# Linear Regression

Jeremy Irvin and
Daniel Spokoyny

Created from Andrew Ng's Stanford CS229 Notes, MIT Linear Algebra Lecture Video 16

# Supervised vs. Unsupervised

- The ultimate goal of a machine learning algorithm is to allow a machine to learn from data and make predictions/ inferences from that data automatically (without hand-made rules).

- There are two main different types of learning algorithms.

- Unsupervised learning algorithms learn from *unlabeled* data, whereas supervised learning algorithms learn from *labeled* data.

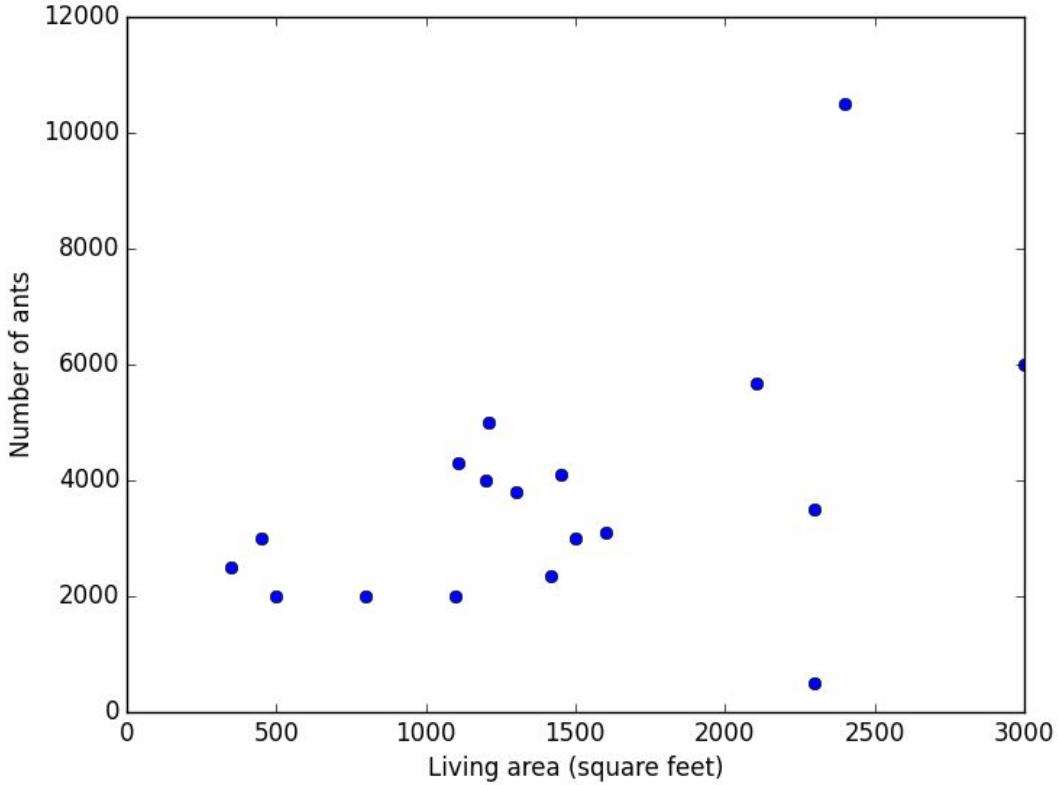
# Supervised Learning Example (Linear Regression)

- Suppose we are given some data from Isla Vista residences:

| Living area ($\text{feet}^2$) | Number of ants |
|---|---|
| 2104 | 5678 |
| 1600 | 100 |
| 2400 | 10500 |
| 1416 | 234 |
| 3000 | 50000 |
| ⋮ | ⋮ |

# Supervised Learning

- We can plot this data:




- We want to predict the number of ants in other residences from the size of their living areas.

# Supervised Learning

- Maybe we have more relevant features in the data to help us predict:

| Living area (feet$^2$) | year built | Number of residents | Number of ants |
|---|---|---|---|
| 2104 | 1950 | 4 | 5678 |
| 1600 | 1975 | 2 | 100 |
| 2400 | 50 | 15 | 10500 |
| 1416 | 1915 | 5 | 234 |
| 3000 | 2010 | 3 | 50000 |
| ⋮ | ⋮ | ⋮ | ⋮ |

# Supervised Learning Notation

- $x^{(i)}$ will denote the "input" variables, called input features (living area, year built, number of residents in our example).

- $y^{(i)}$ will denote the "output" variable, or target variable that we are trying to predict (the number of ants).

- $(x^{(i)}, y^{(i)})$ will denote a training example.

- $\{(x^{(i)}, y^{(i)}) | i = 1, ..., m\}$ will denote a training set.

# Supervised Learning Notation

- $\mathcal{X}$ will denote the space input values and $\mathcal{Y}$ will denote the space of output values.

- We want to learn a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ so that *h(x)* is a good predictor of the corresponding value of *y*.

- *h* is called the hypothesis.

- When the target variable is continuous, the learning problem is called regression. If it is discrete, it is called classification.

# Linear Regression

- In linear regression, we want to find a *best* fit line to our data.

- In our example, we restrict *h* to functions of the form:

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

- The $\theta_i' s$ are the parameters (also called weights).

- We want to choose the $\theta_i' s$ so that *h* is the *best* line.

# Linear Regression

- We can generalize this to arbitrary ($n$) numbers of features, and write (letting $x_0 = 1$):

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$$

- So what does *best* fit line mean?

- We define the cost function $J$ which measures how close the $h(x^{(i)})$'s are to the $y^{(i)}$'s

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

# Gradient Descent

- We want to choose $\theta$ to minimize the error $J(\theta)$.
- Calculus? We will see this later.
- What we can do is use gradient descent, we update $\theta$ by repeatedly taking steps in the steepest decrease of $J$, ie, the opposite direction of the gradient.

# Gradient Descent

- Specifically, we want to perform the update

$$\theta := \theta - \alpha \nabla J(\theta)$$

- Componentwise, for *j=0,...,n*,

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

- $\alpha$ is called the learning rate.

# Gradient Descent

- So what is $\frac{\partial}{\partial \theta_j} J(\theta)$ ? Let's compute it when we only have one training example *(x,y)*:

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \left( h_\theta(x) - y \right)^2 \\
&= 2 \cdot \frac{1}{2} \left( h_\theta(x) - y \right) \cdot \frac{\partial}{\partial \theta_j} \left( h_\theta(x) - y \right) \\
&= \left( h_\theta(x) - y \right) \cdot \frac{\partial}{\partial \theta_j} \left( \sum_{i=0}^{n} \theta_i x_i - y \right) \\
&= \left( h_\theta(x) - y \right) x_j
\end{aligned}
$$

# Gradient Descent

- This gives the update rule:

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

for each individual training example $(x^{(i)}, y^{(i)}), i = 1, ..., m$.

- This is the “least mean squares” (LMS) update rule.

- We can iterate over the examples in our training set and update every time until *convergence* - this is called stochastic gradient descent.

# Gradient Descent

- We could also perform the following update rule until convergence:

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$$

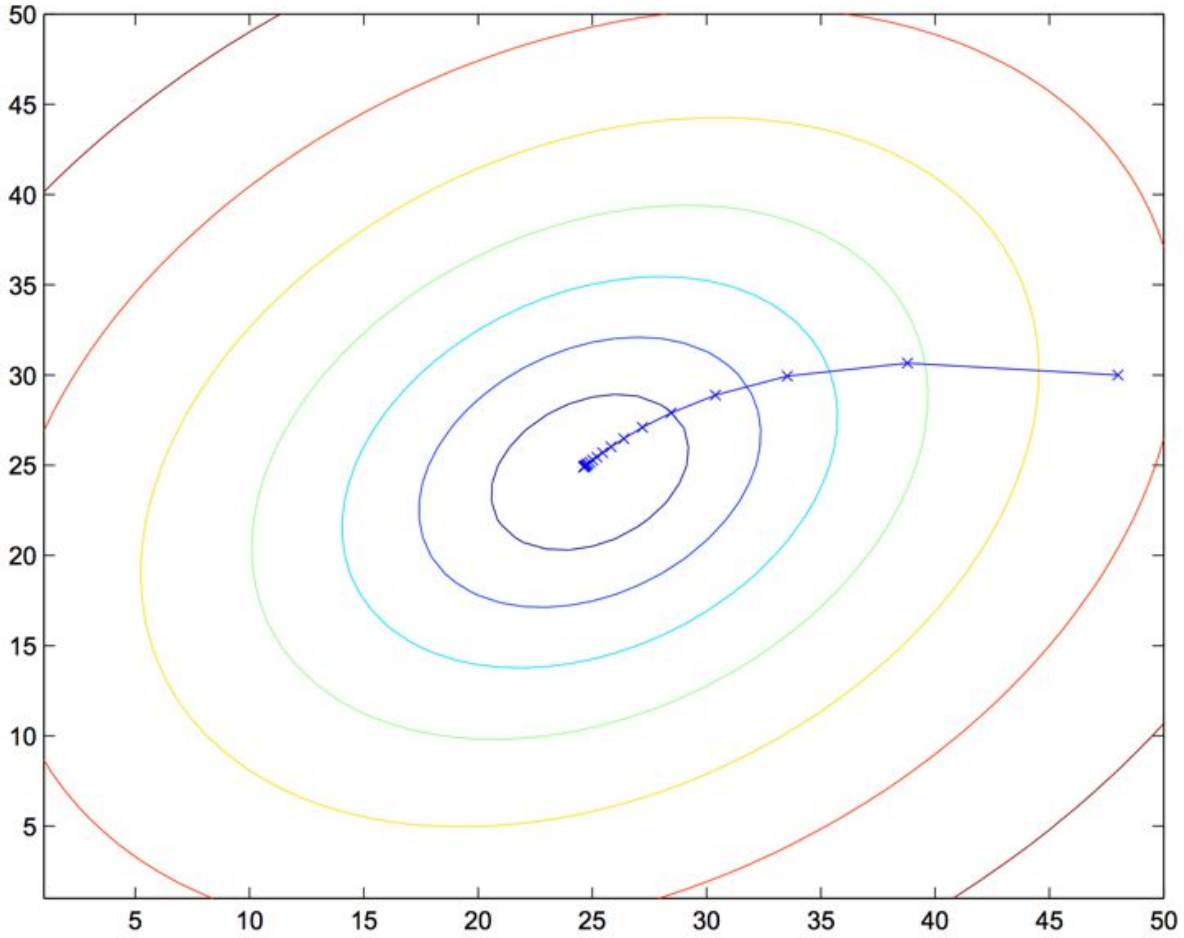
- The right term in the sum is just $\frac{\partial J(\theta)}{\partial \theta_j}$ for the original *J* with all training examples.

- This algorithm is known as batch gradient descent.

- *J* is a convex function, so batch gradient descent 'always' converges (approximately) to the *global* minimum.

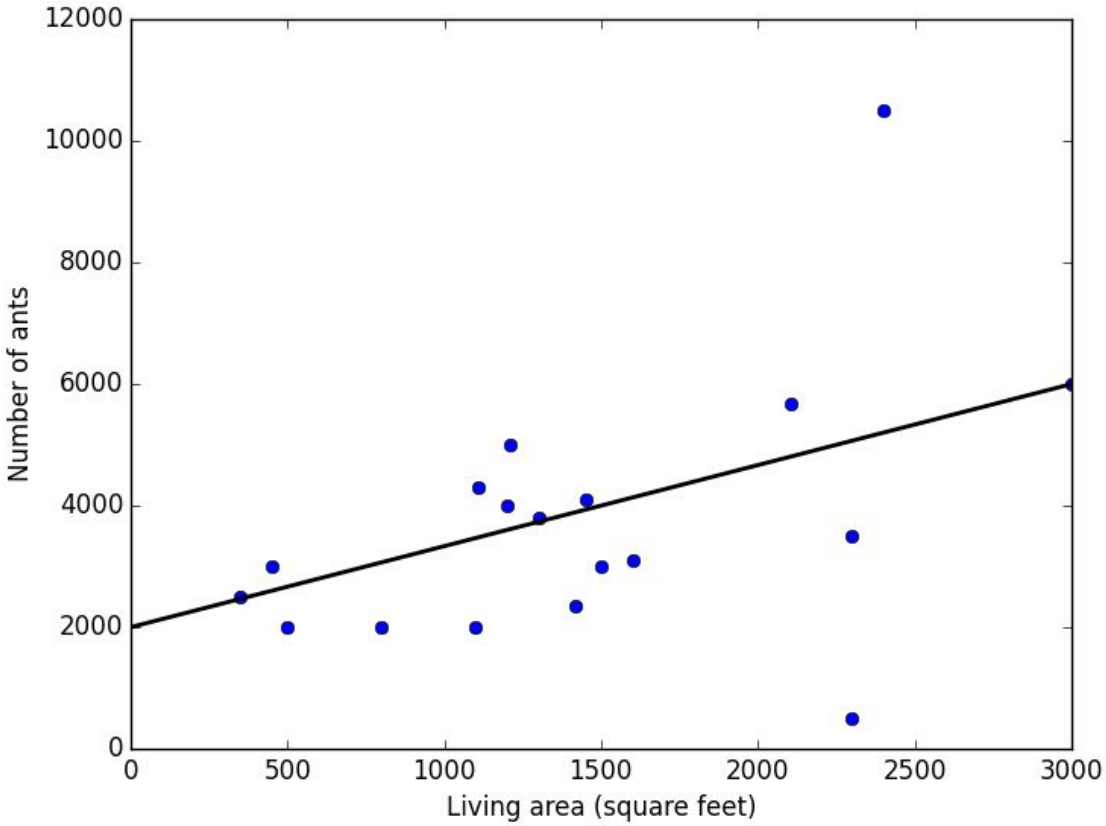
# Gradient Descent

- Here is an example of batch gradient descent:



# Gradient Descent

- Applying this algorithm to our Isla Vista data:



# Batch vs. Stochastic

- Batch has to scan through the whole dataset before taking a step - costly if $m$ is large.
- Stochastic takes a step after every training example, and thus approaches the minimum much faster.
- However, batch always converges, but stochastic may oscillate around the minimum (in practice these are still good approximations of the true minimum)
- Hence stochastic gradient descent is preferred when the training set is large.

# Linear Algebra
# Recall: Projection

- Assuming $A$ is full rank and $n<m$, the projection of $y \in \mathbb{R}^m$ onto the range (column space) of $A$ is

$$\text{Proj}(y; A) = \text{argmin}_{v \in \mathcal{R}(A)} \|v - y\|_2 = A(A^T A)^{-1} A^T y$$

- Call $P = A(A^T A)^{-1} A^T$.

# Projection

- If $b \in R(A)$, then $Pb = b$.
  - $b = Ax$ thus

$$Pb = A(A^T A)^{-1} A^T Ax = Ax = b$$

- If $b \in \mathcal{N}(A^T)$, then $Pb = 0$.
  - $A^T b = 0$ thus

$$Pb = A(A^T A)^{-1} A^T b = 0$$

  - Take for example the column space of $A$ to be a plane and $b$ a perpendicular vector.

# Projection

- See drawing on board.
- So we have that

$$Pb + (I - P)e = p + e = b$$

- Note that $I - P$ projects vectors onto the perpendicular space (check for yourself).

- Also check for yourself that if $P$ is a projection matrix, then

$$(I - P)^2 = I - P$$

# Linear Algebra

- We can actually interpret linear regression as a projection.

- For example, suppose we are given the following points in the plane:

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

- See the drawing on the board.

# Linear Algebra

- We want to find the best fit line, ie, find $C$ and $D$ for the line $y=C+Dt$.

- Equivalently, we want to solve the following systems of equations

$$C + D = 1$$
$$C + 2D = 2$$
$$C + 3D = 2$$

# Linear Algebra

- We can rewrite this using matrix notation:

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$Ax = b$$

- But notice this system has no solution - $b$ is not in the column space of $A$.

- We hope to find the “best” solution!

# Linear Algebra

- We will have some error on the best fit line.
- We will measure this error as before, namely

$$||Ax - b||^2 = ||e||^2$$

- We want to find *x* to minimize this error.
- Notice that the error is *0* iff there exists some *x* such that *Ax=b*, ie, $b \in \mathcal{R}(A)$
- In our example, $||e||^2 = e_1^2 + e_2^2 + e_3^2$, see blackboard.

# Linear Algebra

- In examples with outliers, this choice of error may not be the best. This is something to keep in mind.
- There are two pictures to keep in mind here. See the blackboard.
- We wish to find some $\hat{x} = \begin{bmatrix} \hat{C} \\ \hat{D} \end{bmatrix}$ which minimizes the squared error.
- In order to do this, we solve the normal equations:

$$A^T A\hat{x} = A^T b$$

# Linear Algebra

- In our example,

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} \hat{C} \\ \hat{D} \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix} \begin{bmatrix} \hat{C} \\ \hat{D} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$

- Observe that the matrix is symmetric, invertible, and positive semidefinite (is this always true?).

- Simplifying this yields the normal equations

$$3\hat{C} + 6\hat{D} = 5$$
$$6\hat{C} + 14\hat{D} = 11$$

# Linear Algebra

- If we had used calculus instead by minimizing

$$||e||^2 = (C + D - 1) + (C + 2D - 2)^2 + (C + 3D - 2)^2$$

by taking partial derivatives and setting equal to *0*, it would yield the identical normal equations.

  - This set of equations is always linear because the error function is quadratic!

- Solving this we get

$$\hat{D} = \frac{1}{2}, \hat{C} = \frac{2}{3}$$

# Linear Algebra

- So the best line with respect to squared error is

$$y = \frac{2}{3} + \frac{1}{2}t$$

- This yields the following points, as seen on the blackboard:

$$p_1 = \frac{7}{6}, p_2 = \frac{5}{3}, p_3 = \frac{13}{6}$$

$$e_1 = -\frac{1}{6}, e_2 = \frac{2}{6}, e_3 = -\frac{1}{6}$$

# Linear Algebra

- So in the other picture,

$$p = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} 7/6 \\ 5/3 \\ 13/6 \end{bmatrix} \quad e = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} -1/6 \\ 2/6 \\ -1/6 \end{bmatrix}$$

$$b = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix} = p + e$$

# Linear Algebra

$$p = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} 7/6 \\ 5/3 \\ 13/6 \end{bmatrix} \quad e = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} -1/6 \\ 2/6 \\ -1/6 \end{bmatrix}$$

- Notice that $p$ and $e$ are perpendicular.
- In fact, $e$ is perpendicular to any vector in the column space of $A$.
  - Test each column of $A$.
- $C$ and $D$ is the combination of the 2 columns that give $p$.

# Linear Algebra

- So given a set of points, here is the algorithm to find the best fit line:

1. Construct the matrix $A$ as we did in the example.

2. Solve the normal equations
$$A^T A\hat{x} = A^T b$$
for $\hat{x}$.

3. To find the predicted values, compute
$$p = A\hat{x}$$

# Probabilistic Interpretation

- Assume that the target variables and inputs are related via the equation

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$$

where $\epsilon^{(i)}$ is an error term representing either unmodeled effects or random noise.

- Also assume that $\epsilon^{(i)}$ are IID (independently and identically distributed) from a Gaussian distribution with mean 0 and variance $\sigma^2$.

# Probabilistic Interpretation

- This means that

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right)$$

which means that

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

where $p(y^{(i)}|x^{(i)};\theta)$ is the distribution of $y^{(i)}$ given $x^{(i)}$ and *parameterized* by $\theta$.

# Design Matrix

- Given a training set, define the design matrix $X$ to be the the matrix whose rows are the training examples:

$$X = \begin{bmatrix} - & (x^{(1)})^T & - \\ - & (x^{(2)})^T & - \\ - & (x^{(3)})^T & - \end{bmatrix}$$

- Also let

$$\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

# Likelihood Function

- Given the design matrix $X$ and $\theta$, what is the distribution of the $y^{(i)}$'s?

- The probability of the data is given by $p(\vec{y}|X;\theta)$. This is typically viewed as a function of $\vec{y}$ for a fixed $\theta$.

- When view as a function of $\theta$, it is called the likelihood function:

$$L(\theta) = L(\theta; X, \vec{y}) = p(\vec{y}|X;\theta)$$

# Likelihood Function

- Due to the independence of the $\epsilon^{(i)}$'s (and thus the $y^{(i)}$'s given the $x^{(i)}$'s), then

$$
\begin{aligned}
L(\theta) &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) \\
&= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left( -\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2} \right)
\end{aligned}
$$

# Maximum Likelihood

- So how do we choose $\theta$?
- We want to choose $\theta$ to maximize the probability of our data, ie, to maximize $L(\theta)$.
- But $L(\theta)$ is ugly to maximize - the trick is that any monotone increasing function of $L(\theta)$ will yield the same parameter.
- We will maximize the log likelihood:

$$\ell(\theta) = \log L(\theta)$$

# Maximum Likelihood

- Simplifying $\ell(\theta) = \log L(\theta)$ yields

$$
\begin{aligned}
\ell(\theta) &= \log L(\theta) \\
&= \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= \sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2.
\end{aligned}
$$

# Maximum Likelihood

- So maximizing $\ell(\theta)$ is the same as minimizing

$$\frac{1}{2}\sum_{i=1}^{m}(y^{(i)} - \theta^T x^{(i)})^2$$

- Hence, given our probabilistic assumptions on the data, least-squares-regression corresponds to finding the maximum likelihood estimate of $\theta$.

  Neato!

# Matrix Calculus Interpretation

- Recall the design matrix $X$ whose rows are the training example inputs, and column vector $\vec{y}$ whose entries are the training example outputs.

- Then since $h_\theta(x^{(i)}) = (x^{(i)})^T\theta$,

$$
\begin{aligned}
X\theta - \vec{y} &= \begin{bmatrix} (x^{(1)})^T\theta \\ \vdots \\ (x^{(m)})^T\theta \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \\
&= \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}.
\end{aligned}
$$

# Matrix Calculus Interpretation

- So we're trying to minimize this function *J*. Why don't we just take the derivative and set to 0?

- We can actually do that! We will derive this method using matrix calculus.

- Turns out that to optimize some function F, setting derivatives to 0 and solving is only useful when $\nabla F(x) = 0$ happens to be a linear system (or at least a system in which *x* can be isolated).

# Matrix Calculus Interpretation

- Then since for any vector $z$,

$$z^T z = \sum_i z_i^2$$

we have

$$\begin{aligned} \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y}) &= \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \\ &= J(\theta) \end{aligned}$$

# Matrix Calculus Interpretation

- Then

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

# Matrix Calculus Interpretation

where:

- the third equality uses the fact that the trace of a real number is the real number,
- the fourth equality uses the fact that the trace of a matrix is the trace of its transpose,
- the fifth equality uses

$$\nabla_{A^T} \mathrm{tr} ABA^T C = B^T A^T C^T + BA^T C$$

with $A^T = \theta, B - B^T = X^T X, C = I$, and that $\nabla_A \mathrm{tr} AB = B^T$.

# Matrix Calculus Interpretation

So to minimize *J*, we set its derivatives to zero, and we get the normal equations:

$$X^T X\theta = X^T \vec{y}$$

Solving for $\theta$, if *X* has full column rank, we have

$$\theta = (X^T X)^{-1} X^T \vec{y}$$

Hey, the same as the linear algebra interpretation!

# Matrix Calculus vs. Gradient Descent

- So solving for the maximal $\theta$ reduces to computing the matrix product above (which involves computing an inverse of a very large matrix).

- However, in practice, this inverse is never computed. Instead, the system is posed in the form $(X^T X)\theta = X^T \vec{y}$ and solved using a linear solver.

- This method is cheaper, and allows exploitation of the coefficient matrix (using bandedness, symmetry, sparsity) and other methods.

# Matrix Calculus vs. Gradient Descent

- Bottom-line:

- when the first order derivative system is linear, solving it directly is much more computationally efficient than gradient descent (which can have slow convergence).

- Otherwise, other strategies (including gradient descent) may be better.

- Note: people like to use gradient descent for convex optimization because it is easy to implement and relatively cheap computationally.

# What Just Happened?

- Linear Regression Interpretations:

1. Least Squares
2. Linear Algebra
3. Probabilistic
4. Matrix Calculus