CMSC 726 Lecture 5:Linear Models for Regression

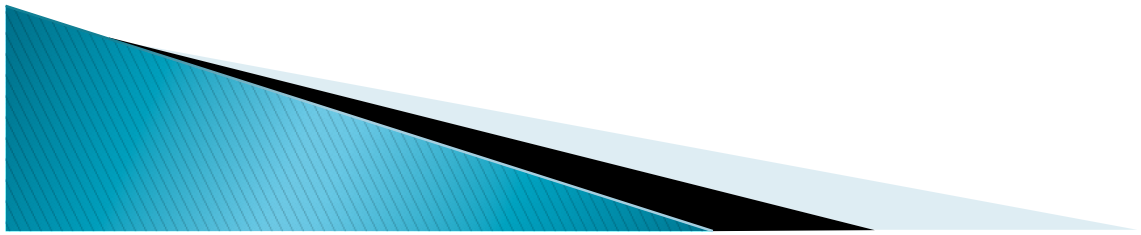
Lise Getoor

September 14, 2010

ACKNOWLEDGEMENTS: The material in this course is a synthesis of materials from many sources, including: **Hal Daume III**, Mark Drezde, Carlos Guestrin, **Andrew Ng**, Ben Taskar, **Eric Xing**, and others. I am very grateful for their generous sharing of insights and materials.

Today's Topics

- Linear Regression
- LMS & the Normal Equations
- LS vs. MLE



ML for apartment hunting

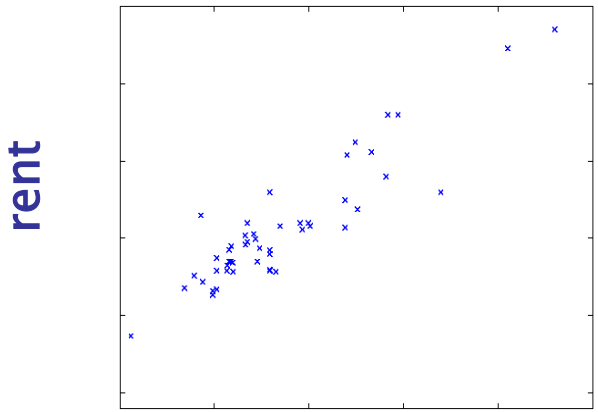
- Now you've moved to College Park!!

  Given information about apartments and their prices, can you learn to predict the price of an apartment?

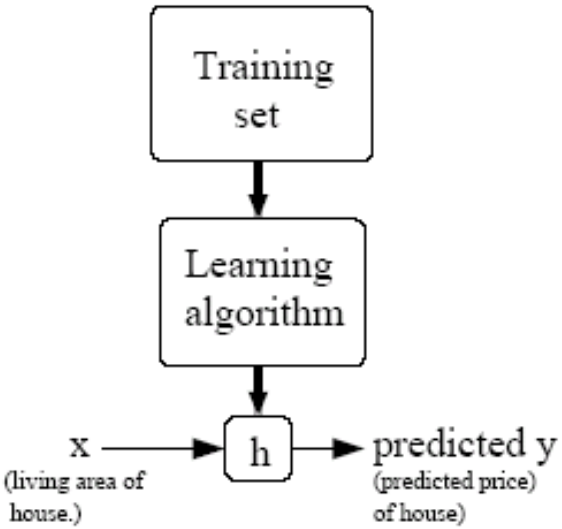
  square-ft., # of bedroom, distance to campus …

Living area (ft^2)	# bedroom	Rent ($)
230	1	600
506	2	1000
433	2	1100
109	1	500
…		
150	1	?
270	1.5	?

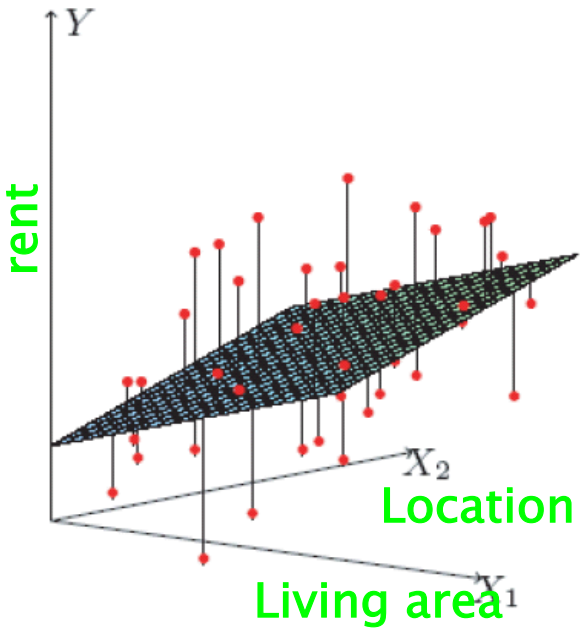
The learning problem in 2D



Our goal:



The learning problem



- Features:
  - Living area, distance to campus, # bedroom …
  - Denote as $\mathbf{x}=[x^1, x^2, \ldots x^m]$
- Target:
  - Rent
  - Denoted as y
- Training set:

$$\mathbf{X} = \begin{bmatrix} -- & \mathbf{x}_1 & -- \\ -- & \mathbf{x}_2 & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_n & -- \end{bmatrix} = \begin{bmatrix} x_1^1 & x_1^2 & \ldots & x_1^m \\ x_2^1 & x_2^2 & \ldots & x_2^m \\ \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^2 & \ldots & x_n^m \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Linear Regression

- Assume that Y (target) is a linear function of X (features):
  - e.g.: $\hat{y} = h(x) = w_0 + w_1 x^1 + w_2 x^2$
  - let's assume a vacuous "feature" $x^0 = 1$ (this is the intercept term), and define the feature vector to be:

$$x = [x^0, x^1, \ldots, x^m]$$

  - then we have the following general representation of the linear function:

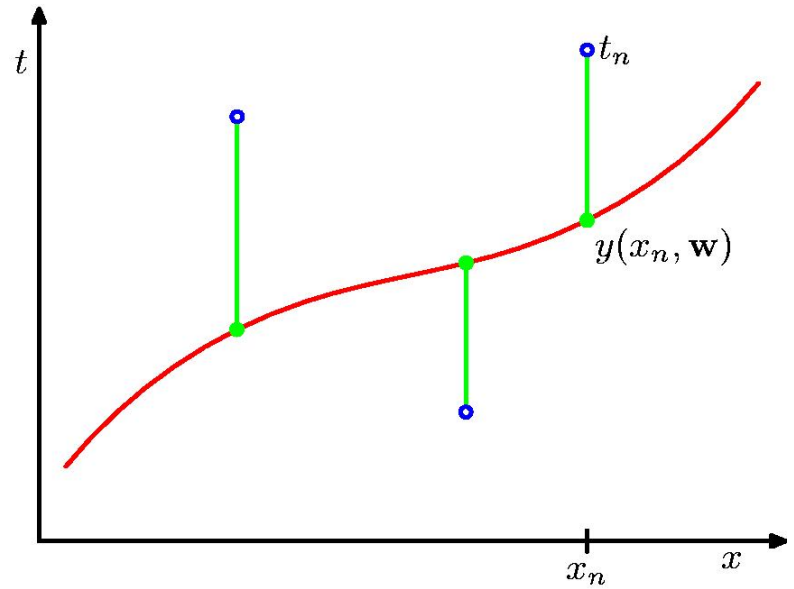
$$h(x) = \sum_{i=1}^{m} w_i x^i = w^T x$$

- Our goal is to pick the optimal w. How??

MSE Cost Function

We seek w to that minimize the following **mean squared error** (MSE) **cost function**:

$$J(w) = \frac{1}{2}\sum_{i=1}^{N}(h(x_i) - y_i)^2$$



Notation from book:
t = y, h(x) = y(x,w)

Least-Mean-Square (LMS) method

- Search algorithm: start with initial guess for w, repeatedly change w to make error smaller, until we hopefully converge to a value of w that minimizes J(w).
- The Cost Function:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(\mathbf{w}^T\mathbf{x}_i - y_i)^2$$

- Consider a **gradient descent** algorithm:

$$w_j^{t+1} = w_j^t - \alpha \frac{\partial}{\partial w_j} J(w)\Big|_t$$

Computing Partial Derivative

- Let's start by working it out for the case where we just have one example, (x,y):

$$\frac{\partial}{\partial w_j} J(w) = \frac{\partial}{\partial w_j} \frac{1}{2}(h(x) - y)^2$$

$$= 2 \cdot \frac{1}{2}(h(x) - y) \frac{\partial}{\partial w_j}(h(x) - y)$$

$$= (h(x) - y) \frac{\partial}{\partial w_j}\left(\sum_{i=1}^{m} w_i x^i - y\right)$$

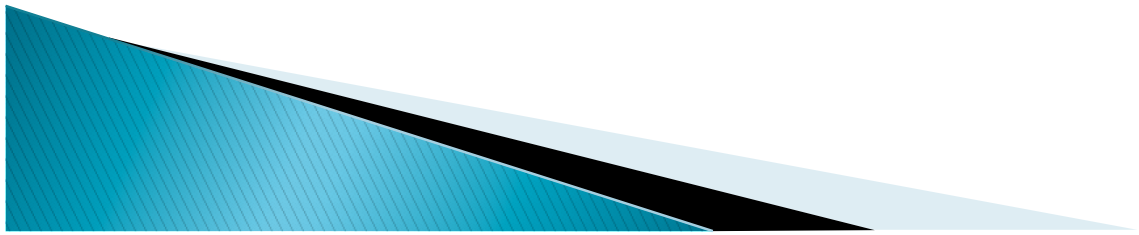
$$= (h(x) - y) x^j$$

The LMS method

- Now we have the following update rule:

$$w_j^{t+1} = w_j^t - \alpha(h(x) - y)x^j$$

  - This is known as the LMS update rule, or the Widrow-Hoff learning rule
  - Desirable properties: magnitude of update depends on error.

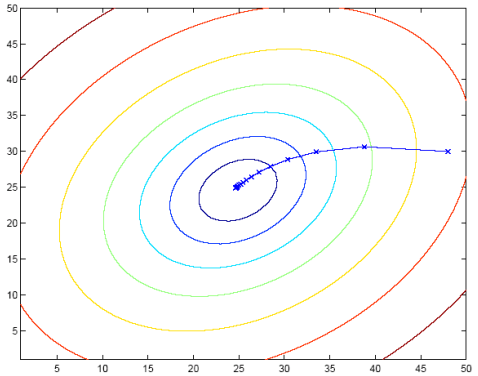


Batch Gradient Descent

- What if we have more than one training example?
- Repeat until convergence {

$$w_j^{t+1} = w_j^t - \alpha \sum_{i=1}^{N} (h(x_i) - y_i) x_i^j, \quad for every\ j$$

}



- While gradient descent can be susceptible to local minima in general, for the MSE for linear regression, J (w), is a convex quadratic function, and (assuming α is not too large) the algorithm always converges

Stochastic Gradient Descent

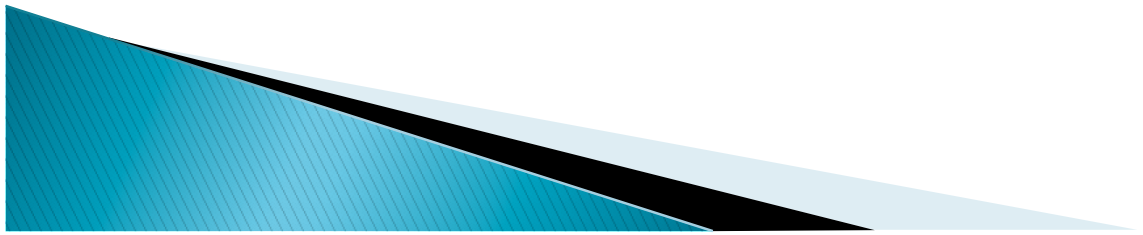
- Alternative… (also called incremental gradient descent)
- Loop (
  - For I = 1 to n {

$$w_j^{t+1} = w_j^t - \alpha(h(x_i) - y_i)x_i^j, \quad for every\ j$$

  }
 }
- Here, we update the parameters according to the gradient of the error with respect to a single example only; in batch, we much process the whole data before taking a step.
- Often, stochastic gradient descent gets close to w much faster than gradient. However, it may oscillate.
- It is often preferred for large data sets.

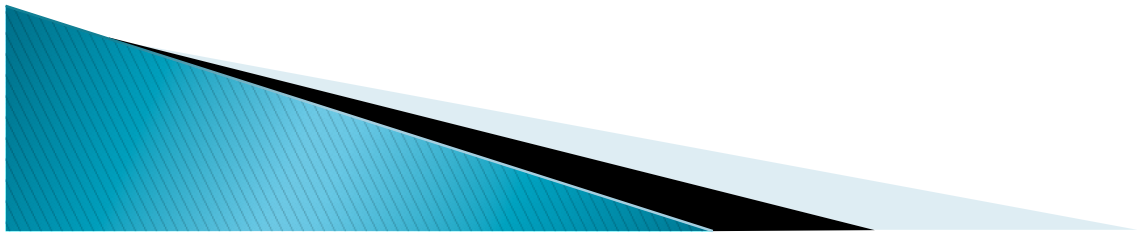
Alternative

- Gradient descent is an iterative algorithm
- We can also minimize J explicitly, by taking its derivatives wrt to W and setting them to 0.
- To do this, we need some matrix calculus and algebra….



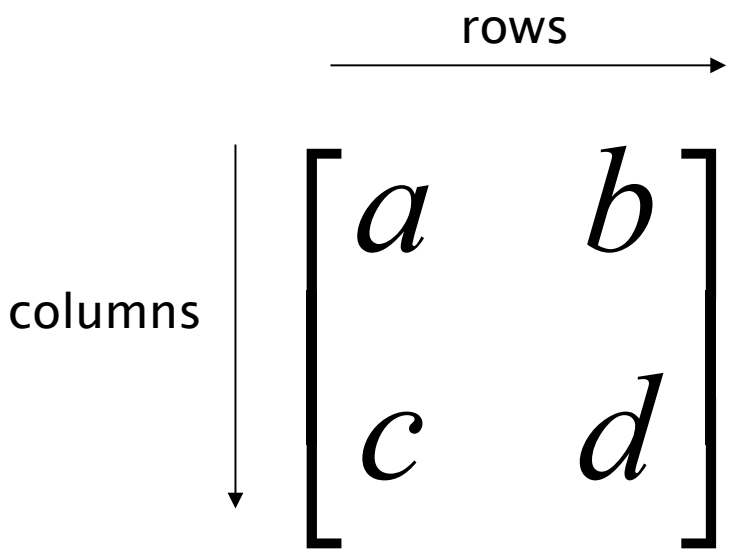
Linear Algebra Refresher….

- A good source is here:
  - http://www.stanford.edu/class/cs229/section/cs229-linalg.pdf
- There is a pointer from elms….



What is a Matrix?

- A matrix is a set of elements, organized into rows and columns



Basic Operations

- Addition, Subtraction, Multiplication

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a+e & b+f \\ c+g & d+h \end{bmatrix}$$

Just add elements

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a-e & b-f \\ c-g & d-h \end{bmatrix}$$

Just subtract elements

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix}$$

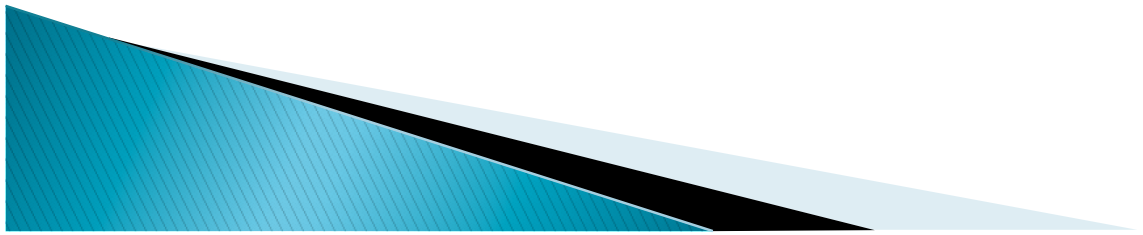
Multiply each row by each column

Multiplication

- Is AB = BA? Maybe, but maybe not!

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+bg & \dots \\ \dots & \dots \end{bmatrix} \quad \begin{bmatrix} e & f \\ g & h \end{bmatrix}\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} ea+f\,c & \dots \\ \dots & \dots \end{bmatrix}$$

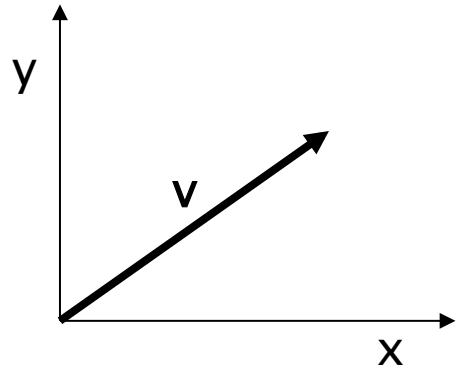
- Heads up: multiplication is NOT commutative!



Vector Operations

- Vector: 1 x N matrix
- Interpretation: a line in N dimensional space
- Dot Product, Cross Product, and Magnitude defined on vectors only

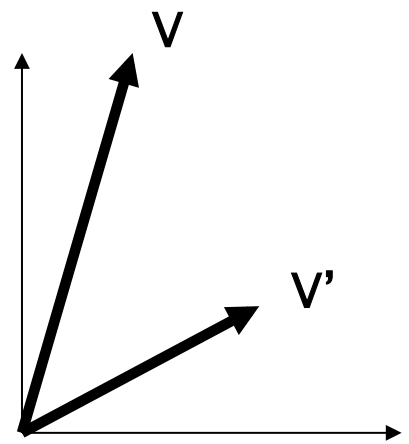
$$\vec{v} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$



Vector Interpretation

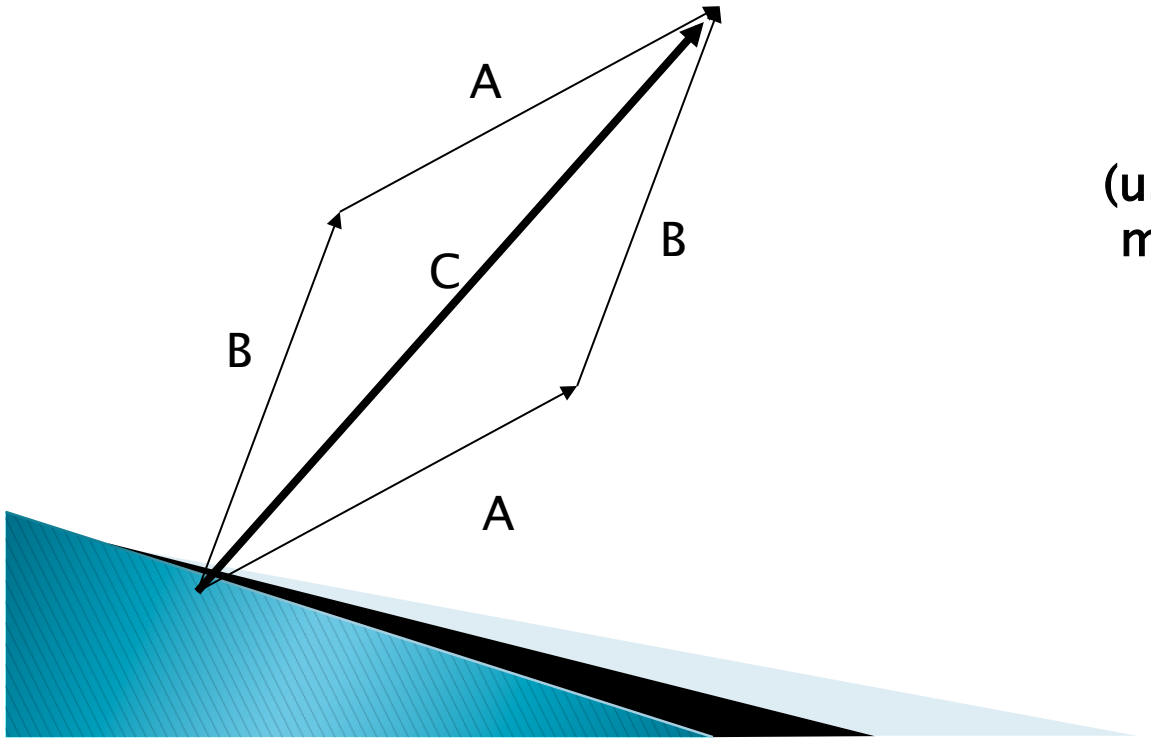
- Think of a vector as a line in 2D or 3D
- Think of a matrix as a transformation on a line or set of lines

$$\begin{bmatrix} x \\ y \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$



Vectors: Dot Product

- Interpretation: the dot product measures to what degree two vectors are aligned

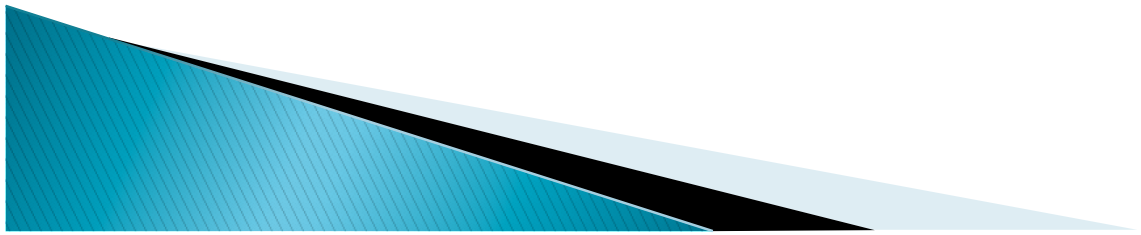


A+B = C
(use the head-to-tail method to combine vectors)

Vectors: Dot Product

$$a \cdot b = ab^T = \begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} d \\ e \\ f \end{bmatrix} = ad + be + cf$$

Think of the dot product as a matrix multiplication



Vector Norms

Norms, magnitude, lengths all different ways of associating a scalar with vector

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

The Euclidean norm is the length or magnitude of a vector; also called the L_2 norm

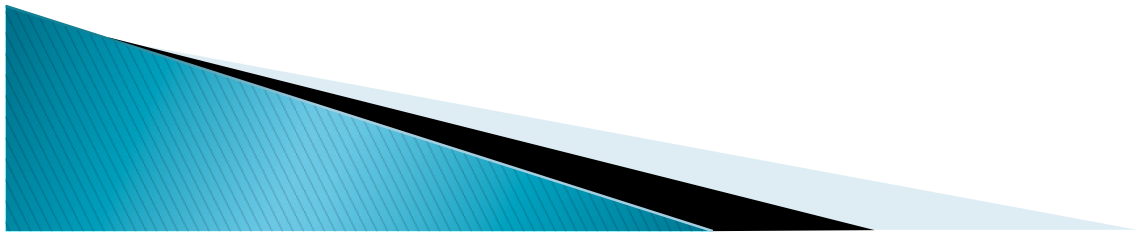
$$\|x\|_2^2 = x^T x$$

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

The Manhattan distance, the L_1 norm

$$\|x\|_\infty = \max_i |x_i|$$

The L infinity norm



Inverse of a Matrix

- Identity matrix:
  $AI = A$
- Some matrices have an inverse, such that:
  $AA^{-1} = I$
- Inversion is tricky:
  $(ABC)^{-1} = C^{-1}B^{-1}A^{-1}$
  Derived from non-commutativity property

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

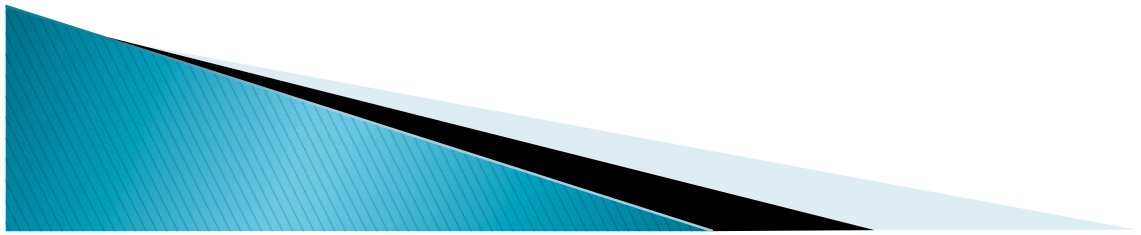
Determinant of a Matrix

- Used for inversion
- If det(A) = 0, then A has no inverse
- Notation: |A|

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

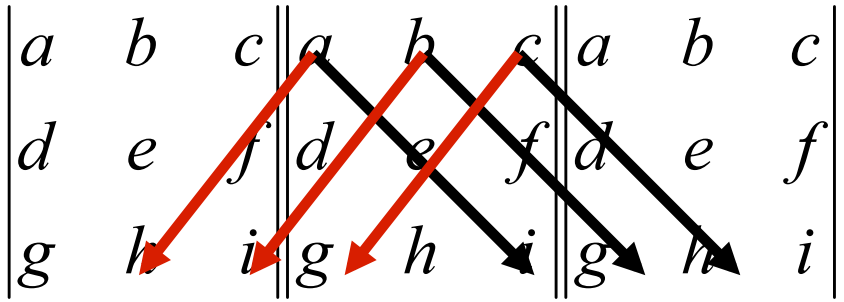
$$\det(A) = ad - bc$$

$$A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$



Determinant of a Matrix

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = aei + bf\,g + cdh - af\,h - bdi - ceg$$



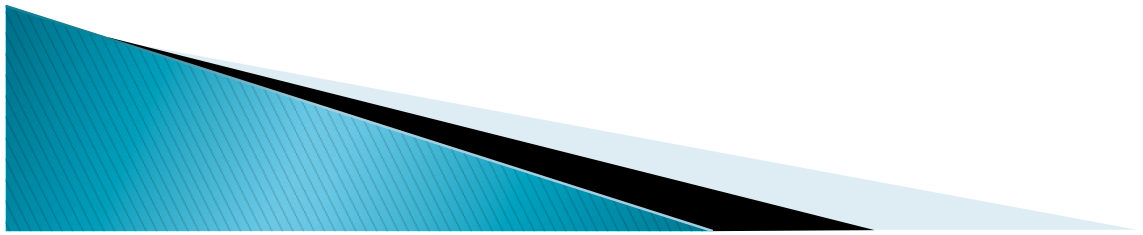
Sum from left to right
Subtract from right to left
Note: N! terms

Other Useful Properties

$$(A^T)_{ij} = A_{ji}$$ Transpose

$$(A^T)^T = A$$

$$(AB)^T = B^T A^T$$



Some matrix math…

- For $f : \mathbf{R}^{m \times n} \mapsto \mathbf{R}$, define:

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial}{\partial A_{11}} f & \cdots & \frac{\partial}{\partial A_{1n}} f \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial A_{1m}} f & \cdots & \frac{\partial}{\partial A_{mn}} f \end{bmatrix}$$

- Trace: $\mathrm{tr}A = \sum_{i=1}^{n} A_{ii}$, $\mathrm{tr}a = a$, $\mathrm{tr}ABC = \mathrm{tr}CAB = \mathrm{tr}BCA$

- Some facts about matrix derivatives

$\nabla_A \mathrm{tr}AB = B^T$, $\nabla_A \mathrm{tr}ABA^T C = CAB + C^T AB^T$, $\nabla_A |A| = |A|\left(A^{-1}\right)^T$

The normal equations

$$\mathbf{X} = \begin{bmatrix} -- & \mathbf{x}_1 & -- \\ -- & \mathbf{x}_2 & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_n & -- \end{bmatrix}$$

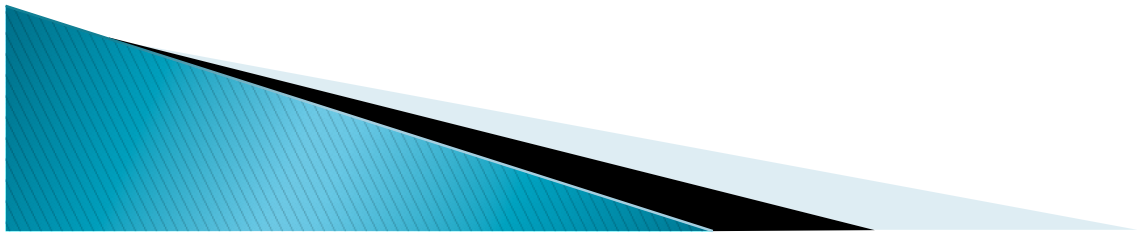
- Write the cost function in matrix form:

$$\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$J(w) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i{}^T w - y_i)^2$$

$$= \frac{1}{2}(Xw - \vec{y})^T(Xw - \vec{y})$$

$$= \frac{1}{2}(w^T X^T X w - w^T X^T \vec{y} - \vec{y}^T X w + \vec{y}^T \vec{y})$$



The normal equations

- To minimize $J(w)$, take derivative and set to zero:

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

$\Rightarrow$ **The normal equations**

$$X^T X \mathbf{w} = X^T \mathbf{y}$$

$$\Downarrow$$

$$\mathbf{w} = \left(X^T X\right)^{-1} X^T \mathbf{y}$$

Comments on the normal equation

- In most situations of practical interest, the number of data points N is larger than the dimensionality m of the input space and the matrix $\mathbf{X}$ is of full column rank. If this condition holds, then it is easy to verify that $X^T X$ is necessarily invertible.

- The assumption that $X^T X$ is invertible implies that it is positive definite, thus the critical point we have found is a minimum.

- What if $\mathbf{X}$ has less than full column rank?

Direct and Iterative methods

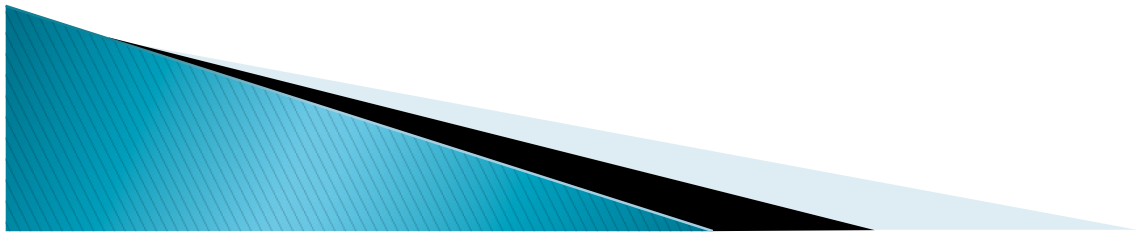
- Direct methods: we can achieve the solution in a single step by solving the normal equation
  - Using Gaussian elimination or QR decomposition, we converge in a finite number of steps
  - It can be infeasible when data are streaming in real time, or very large amount
- Iterative methods: stochastic or batch gradient
  - Attractive in large practical problems
  - Caution is needed for deciding the learning rate α

LMS Summary:

- LMS update rule $w_j^{t+1} = w_j^t - \alpha(w^T \mathbf{x}_i - y_i)x_i^j$
  - Pros: on-line, low per-step cost, fast convergence
  - Cons: convergence to optimum not always guaranteed
- Batch Gradient descent $w_j^{t+1} = w_j^t - \alpha \sum_{i=1}^{N} (h(x_i) - y_i)x_i^j$
  - Pros: easy to implement, conceptually clean, guaranteed convergence
  - Cons: batch, often slow converging
- Normal equations $w = \left(X^T X\right)^{-1} X^T \vec{y}$
  - Pros: a single-shot algorithm! Easiest to implement.
  - Cons: need to compute pseudo-inverse $(X^T X)^{-1}$, expensive, numerical issues (e.g., matrix is singular ..), although there are ways to get around this …

Alternative View of LR

- Probabilistic Interpretation…

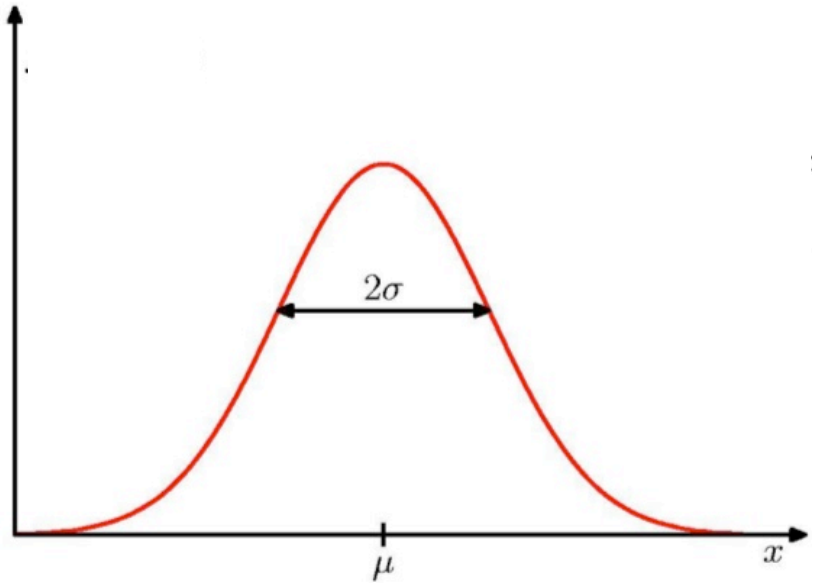


The Gaussian Distribution

$$N(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

$$\mathrm{E}[x] = \mu$$

$$\mathrm{var}[x] = \sigma^2$$



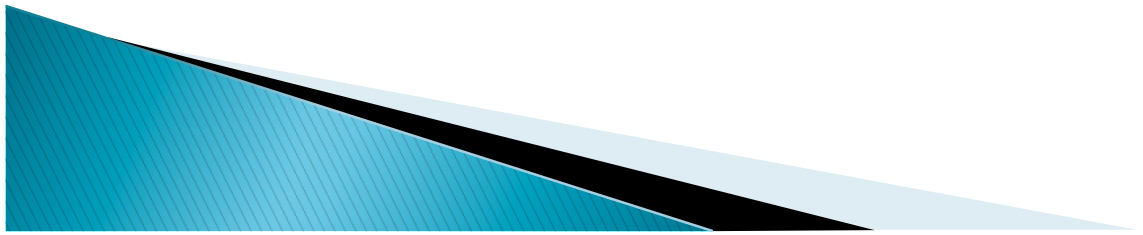
Noise

- Reality: no regression is that easy
- Assume output permuted by Gaussian noise

$$\varepsilon \sim N(\mu,\sigma^2), \qquad \mu = 0, \quad \sigma^2 = 1$$

$$y = h_w(x) + \varepsilon$$

- The data isn't really generated in this way
  - Assume that it is for sake of modeling



Probabilistic Interpretation of LMS

- Let us assume that the target variable and the inputs are related by the equation:

$$y_i = w^T \mathbf{x}_i + \varepsilon_i$$

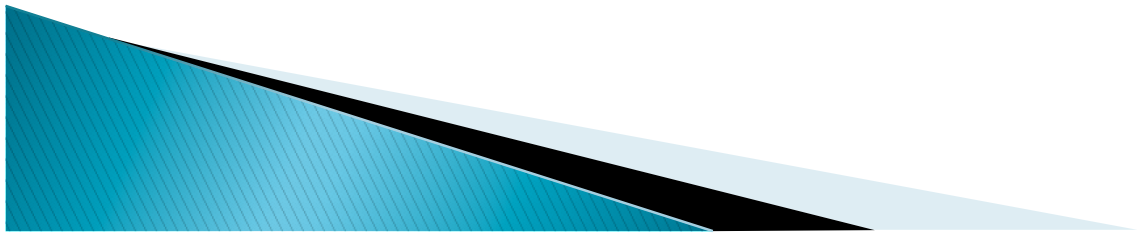
where ϵ is an error term of unmodeled effects or random noise

- Now assume that ϵ follows a Gaussian $N(0,\sigma)$, then we have:

$$p(y_i \mid x_i; w) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - w^T \mathbf{x}_i)^2}{2\sigma^2} \right)$$

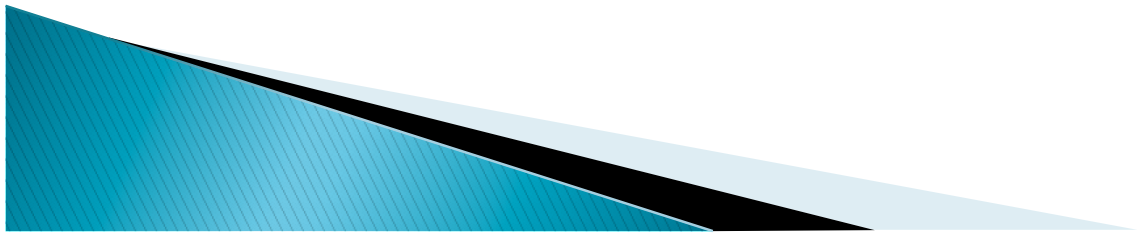
How to choose w?

- Select the parameters that maximize the likelihood of the data
  - Likelihood = probability of data occurring
- How do we measure the likelihood of the data?



Maximum Likelihood

- Maximum likelihood estimation
  - Find the parameters that maximize the likelihood function
  - Function optimization
- Strategy
  - Write a function
  - Maximize
    - Take derivative with respect to parameters
    - Set to 0
    - Solve

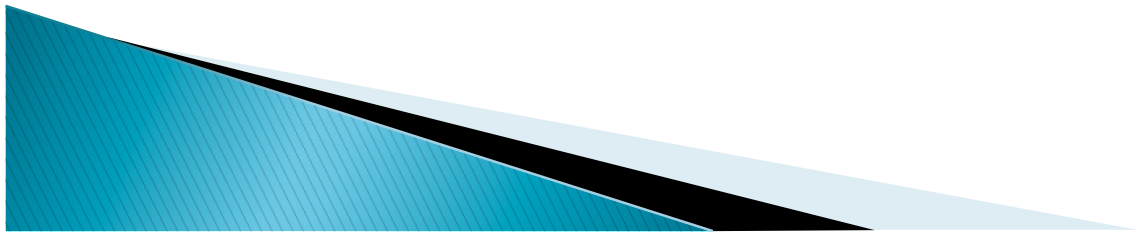


Likelihood

- Recall that the data is generated from a Gaussian distribution
  - This is a probability distribution, so we can measure the probability of the data!

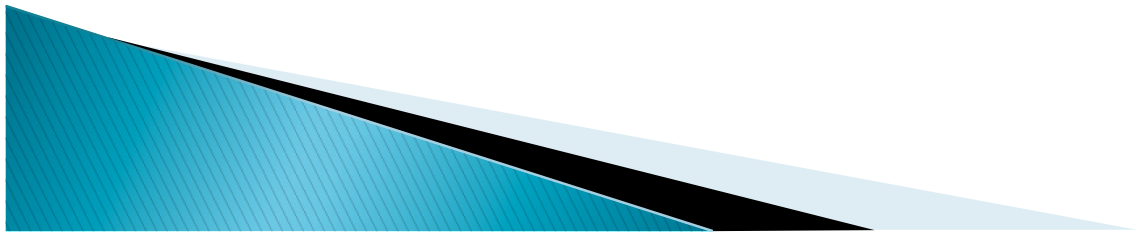
$$p(Y \mid X, w, \sigma^2) = \prod_{i=1}^{N} N(w \cdot x_i, \sigma^2)$$

$$L(w) = \prod_{i=1}^{N} p(y_i \mid x_i; w) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{\sum_{i=1}^{N}(y_i - w^T \mathbf{x}_i)^2}{2\sigma^2}\right)$$



Log Likelihood

- Why log likelihood?
  - A useful way to decompose terms of the function
  - Computationally advantageous to represent small numbers using their log



Probabilistic Interpretation of LMS, cont.

- Hence the log-likelihood is:

$$l(\theta) = N \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2}\frac{1}{2}\sum_{i=1}^{n}(y_i - w^T \mathbf{x}_i)^2$$

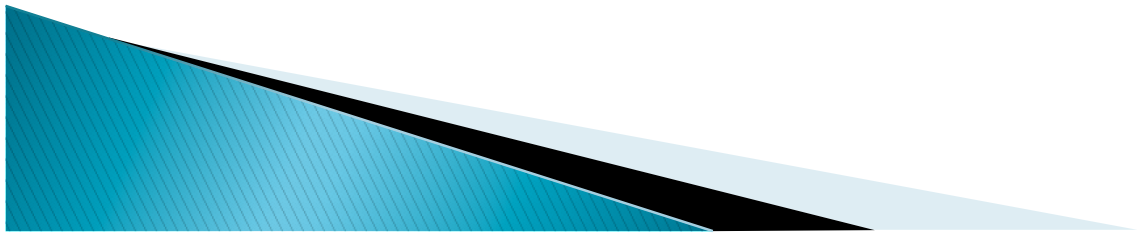
- Do you recognize the last term?

Yes it is: $$J(w) = \frac{1}{2}\sum_{i=1}^{N}(w^T \mathbf{x}_i - y_i)^2$$

- Thus under independence assumption, LMS is equivalent to MLE of w !

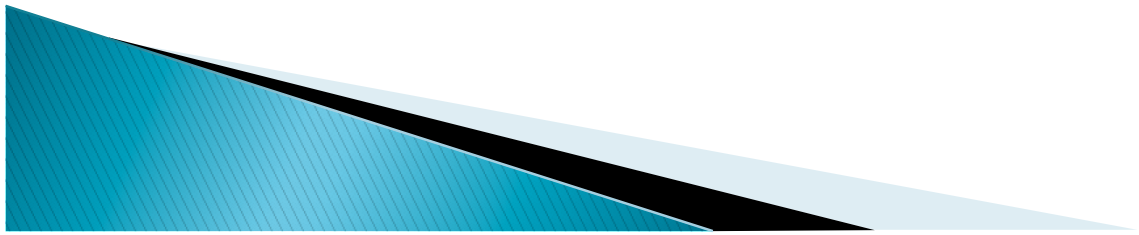
Summary: MLE = LMS

- Minimizing the squared error is equivalent to maximizing the likelihood (under the assumption of iid Gaussian noise)



Beyond basic LR

- LR with non-linear basis functions
- Locally weighted linear regression
- Regression trees



LR with non-linear basis functions

- LR does not mean we can only deal with linear relationships
- We are free to design (non-linear) features under LR

$$y = w_0 + \sum_{j=1}^{m} w_j \phi(x) = w^T \phi(x)$$

where the $\phi_j(x)$ are fixed basis functions (and we define $\phi_0(x) = 1$).

- Example: polynomial regression:

$$\phi(x) := \lfloor 1, x, x^2, x^3 \rfloor$$