(2\times 2)} \to \mathbb{R}$ is: $f(A) = \frac{3}{2}A_{11} + 5A_{12}^2 + A_{21}A_{22}$. Then $\nabla_A f(A) = \begin{bmatrix} 3/2 & 10A_{12} \\ A_{22} & A_{21} \end{bmatrix}$

A.2.4 Trace

$$\mathrm{tr}A = \sum_{i=1}^{n} A_{ii} \text{ ifA} \in \mathbb{R}^{n\times n}$$

$$\mathrm{tr}(A\times B) = \mathrm{tr}(B\times A)$$

$$\mathrm{tr}(ABC) = \mathrm{tr}(CAB) = \mathrm{tr}(BCA)$$

$$\mathrm{tr}(ABCD) = \mathrm{tr}(DABC) = \mathrm{tr}(CDAB)$$

If A and B are square matrix:

$$\mathrm{tr}A = \mathrm{tr}A^{\mathrm{T}}$$

$$\mathrm{tr}(A+B) = \mathrm{tr}A + \mathrm{tr}B$$

If $a \in \mathbb{R}$:

$$\mathrm{tr}(a) = a$$

If $f(A) = \mathrm{tr}(AB)$

$$\nabla_A \mathrm{tr}(AB) = B^{\mathrm{T}}$$

$$\nabla_A \mathrm{tr}(ABA^{\mathrm{T}}C) = CAB + C^{\mathrm{T}}AB^{\mathrm{T}}$$

Appendix B

Matlab

- disp(a) /*show the matrix
- a=3.14567
  disp(sprintf('2 decimals: %0.2f', a))
- define a row vector: v=[1 2 3] $\rightarrow$ v[1,2,3]
- define a col vector: v=[1; 2; 3] $\rightarrow a = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$
- define a matrix (3×2): a=[1 2; 3 4; 5 6] $\rightarrow a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$
- eye(2) gives a 2×2 Identity matrix $\rightarrow a = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
- zeros(2) gives a 2×2 matrix with only elements 0 $\rightarrow a = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$
- ones(2) gives a 2×2 matrix with only elements 1 $\rightarrow a = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$
- a matrix (3×2): a=[1 2; 3 4; 5 6] $\rightarrow a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$
  sz = size(A) $\rightarrow$ [3,2]
  size(A,1) $\rightarrow$ nbrs of rows (3)
  size(A,2) $\rightarrow$ nbrs of cos (2)
  For a vector A use length(A)
- pwd $\rightarrow$ current directory
- load(filename)
- who $\rightarrow$ show what variables are in memory
- whos $\rightarrow$ show list of variables (with size, type) in memory
- clear $\rightarrow$ remove all variables from memory
- save hello.txt v -ascii $\rightarrow$ save v as ascii
- save hello.m v $\rightarrow$ save v as matlab file
- Elements of $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$
  A(3,2) $\rightarrow$ 6
  A(2, :) $\rightarrow$ return row # 2 : (3 4)
  A(:, 2) $\rightarrow$ return col # 2 : $\begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}$

- pinv(A) $\rightarrow A^{-1}$
- $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$ and $B = \begin{bmatrix} 11 & 12 \\ 13 & 14 \\ 15 & 16 \end{bmatrix}$
  $A. * B \rightarrow \begin{bmatrix} 11 & 24 \\ 39 & 56 \\ 75 & 96 \end{bmatrix}$
  (.*) element-wise multiplication of 2 matrix

Appendix C

Exercise: Gradient Descent Matlab

```
data = load( 'data.txt' );
X = data( : , 1:2 );
y = data( : , 3 );
m = length( y );
# scale features
mu = zeros( 1, size( X, 2 ) );
sigma = zeros( 1, size( X, 2 ) );
X_norm = X;

mu = mu * mean( X )
sigma = sigma * std( X )

X_norm = ( X - mu ) ./sigma;
X_norm = [ ones( m, 1 ), X_norm ]  #add intercept term to X

# Gradient descent
alpha = 0.01;
num_iters = 400;

# Initialize theta and J
theta = zeros( 3, 1);
J = 0;
J_history = zeros( num_iters, 1);

for iter = 1:num_iters
        theta = theta - alpha/m * X' * ( X * theta - y );
  # Compute Cost'
        prediction = X * theta;
        sqErr = ( prediction - y ).^2 ;
        J = 1/(2 * m) sum( sqErr );
        J_history ( iter ) = J;

plot( J_history );
xlabel( 'Nb of iterations' );
ylabel( 'Cost J' );
```