Contents

1 Week 1 **1**
- 1.1 Intro . . . 1
- 1.2 Supervised Learning . . . 2
- 1.3 Unsupervised Learning . . . 2
- 1.4 Model and Cost Function . . . 3
- 1.5 Gradient Descent Algorithm . . . 4
  - 1.5.1 intuition . . . 4
  - 1.5.2 pitfalls . . . 4
- 1.6 Putting it all together . . . 5
  - 1.6.1 calculating the partial derivative . . . 5
- 1.7 Linear Algebra Review . . . 6
  - 1.7.1 Addition and Scalar Multiplication . . . 6
  - 1.7.2 Matrix Vector Multiplication . . . 7
  - 1.7.3 Matrix Matrix Multiplication . . . 7
  - 1.7.4 Matrix Multiplication Properties . . . 7
  - 1.7.5 Matrix inverse and matrix transpose operation . . . 7

2 Week 2 **7**
- 2.1 Multivariant Linear Regression . . . 7
  - 2.1.1 Notation . . . 7
  - 2.1.2 Hypothesis . . . 8
  - 2.1.3 Gradient descent for multiple variables . . . 8
  - 2.1.4 Feature scaling & Mean normalization . . . 8
  - 2.1.5 Learning Rate . . . 9
  - 2.1.6 Features & Polynomial Regression . . . 9
- 2.2 Computing Parameters Analytics . . . 10
  - 2.2.1 Normal Equation . . . 10

3 @todo **11**

1 Week 1

1.1 Intro

A computer is said to learn from experience E with respect to task T, and some performance measure P, if its performance on T as measured by P improves with experience E.

1.2 Supervised Learning

- Example problem given a dataset of houses where price is on the X axis and sq. ft is on the Y axis, predict the price for a given square footage.
- You input is a data set with the "right" answers. The task is to produce more of the "right" answers. Since you are trying to predict a continuous value this is called a regression problem.
- Knowing the best function to fit to your data is something you gain intuition for?
- Classification problem: discrete output (0, 1, 2)
- Regression problem: continuous output (0 - 2)
- Support vector machines are a thing and they allow a computer to deal with an infinite number of features.

1.3 Unsupervised Learning

- We are given data that has few or no labels and the goal is to find some structure in the data
- Google News is an example of this clustering by un-labeled topic, you don't know what the stories will be about in advanced
- You don't know in advance any of the labels and you want the machine to label things for you
- analyzing communications between computers in data center and optimizing topology for clusters that frequency work together
- applications of unsupervised learning:
  - clustering
  - separating voices out at a cocktail party ("cocktail party" algorithm) e.g. signal correlation

1.4 Model and Cost Function

Notation:

- input variables are often called features
- (x,y) is a single training example
- (x^i, y^i) the ith training example
- the function that a learning algorithm produces is often called a hypothesis
- training set -> learning algorithm -> h()
  - h is a function that maps from x's to y's
- Linear regression with one variable AKA Univarite linear regression

Hypothesis $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters θ_0, θ_{01}

Cost function $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

Goal minimize $J(\theta_0, \theta_1)$

- This cost function is called the "squared error" or "mean squared error" function and works well for a lot of linear regression problems
- You can get better intuition around the goal by simplifying to one parameter θ_1 graphing $h_\theta(x)$ with some inputs you'll notice that the corresponding cost function graph for θ_1 is a parabola converging on $\theta_1 = 1$.
- The best possible line will be such so that the average squared vertical distances of the scattered points from the line will be the least. Ideally, the line should pass through all the points of our training data set. In such a case, the value of J(0,1) will be 0
- two parameters can be visualized on a 3D graph or in two dimensions via a contour plot. We put 0 on the x axis and 1 on the y axis, with the cost function on the vertical z axis.

1.5 Gradient Descent Algorithm

- the gradient descent algorithm can help minimize functions and is used in many machine learning techniques in our case this is $\overset{min}{\theta_0, \theta_1} J(\theta_0, \theta_1)$
- start with some values for \(θ_0, θ_1)\ (zero is common default value)
- keep changing $\theta_0, \theta_1)$ $to reduce J(\theta_0, \theta_1)$ until you end up at a minimum (hopefully)

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

assignment operator :=

learning rate α, controls how big of a step you take on the descent

derivative term $\alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$

- *θ_0 and θ_1 should be updated simultaneously as they have inputs dependant on one another
- at the local optimum your derivative is zero so θ_1 is unchanged
- with a fixed learning rate α each successive step is smaller because the derivative slope value is smaller

1.5.1 intuition

simplify to 1 feature $\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_j} J(\theta_1)$

graph a convex function and pick a point on that line

the derivative is the slope of the line that is tagent to the function

if the point we picked has a positive slope we get $\theta_1 := \theta_1 - \alpha(positivenumber)$ which helps us move closer to the minimum

if the point we picked has a negative slope we get $\theta_1 := \theta_1 - \alpha(negativenumber)$ which helps us move closer to the minimum

1.5.2 pitfalls

- if α is too small than gradient descent can be slow
- if α is too big than you can overshoot the minimum and not converge

1.6 Putting it all together

- we find the best linear fit to the data by finding the slope and offset values that minimize the cost function
- the cost function for linear regression is always going to be a bow-shaped or "convex" function, so there is only one global optimum (no local optimum)
- "batch" gradient descent means we are using all the training examples in every step of gradient descent (each update sum's examples)
- there are other versions of gradient descent that window data sets

linear regression model \(h_\theta(x)=\theta_0+\theta_1 x

cost function $J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$

gradient descent algorithm $\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1)(for j = 1 and j = 0)$

- we find the best linear fit to the data by finding the slope and offset values that minimize the cost function

1.6.1 calculating the partial derivative

- expand equation
  - $\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1) = \frac{\partial}{\partial\theta_j} * \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$
  - $\frac{\partial}{\partial\theta_j} * \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 = \frac{\partial}{\partial\theta_j} * \frac{1}{2m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2$
- figure out partial derivative for $j = 0$ and $j = 1$
  - $j = 0 : \frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})$ \)
  - $j = 1 : \frac{\partial}{\partial\theta_1}J(\theta_0,\theta_1) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) * x^{(i)}$ \)
  - @todo calculate these derivations yourself
- plug into gradient descent (repeat until convergence, simultaneously update assignments)

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

1.7 Linear Algebra Review

matrix rectangular array of values, dimension is determined by rows x columns

matrices of a specific dimension containing real number are often notated like this: R^{4x2}

to refer to specific elements matrix A you can use the notation A_{ij} to meant the ith row, jth column

vector a nx1 matrix, notated as $\mathbb{R}^4$, values accessed as A_i, a 4-dimensional vector has 4 elements

in mathematical notation 1-indexed notation is preferred

lowercase variables often denote vectors and uppercase variables denote matrices

1.7.1 Addition and Scalar Multiplication

- matrices of the same dimensions add in a straightforward way
  - $\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 6 & 8 \\ 10 & 12 \end{bmatrix}$
- matrices of different dimensions cannot be added
  - $\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = undefined$
- \(3·[1&2\ 3 & 4] produces a matrix of the same dimension where each value is the product of the previous value and 3
- scalar division works in a similar way

1.7.2 Matrix Vector Multiplication

- multiplying a m x n matrix with a vector is done calculating the dot product of each matrix row with the vector, yeilding a m-dimensional vector

1.7.3 Matrix Matrix Multiplication

- for matrix-matrix multiplication you break the second term into vectors and then combine the columns at the end. a MxN matrix multipled by a NxO matrix yields a MxO matrix. The N must match in order to be able to multiply them
- The ith column of matrix C is obtained by multiplying A with the ith column of B
- To multiply two matrices, the number of columns of the first matrix must equal the number of rows of the second matrix.

1.7.4 Matrix Multiplication Properties

- regular multiplication is commutative, matrix multiplication is not (A x B is not equal to B x A)
- regular multiplication is associative and so is matrix multiplication
- for the identity matrix AB is equal to AB

1.7.5 Matrix inverse and matrix transpose operation

- if A is an mxm (square) matrix and has an inverse: $AA^{-1} = A^{-1}A = I$
- some mxm matrices don't have an inverse (e.g. all values are zero)
- singular or degenerate matrices are matrices that don't have an inverse

2 Week 2

2.1 Multivariant Linear Regression

2.1.1 Notation

n number of features

m number of training examples

$x^{(i)}$ input of the i^{th} training example

$x_j^{(i)}$ value of feature j in i^{th} training example

2.1.2 Hypothesis

- previously with univariant linear regression our hypothesis was $h_\theta(x) = \theta_0 + \theta_1 x$
- for multivariant we have a polynomial hypothesis $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n$
- if you define $x_0^{(i)} = 1$ than you can normalize the above as $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n$ express in matrix multiplication as $h_\theta(x) = \theta^T x$
- $\theta^T x = \begin{bmatrix}\theta_0 & \theta_1 & ... & \theta_n\end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ ... \\ x_n \end{bmatrix} = h_\theta(x)$

2.1.3 Gradient descent for multiple variables

- the cost function is again $1_{\overline{2m}\sum}^{\mathrm{m}}{}_{\mathrm{i=1}}(\mathrm{h}_\theta(\mathrm{x}^{(\mathrm{i})})\text{-}\mathrm{y}^{(\mathrm{i})})^2\backslash)$ but $h_\theta(x^{(i)}) = \theta^T(x^i)$
- in the gradient descent formula the x in the derivative term has to be altered slightly to account for more than one feature $\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) * x_j^{(i)}$
- @todo understand this more

2.1.4 Feature scaling & Mean normalization

why? if you ensure your features have similar ranges of values than gradient descent with converge quicker

conventionally things are usually scaled $-1 \leq 0 \leq 1$ but having a similar range $max - min$ across features is most important, not the specific high and low value

feature scaling is dividing the input values by the range to get a range of 1

mean normalization is a technique for getting the average value of 0 for a feature

combining mean normalization & feature scaling $x_i := \frac{x_i - \mu_i}{s_i}$ where μ_i is average value of x in training set and s_i is the range $max - min$ (or standard deviation)

@todo what is standard deviation

2.1.5 Learning Rate

- to make sure gradient descent is working correctly you can plot the the cost function over the number iterations and the cost function should go down as iterations go up
- you can also automate the above by defining a threshold in code but that can be tricky
- if your cost function value is increasing over the number of iterations or has a scalloped pattern than you learning rate is probably too large
- to choose a proper α value plot $J(\theta)$ with different successively larger learning rates $.001, .003, .01, .03, .1, .3, 1, 3$

2.1.6 Features & Polynomial Regression

housing prices prediction $h_\theta(x) = \theta_0 + \theta_1 \cdot frontage + \theta_2 \cdot depth$

instead of working with the simple features you have, you free to create your own compound features (e.g. $area$ is a compound feature of $frontage$)

housing prices prediction with compound feature $h_\theta(x) = \theta_0 + \theta_1 area$

say we wanted had a hypothesis that a cubic function was a good fit to our data $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$, the machinery of linear regression still works for this $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 = \theta_0 + \theta_1 (size) + \theta_2 (size)^2 + \theta_3 (size)^3$

some algorithms can solve the `what features do i use?` question by automatically choosing what features to use and what functions you want to fit to that data

2.2 Computing Parameters Analytics

2.2.1 Normal Equation

- solves for optimal values of the parameters to ($\theta\backslash$) in one step for some linear regression problems
- there is no need to do feature scaling with the normal equation
- $\theta = (X^T X)^{-1} X^T y$ where $(X^T X)^{-1}$ is the inverse of the matrix $X^T X$ gives you the optimal value for θ

Gradient Descent	Normal Equation
needs to choose α	no need to choose α
needs many iterations	don't need to iterate
scales with large set of features	doesn't scale with large set of features (inverting a very large +10,0
	less versitile (doesn't work with logisitic regression)

1. Intuition
   - in one-dimension given a scalar value for θ, and the quadratic function $J(\theta) = a\theta^2 + b\theta + c$
   - to minimize, take the derivative and set to 0, $\frac{\partial}{\partial\theta} J(\theta) = ... = 0$ and solve for θ
   - in the multivariate version if you take the partial derivative of J with respect to every parameter if θ_j and set that to 0 and solve for $\theta_0, ..., \theta n$ than you can get all the values to minimize the cost function
2. computing this for Singular / Degenerate Matrices
   - If $X^T X$ is noninvertible, the common causes might be:
     - Redundant features, where two features are very closely related (i.e. they are linearly dependent)
     - Too many features (e.g. m n). In this case, delete some features or use "regularization" (to be explained in a later lesson).

   Solutions to the above problems include deleting a feature that is linearly dependent with another or deleting one or more features when there are too many features.

   - @todo what types of matrices are non-invertible?

3 @todo

- replicate Solvvy,
- `https://github.etsycorp.com/Engineering/Etsyweb/commit/0c6754d73fbae3b60310b2fdd1e`
- come up with basic ML project ideas
  - "cocktail party" algorithm is sweet, separates audio without any complicated audio processing libraries
- read more about mean squared error function