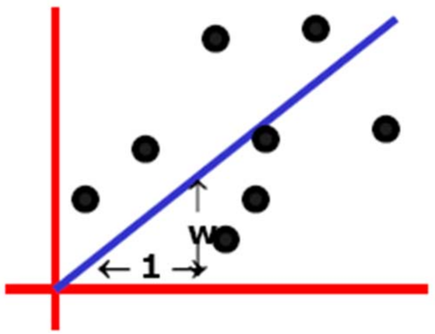
Advanced Introduction to Machine Learning

10715, Fall 2014

Linear Regression and Sparsity

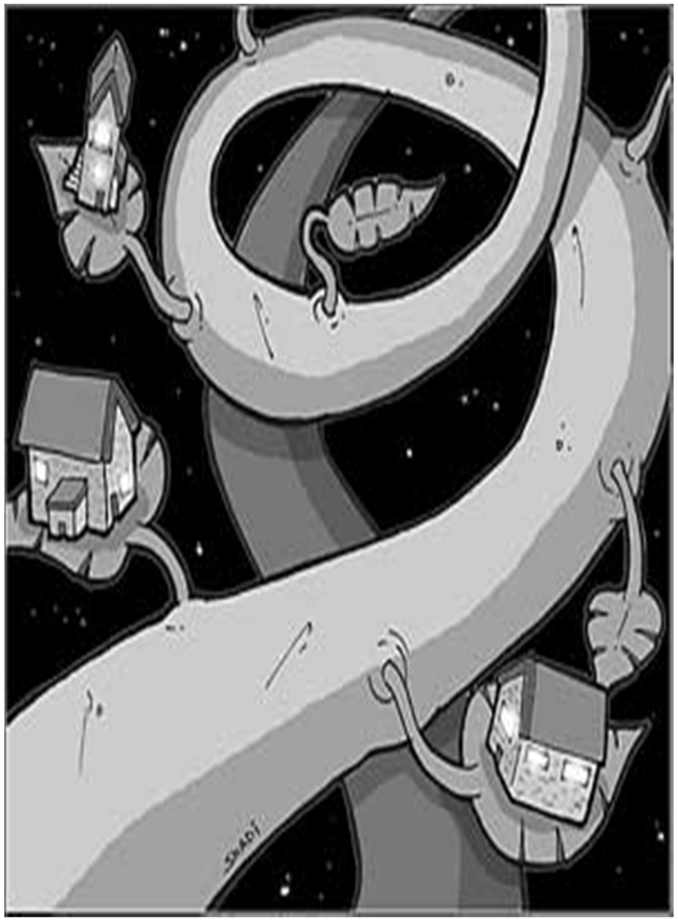


Eric Xing

Lecture 2, September 10, 2014

Reading:

Machine learning for apartment hunting



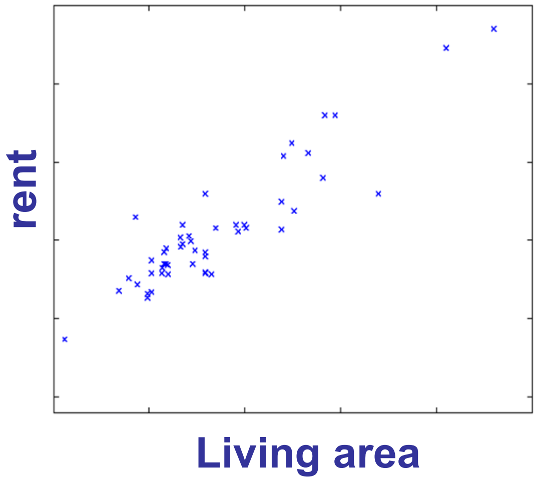
- Now you've moved to Pittsburgh!!

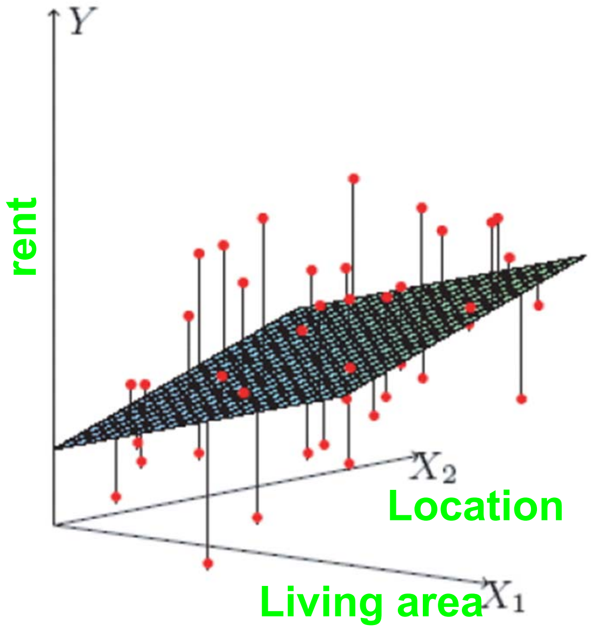
  And you want to find the **most reasonably priced** apartment satisfying your **needs:**

  - 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



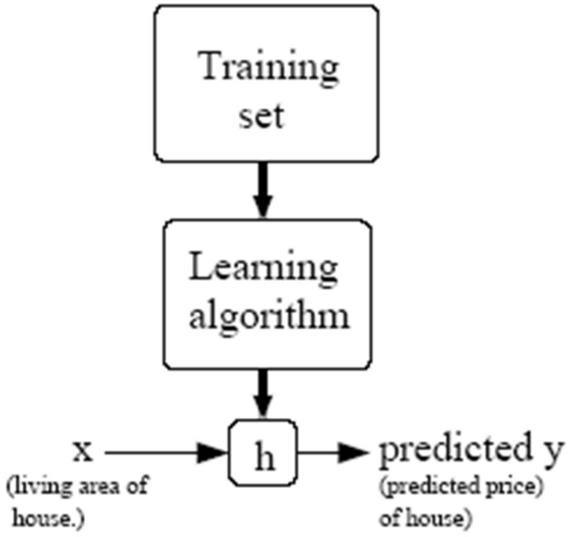


- Features:
  - Living area, distance to campus, # bedroom …
  - Denote as $\mathbf{x}=[x^1, x^2, \ldots x^k]$
- Target:
  - Rent
  - Denoted as y
- Training set:

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

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} - & \mathbf{y}_1 & - \\ - & \mathbf{y}_2 & - \\ \vdots & \vdots & \vdots \\ - & \mathbf{y}_n & - \end{bmatrix}$$

Our goal:



$y = f(x)$

Linear Regression

- Assume that Y (target) is a linear function of X (features):

  - e.g.:

$$\underline{\hat{y}} = \theta_0 + \theta_1 \boxed{x^1} + \theta_2 \boxed{x^2}$$

  - let's assume a vacuous "feature" X^0=1 (this is the intercept term, why?), and define the feature vector to be:

$$X = [1,\ x^1\ \ x^2 \cdots x^n]$$

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

$$\hat{y} = \theta^T X$$
$$y = f(x)$$

- Our goal is to pick the optimal θ . How!
  - We seek θ that minimize the following **cost function**:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (\hat{y}_i(\vec{x}_i) - y_i)^2$$

The Least-Mean-Square (LMS) method

- The Cost Function:

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

- Consider a **gradient descent** algorithm:

$$\theta_j^{t+1} = \theta_j^t - \alpha \left. \frac{\partial}{\partial \theta_j} J(\theta) \right|_t$$

$$= \theta_j^t + \alpha \, \frac{1}{2} \cdot 2 \sum_{i=1}^{n} (x_i^T\theta - y_i)\, x_i$$

The Least-Mean-Square (LMS) method

- Now we have the following descent rule:

$$\theta_j^{t+1} = \theta_j^{t} + \alpha \sum_{i=1}^{n} (y_i - \vec{\mathbf{x}}_i^T \theta^t) x_i^j$$

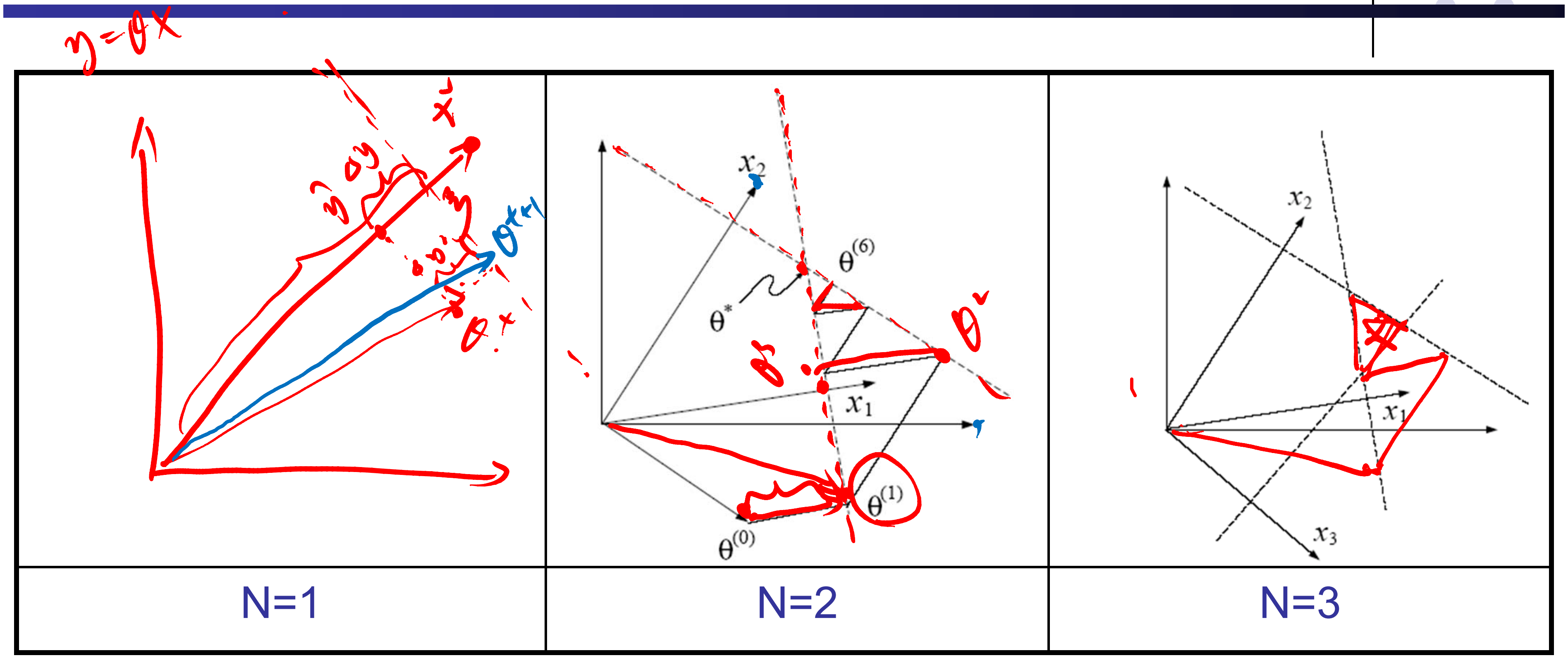
- For a single training point, we have:

$$\theta_j^{t+1} = \theta_j^{t} + \alpha (y_i - \vec{x}_i^T \theta^t) x_i^j$$

$$\vec{\theta}^{t+1} = \vec{\theta}^{t} + \alpha (y_i - \vec{x}_i^T \theta^t) \vec{x}_i$$

  - This is known as the LMS update rule, or the Widrow-Hoff learning rule
  - This is actually a "**stochastic**", "**coordinate**" descent algorithm
  - This can be used as a **on-line** algorithm

Geometry and Convergence of LMS



N=1 N=2 N=3

$$\theta^{t+1} = \theta^t + \alpha(y_i - \vec{\mathbf{x}}_i^{\,T}\theta^t)\vec{\mathbf{x}}_i$$

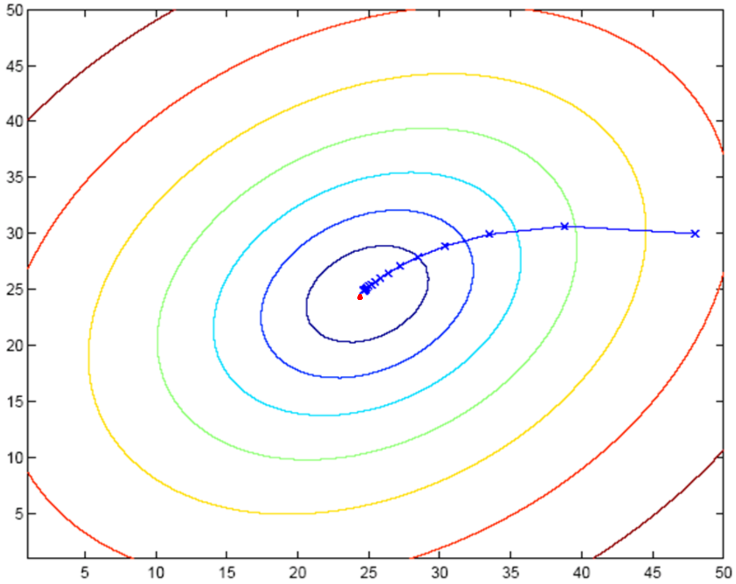
Claim: when the step size α satisfies certain condition, and when certain other technical conditions are satisfied, LMS will converge to an "optimal region".

Steepest Descent and LMS

- Steepest descent
  - Note that:

$$\nabla_\theta J = \left[\frac{\partial}{\partial\theta_1}J, \ldots, \frac{\partial}{\partial\theta_k}J\right]^T = -\sum_{i=1}^{n}(y_n - \mathbf{x}_n{}^T\theta)\mathbf{x}_n$$

$$\theta^{t+1} = \theta^t + \alpha\sum_{i=1}^{n}(y_n - \mathbf{x}_n{}^T\theta^t)\mathbf{x}_n$$



  - This is as a **batch** gradient descent algorithm

The normal equations

- Write the cost function in matrix form:

$$\begin{aligned} J(\theta) &= \underline{\frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^T\theta - y_i)^2} \\ &= \underline{\frac{1}{2}(X\theta-\vec{y})^T(X\theta-\vec{y})} \\ &= \frac{1}{2}\left(\theta^T X^T X\theta - \theta^T X^T\vec{y} - \vec{y}^T X\theta + \vec{y}^T\vec{y}\right) \end{aligned}$$

$\rightarrow \theta^*$

$$\mathbf{X} = \begin{bmatrix} -- & \mathbf{x}_1 & -- \\ -- & \mathbf{x}_2 & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_n & -- \end{bmatrix} \qquad \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

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

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

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

The normal equations

$$\Downarrow$$

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

© Eric Xing @ CMU, 2014

Comments on the normal equation

- In most situations of practical interest, the number of data points N is larger than the dimensionality k 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 at the critical point we have found is a minimum.

- What if $\mathbf{X}$ has less than full column rank? → regularization (later).

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 in real time, or of very large amount

- Iterative methods: stochastic or steepest gradient
  - Converging in a limiting sense
  - But more attractive in large practical problems
  - Caution is needed for deciding the learning rate α

Convergence rate

- **Theorem**: the steepest descent equation algorithm converge to the minimum of the cost characterized by normal equation:

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

If

$$0 < \alpha < 2/\lambda_{\max}[X^T X]$$

- A formal analysis of LMS need more math-mussels; in practice, one can use a small α, or gradually decrease α.

A Summary:

- LMS update rule

$$\theta_j^{t+1} = \theta_j^{t} + \alpha(y_n - \mathbf{x}_n^T \theta^t) x_{n,i}$$

  - Pros: on-line, low per-step cost, fast convergence and perhaps less prone to local optimum
  - Cons: convergence to optimum not always guaranteed

- Steepest descent

$$\theta^{t+1} = \theta^{t} + \alpha \sum_{i=1}^{n} (y_n - \mathbf{x}_n^T \theta^t) \mathbf{x}_n$$

  - Pros: easy to implement, conceptually clean, guaranteed convergence
  - Cons: batch, often slow converging

- Normal equations

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

  - 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 …

Geometric Interpretation of LMS

- The predictions on the training data are:

$$\hat{\vec{y}} = X\theta^* = X\left(X^T X\right)^{-1} X^T \vec{y}$$

$$\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} -- & \mathbf{x}_1 & -- \\ -- & \mathbf{x}_2 & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_n & -- \end{bmatrix}$$

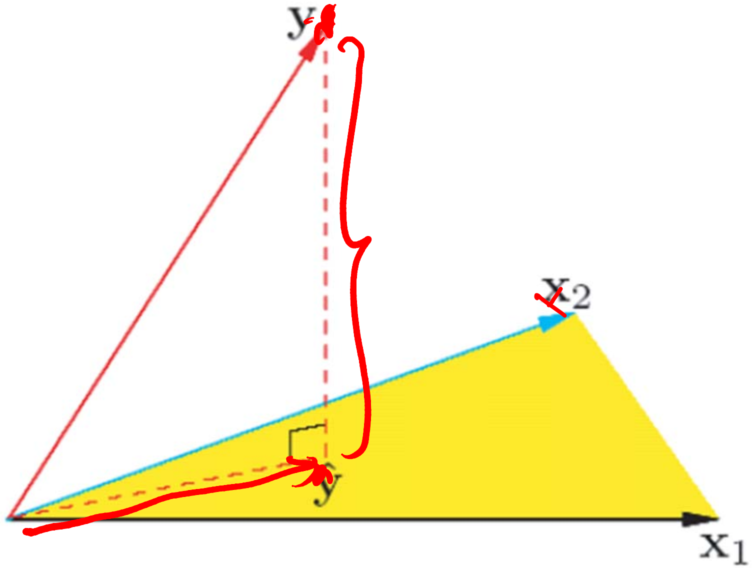
- Note that

$$\hat{\vec{y}} - \vec{y} = \left(X\left(X^T X\right)^{-1} X^T - I\right)\vec{y}$$

and

$$\begin{aligned} X^T\boxed{\left(\hat{\vec{y}} - \vec{y}\right)} &= X^T\left(X\left(X^T X\right)^{-1} X^T - I\right)\vec{y} \\ &= \left(X^T X\left(X^T X\right)^{-1} X^T - X^T\right)\vec{y} \\ &= 0 \quad \textbf{!!} \end{aligned}$$

$\hat{\vec{y}}$ is the orthogonal projection of $\vec{y}$ into the space spanned by the column of X



Probabilistic Interpretation of LMS

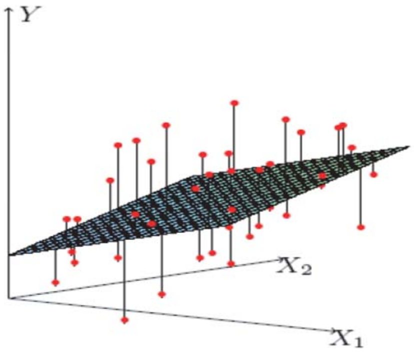
$y = \theta^T x$

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

$y \approx \theta^T x$

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

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:

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

$\theta^* = \arg\max \prod_i^n p(y_i | x_i)$

- By independence assumption:

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

Probabilistic Interpretation of LMS, cont.

- Hence the log-likelihood is:

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

- Do you recognize the last term?

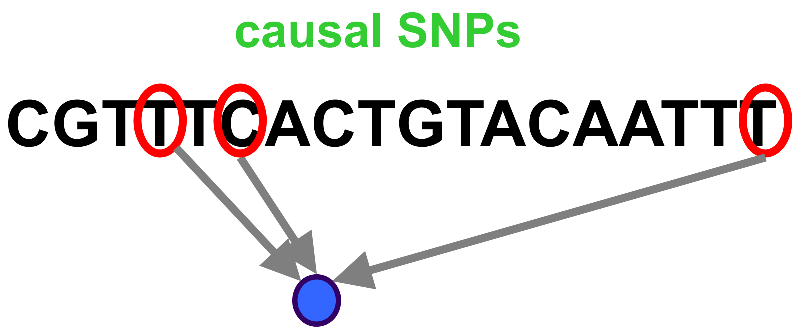
Yes it is: $$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^T\theta - y_i)^2$$

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

Case study:
predicting gene expression

$y = f(x)$

The genetic picture

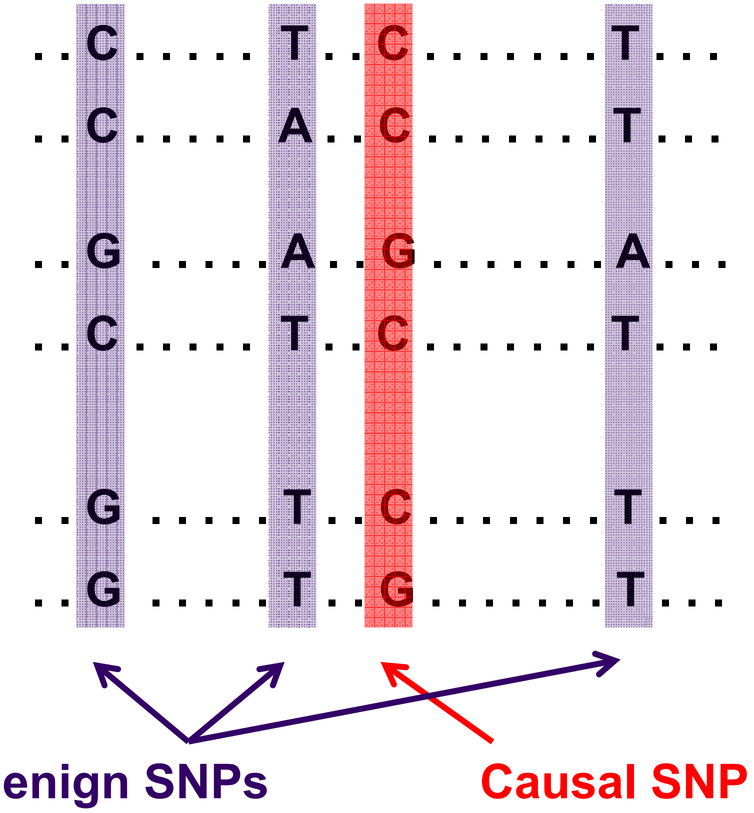


a univariate phenotype:

i.e., the expression intensity of a gene

Association Mapping as Regression

	Phenotype (BMI)	Genotype
Individual 1	2.5	..C......T..C.........T... ..C......A..C.........T...
Individual 2	4.8	..G .....A..G........A... ..C......T..C.........T...
⋮		
Individual N	4.7	..G .....T..C.........T... ..G .....T..G........T...



Association Mapping as Regression

	Phenotype (BMI)	Genotype
Individual 1	2.5	. . 0 . . . . . 1 . . 0 . . . . 0 . . .
Individual 2	4.8	. . 1 . . . . . 1 . . 1 . . . . 1 . . .
⋮		
Individual N	4.7	. . 2 . . . . . 2 . . 1 . . . . 0 . . .

⬇ ⬇

$$\mathbf{y}_i = \sum_{j=1}^{J} x_{ij}\beta_j$$

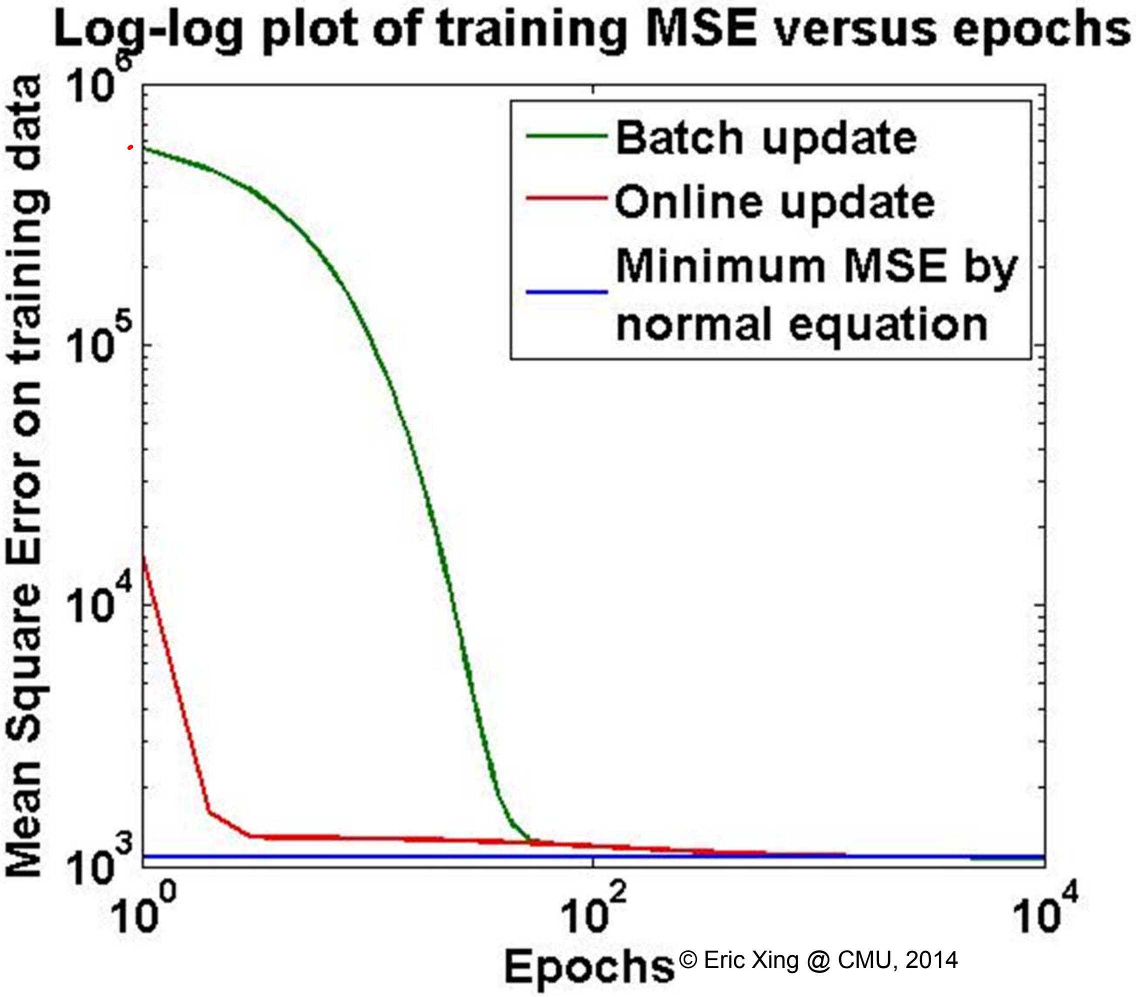
SNPs with large $|\beta_j|$ are relevant

Experimental setup

- Asthama dataset
  - 543 individuals, genotyped at 34 SNPs
  - Diploid data was transformed into 0/1 (for homozygotes) or 2 (for heterozygotes)
  - X=543x34 matrix
  - Y=Phenotype variable (continuous)
- A single phenotype was used for regression

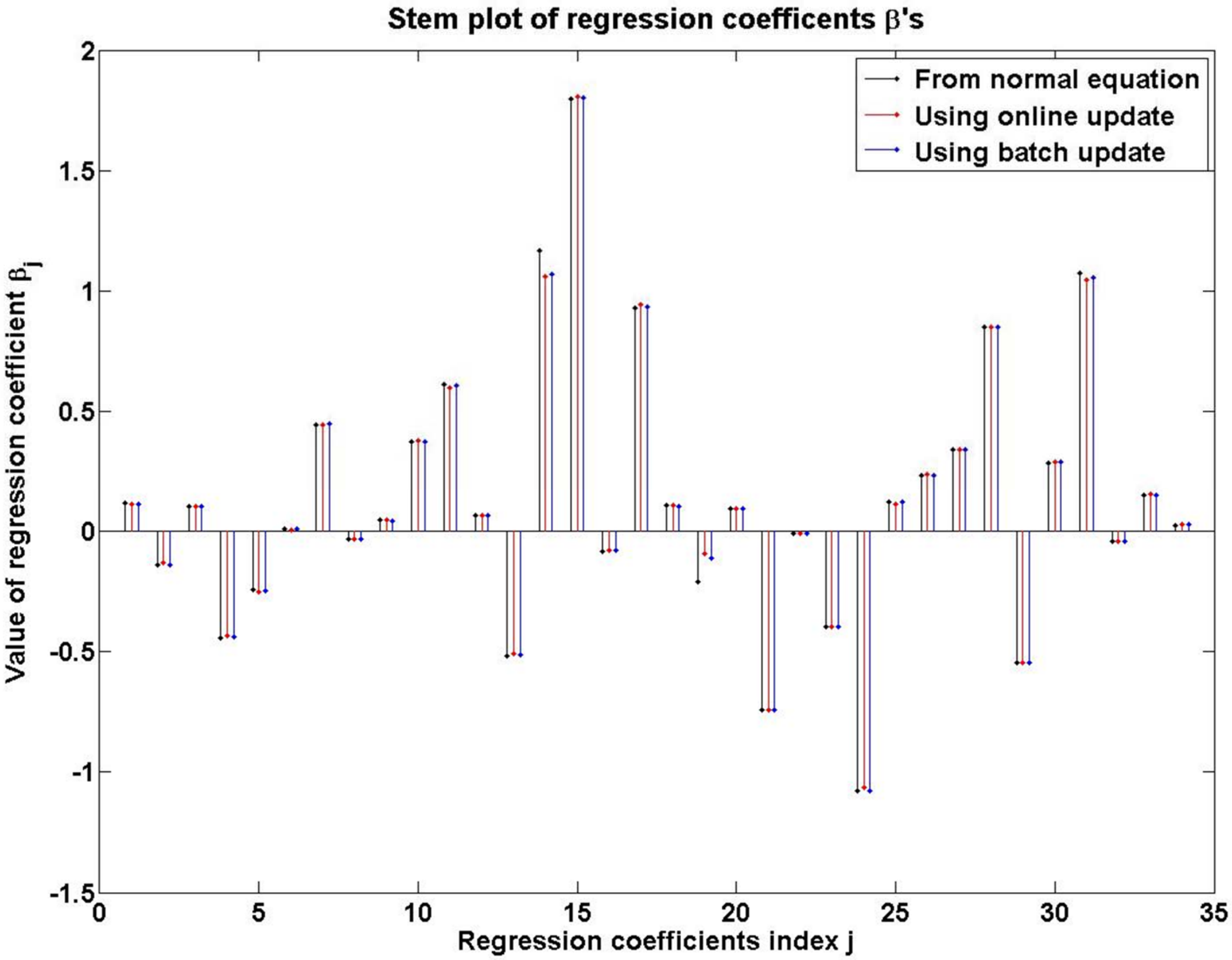
- Implementation details
  - Iterative methods: Batch update and online update implemented.
  - For both methods, step size α is chosen to be a small fixed value (10^{-6}). This choice is based on the data used for experiments.
  - Both methods are only run to a maximum of 2000 epochs or until the change in training MSE is less than 10^{-4}

Convergence Curves



- For the batch method, the training MSE is initially large due to uninformed initialization
- In the online update, N updates for every epoch reduces MSE to a much smaller value.

The Learned Coefficients



Multivariate Regression for Trait Association Analysis

Trait		Genotype		Association Strength
2.1	=	TGAACCATGAAGTA	x	?
$\boldsymbol{y}$	=	$\boldsymbol{X}$	x	$\boldsymbol{\beta}$

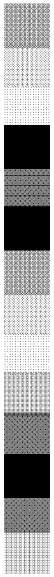
Multivariate Regression for Trait Association Analysis

Trait

Genotype

Association Strength

2.1 = TGAACCATGAAGTA x



$$\beta^* = \arg\min_{\beta} (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta)$$

Many non-zero associations:
Which SNPs are truly significant?

Sparsity

- One common assumption to make **sparsity.**

- **Makes biological sense:** each phenotype is likely to be associated with a small number of SNPs, rather than all the SNPs.

- **Makes statistical sense:** Learning is now feasible in high dimensions with small sample size

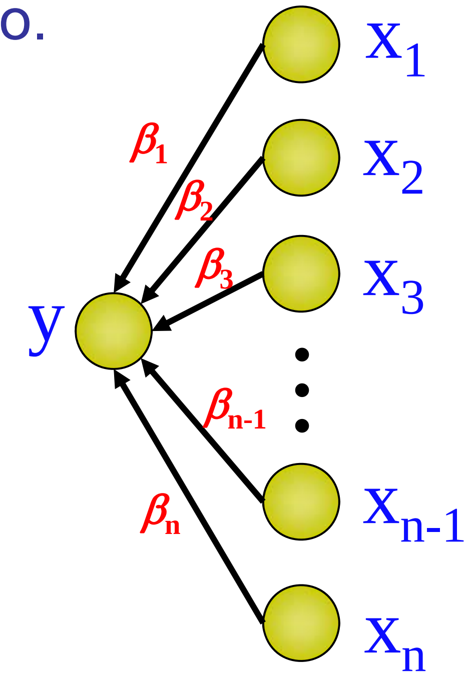
Sparsity: In a mathematical sense

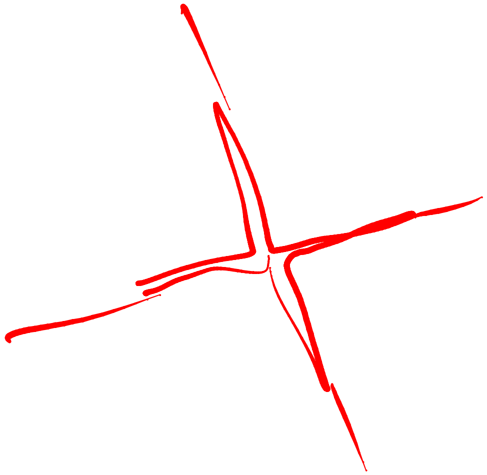
- Consider least squares linear regression problem:
- Sparsity means most of the beta's are zero.

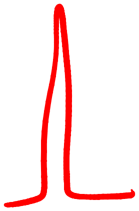
$$\hat{\boldsymbol{\beta}} = \text{argmin}_{\beta} ||\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}||^2$$

subject to:

$$\sum_{j=1}^{p} \mathbb{I}[|\beta_j| > 0] \leq C$$







- But this is not convex!!! Many local optima, computationally intractable.

L1 Regularization (LASSO)

(Tibshirani, 1996)

- A convex relaxation.

Constrained Form

$$\hat{\boldsymbol{\beta}} = \operatorname{argmin}_{\beta} \|\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\|^2$$

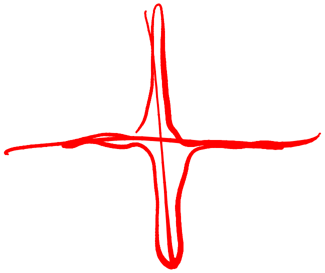
subject to:

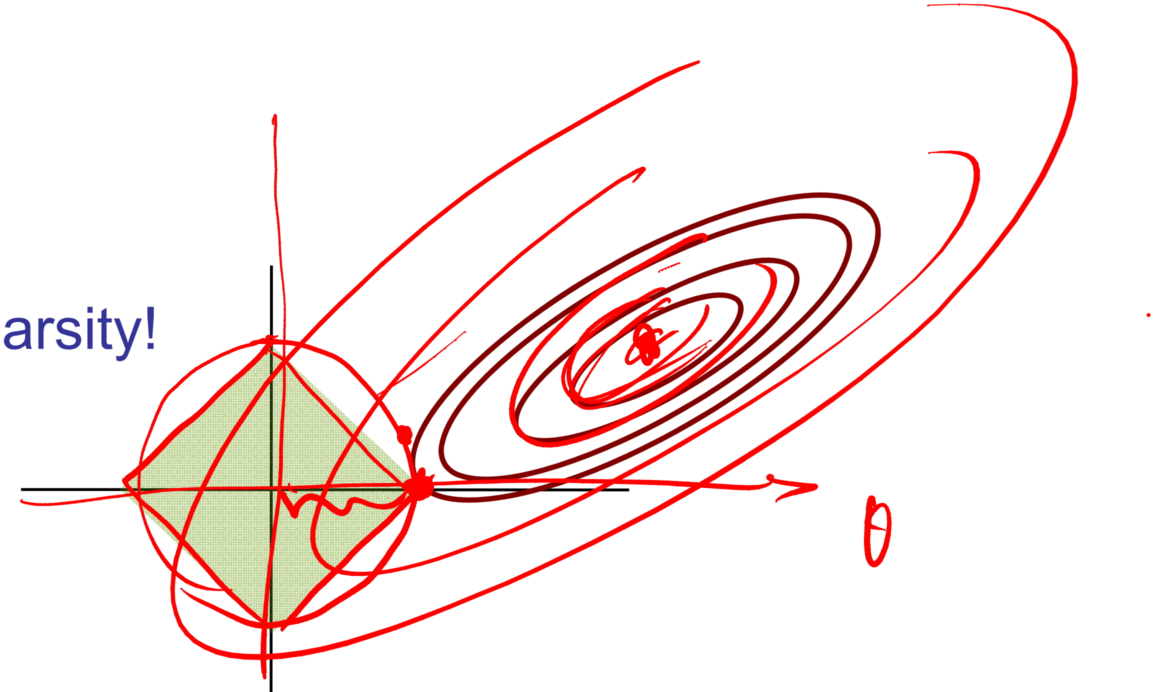
$$\sum_{j=1}^{p} |\beta_j| \leq C$$

Lagrangian Form

$$\hat{\boldsymbol{\beta}} = \operatorname{argmin}_{\beta} \|\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \|\boldsymbol{\beta}\|_1$$

- Still enforces sparsity!





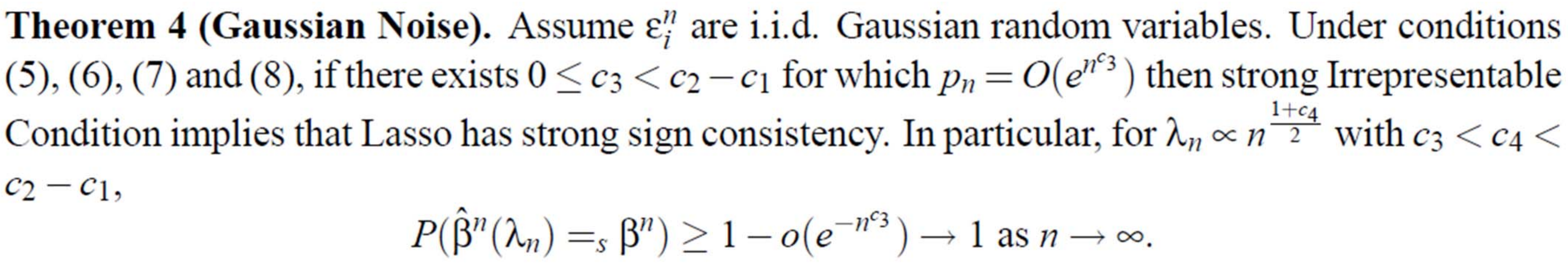
Theoretical Guarantees

- Assumptions
  - Dependency Condition: Relevant Covariates are not overly dependent
  - Incoherence Condition: Large number of irrelevant covariates cannot be too correlated with relevant covariates
  - Strong concentration bounds: Sample quantities converge to expected values quickly

If these are assumptions are met, LASSO will asymptotically recover correct subset of covariates that relevant.

Consistent Structure Recovery

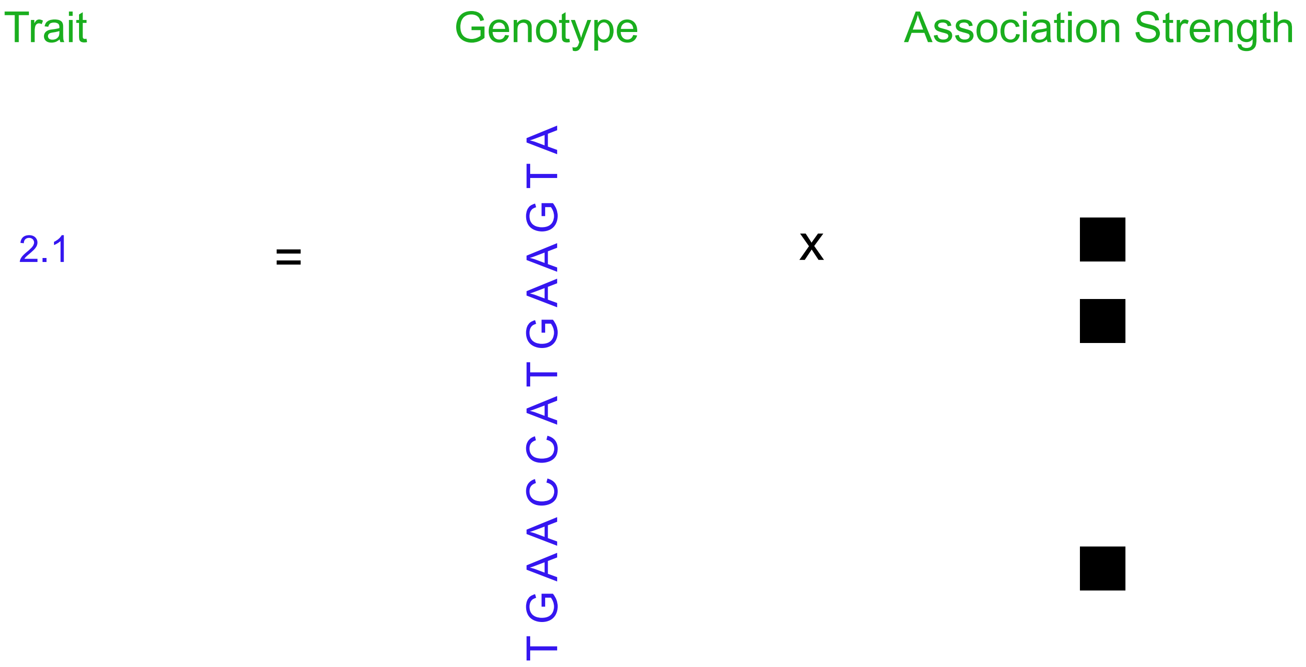
[Zhao and Yu 2006]



Theorem 4 (Gaussian Noise). Assume ε_i^n are i.i.d. Gaussian random variables. Under conditions (5), (6), (7) and (8), if there exists $0 \leq c_3 < c_2 - c_1$ for which $p_n = O(e^{n^{c_3}})$ then strong Irrepresentable Condition implies that Lasso has strong sign consistency. In particular, for $\lambda_n \propto n^{\frac{1+c_4}{2}}$ with $c_3 < c_4 < c_2 - c_1$,

$$P(\hat{\beta}^n(\lambda_n) =_s \beta^n) \geq 1 - o(e^{-n^{c_3}}) \rightarrow 1 \text{ as } n \rightarrow \infty.$$

Lasso for Reducing False Positives



Trait

Genotype

Association Strength

2.1 = TGAACCATGAAGTA x

Lasso Penalty for sparsity

$$\beta^* = \arg\min_{\beta} (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) + \lambda \sum_{j=1}^{J} |\beta_j|$$

Many zero associations (**sparse** results), but what if there are multiple related traits?

Ridge Regression vs Lasso

$$\min_{\beta}(\mathbf{X}\beta - \mathbf{Y})^T(\mathbf{X}\beta - \mathbf{Y}) + \lambda \text{pen}(\beta) = \min_{\beta} J(\beta) + \lambda \text{pen}(\beta)$$

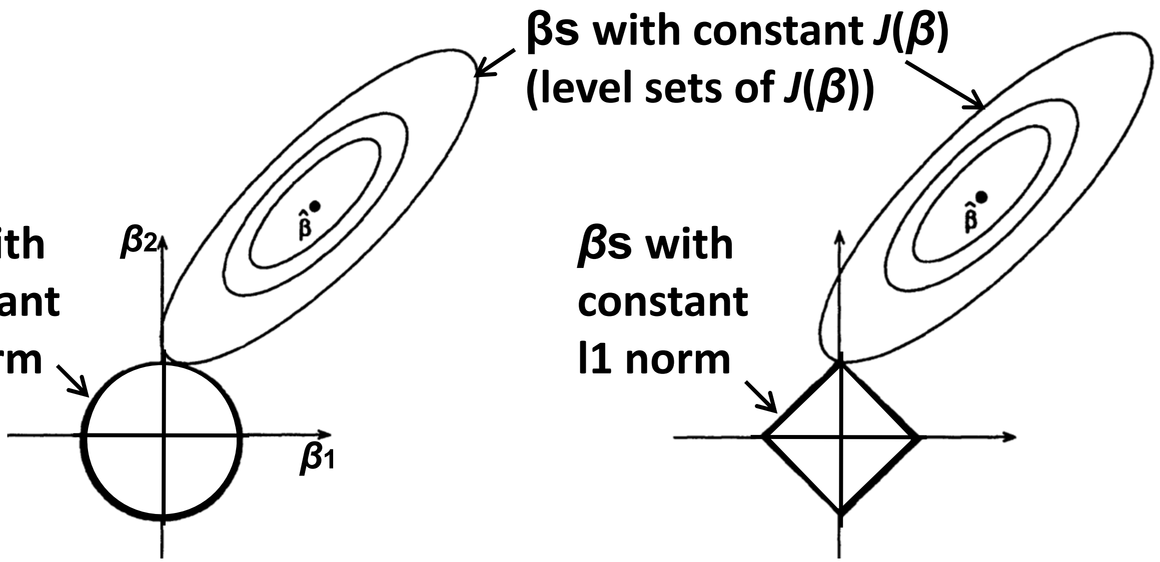
Ridge Regression:

$$\text{pen}(\beta) = \|\beta\|_2^2$$

Lasso:

$$\text{pen}(\beta) = \|\beta\|_1$$

HOT
!



βs with constant *J*(*β*)
(level sets of *J*(*β*))

βs with constant l2 norm

βs with constant l1 norm

Lasso (l1 penalty) results in sparse solutions – vector with more zero coordinates
Good for high-dimensional problems – don't have to store all coordinates!

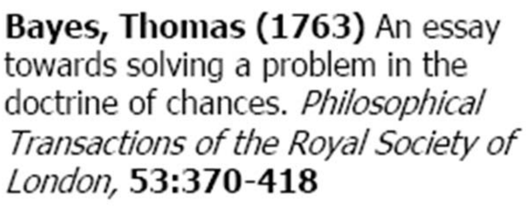
Bayesian Interpretation

- Treat the distribution parameters θ also as a *random variable*
- The *a posteriori* distribution of θ after seem the data is:

$$p(\theta \mid D) = \frac{p(D \mid \theta)\, p(\theta)}{p(D)} = \frac{p(D \mid \theta)\, p(\theta)}{\int p(D \mid \theta)\, p(\theta) d\theta}$$

This is Bayes Rule

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}$$



Bayes, Thomas (1763) An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, **53:370-418**

The prior p(.) encodes our prior knowledge about the domain

Regularized Least Squares and MAP

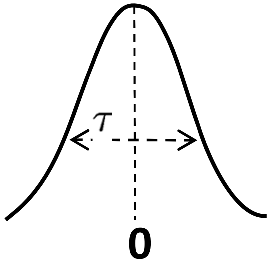
$\beta = \hat{\theta}$

What if (X^TX) is not invertible ?

$$\widehat{\beta}_{\text{MAP}} = \arg\max_{\beta} \underbrace{\log p(\{(X_i, Y_i)\}_{i=1}^{n} | \beta, \sigma^2)}_{\text{log likelihood}} + \underbrace{\log p(\beta)}_{\text{log prior}}$$

I) Gaussian Prior

$$\beta \sim \mathcal{N}(0, \tau^2\mathbf{I}) \qquad p(\beta) \propto e^{-\beta^T\beta/2\tau^2}$$



$$\widehat{\beta}_{\text{MAP}} = \arg\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \underset{\substack{\downarrow \\ \text{constant}(\sigma^2, \tau^2)}}{\lambda} \|\beta\|_2^2$$

Ridge Regression

Closed form: HW

Prior belief that β is Gaussian with zero-mean biases solution to "small" β

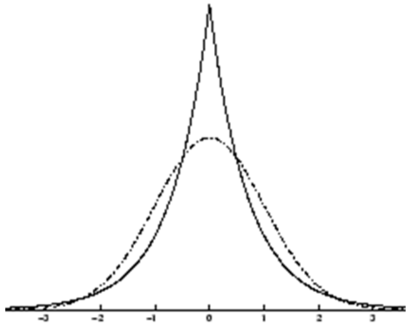
Regularized Least Squares and MAP

What if (X^TX) is not invertible ?

$$\widehat{\beta}_{\mathrm{MAP}} = \arg\max_{\beta} \underbrace{\log p(\{(X_i, Y_i)\}_{i=1}^{n} | \beta, \sigma^2)}_{\textbf{log likelihood}} + \underbrace{\log p(\beta)}_{\textbf{log prior}}$$

II) Laplace Prior

$$\beta_i \overset{iid}{\sim} \mathrm{Laplace}(0, t) \qquad p(\beta_i) \propto e^{-|\beta_i|/t}$$



$$\widehat{\beta}_{\mathrm{MAP}} = \arg\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \underset{\substack{\downarrow \\ \mathrm{constant}(\sigma^2, t)}}{\lambda} \|\beta\|_1$$

Lasso

Closed form: HW

Prior belief that β is Laplace with zero-mean biases solution to “small” β

Take home message

- Gradient descent
  - On-line
  - Batch
- Normal equations
- Geometric interpretation of LMS
- Probabilistic interpretation of LMS, and equivalence of LMS and MLE under certain assumption (what?)
- Sparsity:
  - Approach: ridge vs. lasso regression
  - Interpretation: regularized regression versus Bayesian regression
  - Algorithm: convex optimization (we did not discuss this)
- LR does not mean fitting linear relations, but linear combination or basis functions (that can be non-linear)
- Weighting points by importance versus by fitness

After class material …

Advanced Material: Beyond basic LR

- LR with non-linear basis functions

- Locally weighted linear regression

- Regression trees and Multilinear Interpolation

We will discuss this in next class after we set the state right!
(if we've got time ☺)

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

$$y = \theta_0 + \sum_{j=1}^{m} \theta_j \phi(x) = \theta^T \phi(x)$$

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

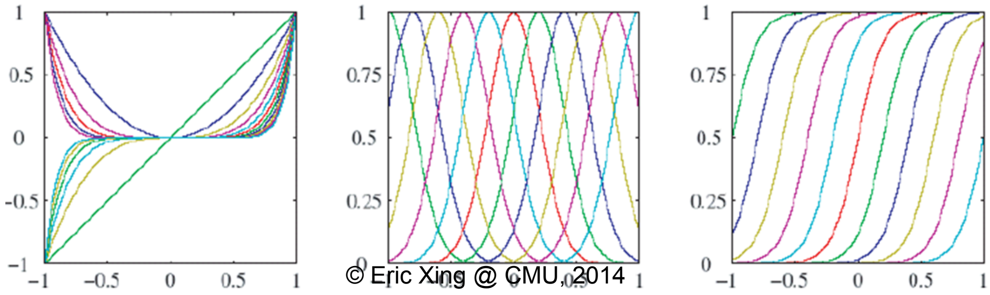
- Example: polynomial regression:

$$\phi(x) := \left[1, x, x^2, x^3\right]$$

- We will be concerned with estimating (distributions over) the weights θ and choosing the model order *M*.

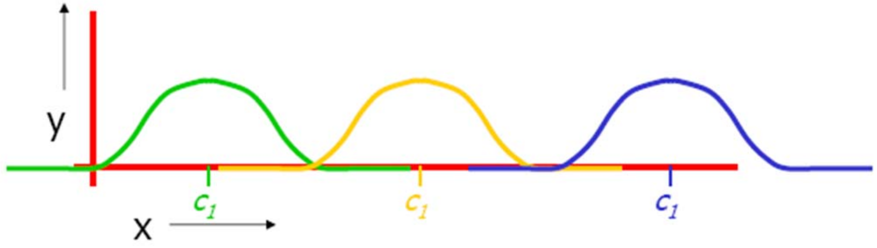
Basis functions

- There are many basis functions, e.g.:
  - Polynomial $\phi_j(x) = x^{j-1}$
  - Radial basis functions $\phi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2s^2}\right)$
  - Sigmoidal $\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$
  - Splines, Fourier, Wavelets, etc



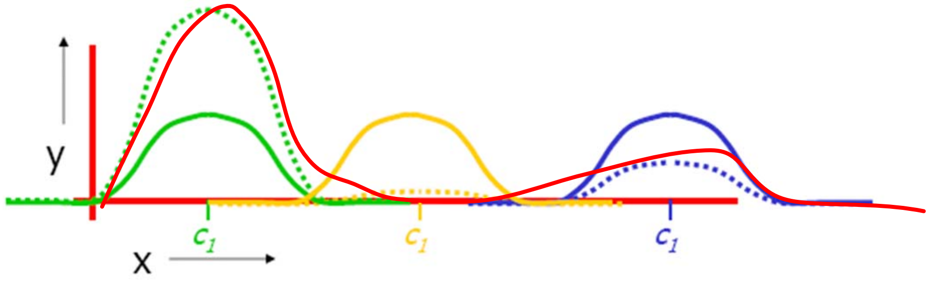
1D and 2D RBFs

- 1D RBF



$$y^{est} = \beta_1 \phi_1(x) + \beta_2 \phi_2(x) + \beta_3 \phi_3(x)$$

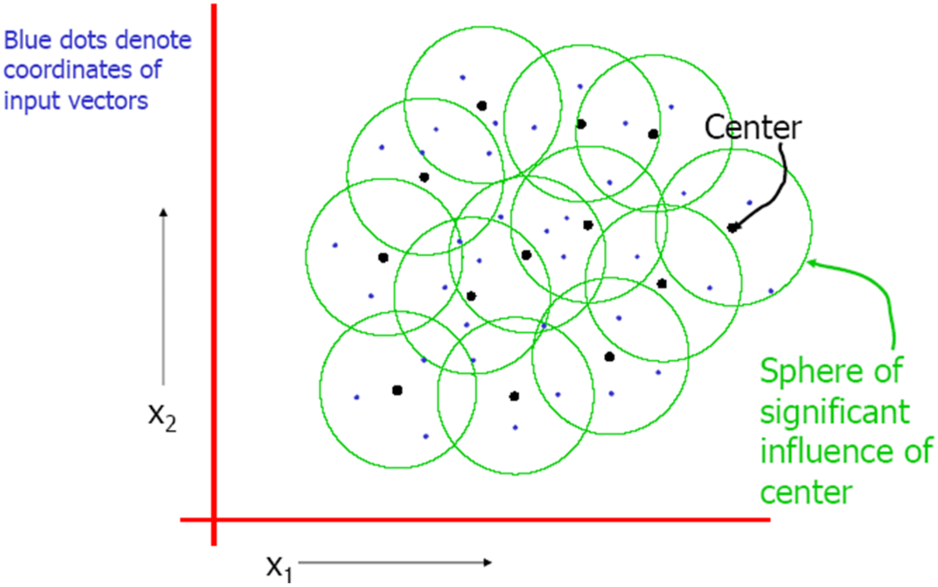
- After fit:



$$y^{est} = 2\phi_1(x) + 0.05\phi_2(x) + 0.5\phi_3(x)$$

Good and Bad RBFs

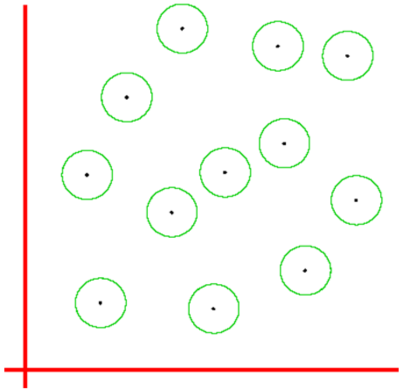
- A good 2D RBF

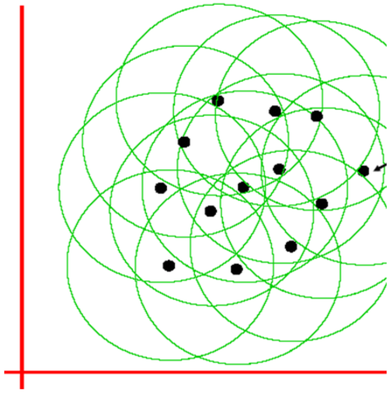


Blue dots denote coordinates of input vectors

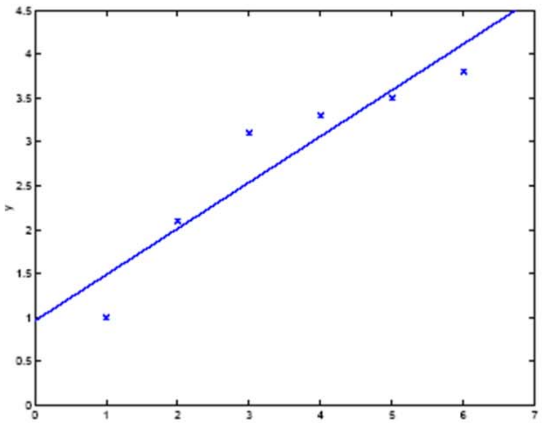
Sphere of significant influence of center

- Two bad 2D RBFs

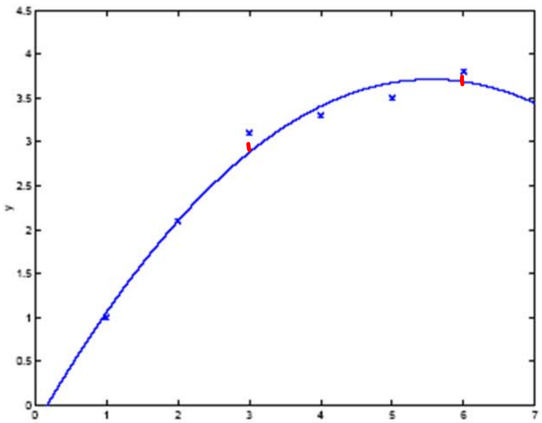




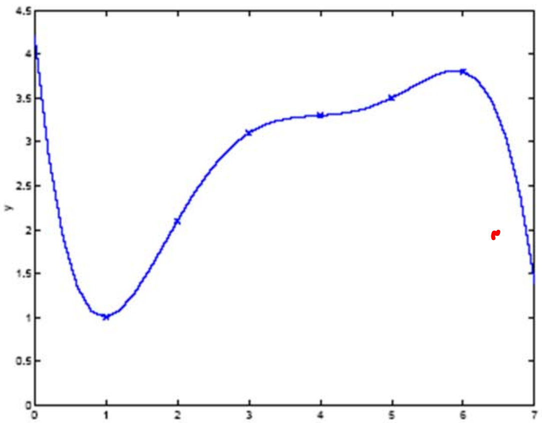
Overfitting and underfitting



$y = \theta_0 + \theta_1 x$



$y = \theta_0 + \theta_1 x + \theta_2 x^2$

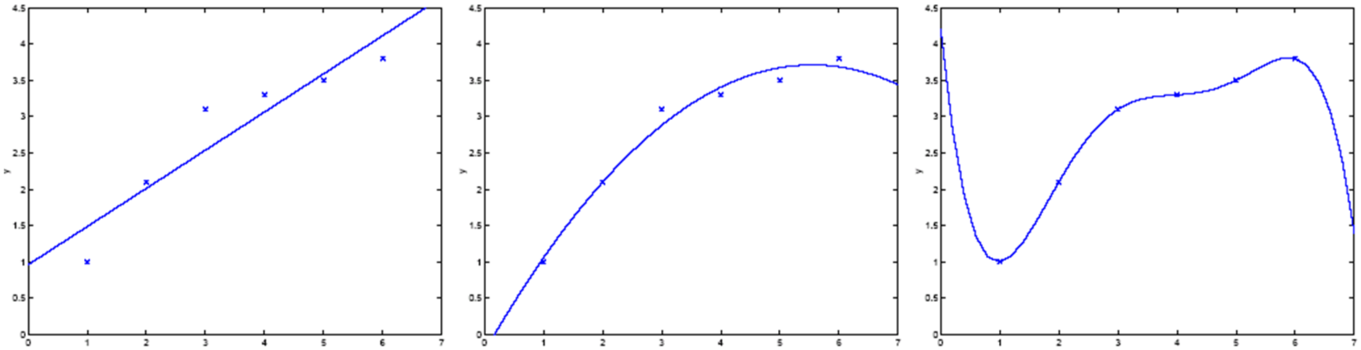


$y = \sum_{j=0}^{5} \theta_j x^j$

Bias and variance

- We define the bias of a model to be the expected generalization error even if we were to fit it to a very (say, infinitely) large training set.

- By fitting "spurious" patterns in the training set, we might again obtain a model with large generalization error. In this case, we say the model has large variance.



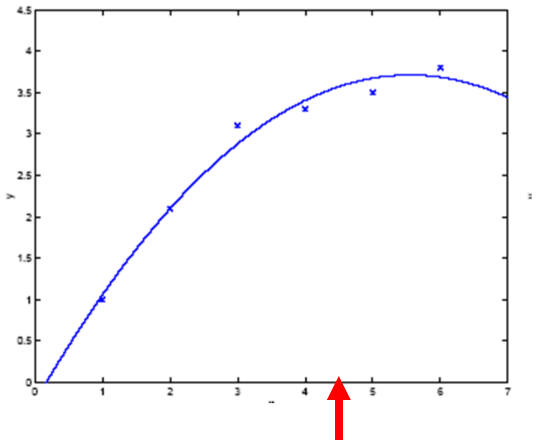
Locally weighted linear regression

- The algorithm:

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

  now we fit θ to minimize $J(\theta) = \frac{1}{2}\sum_{i=1}^{n}w_i(\mathbf{x}_i^T\theta - y_i)^2$

  Where do w_i's come from? $w_i = \exp\left(-\frac{(\mathbf{x}_i - \mathbf{x})^2}{2\tau^2}\right)$



  - where $\mathbf{x}$ is the query point for which we'd like to know its corresponding $\mathbf{y}$

→ Essentially we put higher weights on (errors on) training examples that are close to the query point (than those that are further away from the query)

Parametric vs. non-parametric

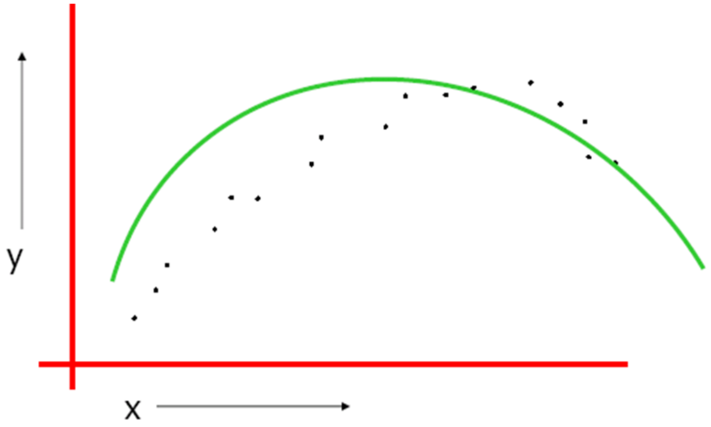
- Locally weighted linear regression is the second example we are running into of a **non-parametric** algorithm. (what is the first?)

- The (unweighted) linear regression algorithm that we saw earlier is known as a **parametric** learning algorithm
  - because it has a fixed, finite number of parameters (the θ), which are fit to the data;
  - Once we've fit the θ and stored them away, we no longer need to keep the training data around to make future predictions.
  - In contrast, to make predictions using locally weighted linear regression, we need to keep the entire training set around.

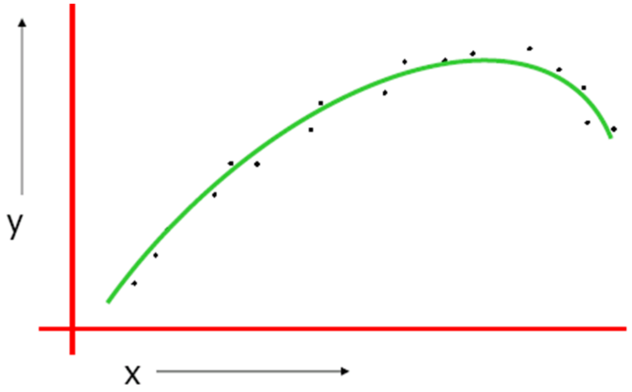
- The term "non-parametric" (roughly) refers to the fact that the amount of stuff we need to keep in order to represent the hypothesis grows linearly with the size of the training set.

Robust Regression

- The best fit from a quadratic regression



- But this is probably better …



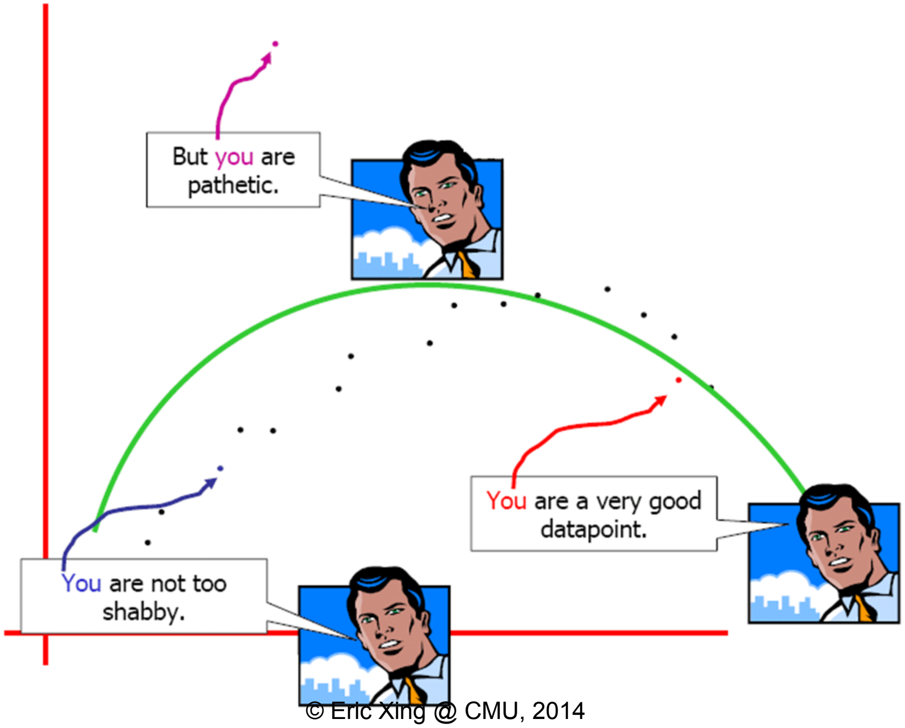
How can we do this?

LOESS-based Robust Regression

- Remember what we do in "locally weighted linear regression"?

  → we "score" each point for its impotence

- Now we score each point according to its "fitness"

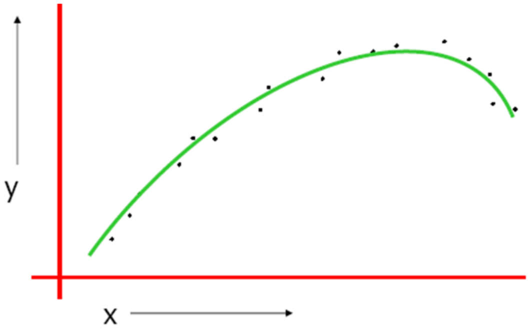


(Courtesy to Andrew Moor)

Robust regression

- For k = 1 to R…
  - Let (x_k, y_k) be the kth datapoint
  - Let y^{est}_k be predicted value of y_k
  - Let w_k be a weight for data point k that is large if the data point fits well and small if it fits badly:

$$w_k = \phi\left((y_k - y_k^{\text{est}})^2\right)$$



- Then redo the regression using weighted data points.
- Repeat whole thing until converged!

Robust regression—probabilistic interpretation

- **What regular regression does:**

Assume y_k was originally generated using the following recipe:

$$y_k = \theta^T \mathbf{x}_k + \mathcal{N}(0, \sigma^2)$$

Computational task is to find the Maximum Likelihood estimation of θ

Robust regression—probabilistic interpretation

- **What LOESS robust regression does:**

  Assume y_k was originally generated using the following recipe:

  with probability p: $\quad y_k = \theta^T \mathbf{x}_k + \mathcal{N}(0, \sigma^2)$

  but otherwise $\quad y_k \sim \mathcal{N}(\mu, \sigma^2_{\text{huge}})$

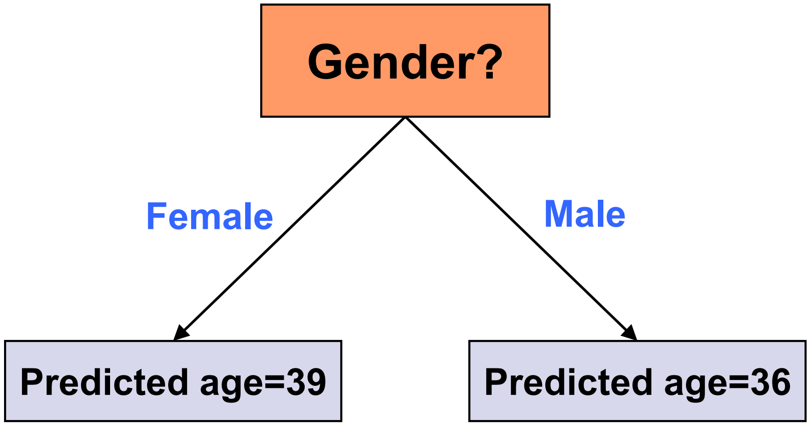
  Computational task is to find the Maximum Likelihood estimates of θ, p, μ and σ_{huge}.

- The algorithm you saw with iterative **reweighting/refitting** does this computation for us. Later you will find that it is an instance of the famous **E.M.** algorithm

Regression Tree

- Decision tree for regression

Gender	Rich?	Num. Children	# travel per yr.	Age
F	No	2	5	38
M	No	0	2	25
M	Yes	1	0	72
:	:	:	:	:



Gender?

Female

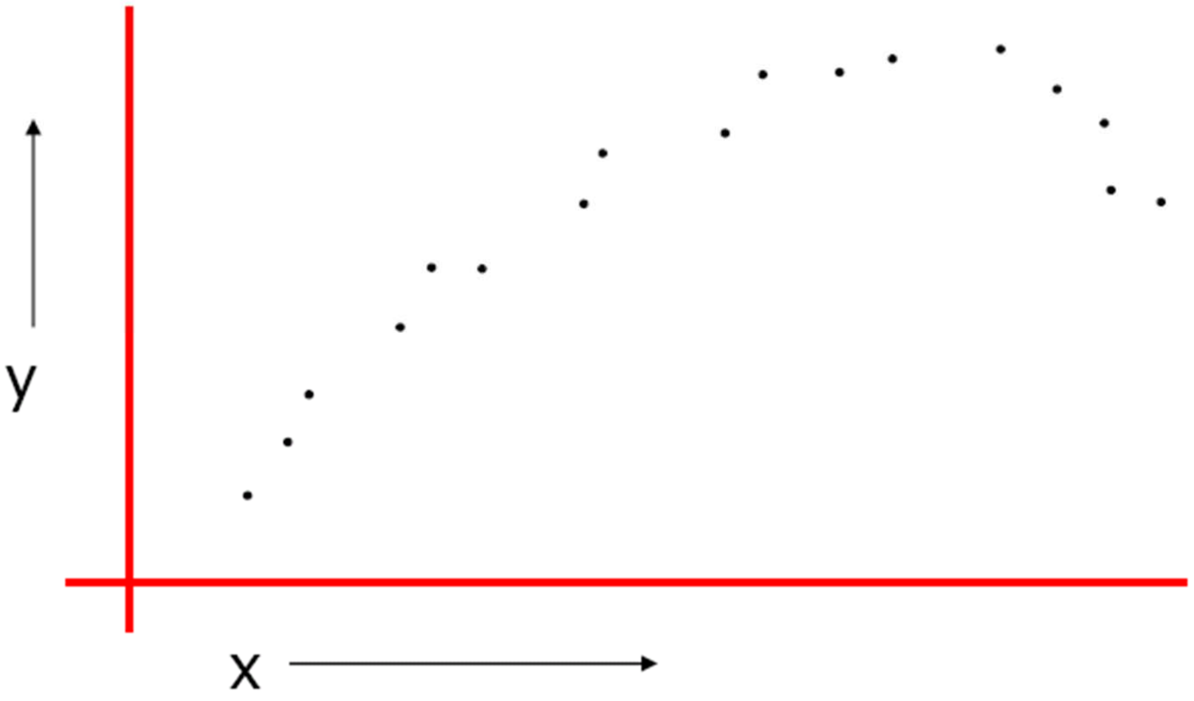
Male

Predicted age=39

Predicted age=36

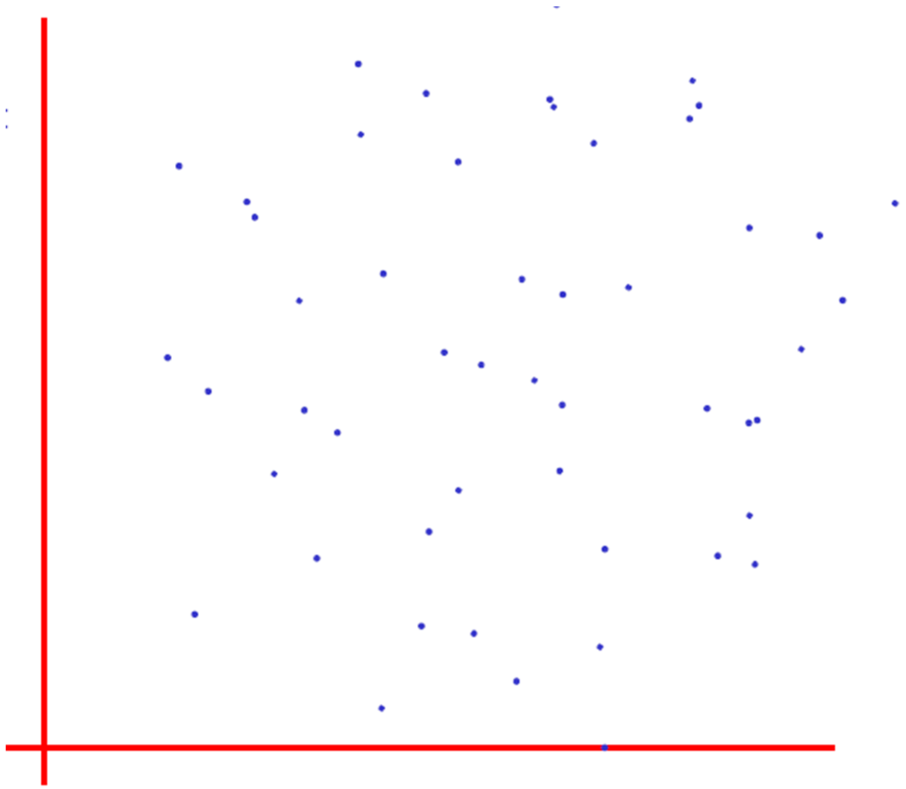
A conceptual picture

- Assuming regular regression trees, can you sketch a graph of the fitted function y*(x) over this diagram?



How about this one?

- Multilinear Interpolation



- We wanted to create a continuous and piecewise linear fit to the data

Take home message

- Gradient descent
  - On-line
  - Batch
- Normal equations
- Geometric interpretation of LMS
- Probabilistic interpretation of LMS, and equivalence of LMS and MLE under certain assumption (what?)
- Sparsity:
  - Approach: ridge vs. lasso regression
  - Interpretation: regularized regression versus Bayesian regression
  - Algorithm: convex optimization (we did not discuss this)
- LR does not mean fitting linear relations, but linear combination or basis functions (that can be non-linear)
- Weighting points by importance versus by fitness

Appendix

Parameter Learning from *iid* Data

- Goal: estimate distribution parameters $\boldsymbol{\theta}$ from a dataset of N independent, identically distributed (*iid*), fully observed, training cases

$$D = \{x_1, \ldots, x_N\}$$

- Maximum likelihood estimation (MLE)
  1. One of the most common estimators
  2. With iid and full-observability assumption, write $L(\theta)$ as the likelihood of the data:

$$\begin{aligned} L(\theta) &= P(x_{1,} x_2, \ldots, x_N; \theta) \\ &= P(x;\theta)P(x_2;\theta), \ldots, P(x_N;\theta) \\ &= \prod_{i=1}^{N} P(x_i;\theta) \end{aligned}$$

  3. pick the setting of parameters most likely to have generated the data we saw:

$$\theta^* = \arg\max_\theta L(\theta) \;\; = \arg\max_\theta \log L(\theta)$$

Example: Bernoulli model

- Data:
  - We observed *N* ***iid*** coin tossing: *D*={1, 0, 1, …, 0}
- Representation:

  Binary r.v: $$x_n = \{0,1\}$$

- Model: $$P(x) = \begin{cases} 1-\theta & \text{for } x=0 \\ \theta & \text{for } x=1 \end{cases} \quad \Rightarrow \quad P(x) = \theta^x (1-\theta)^{1-x}$$

- How to write the likelihood of a single observation x_i ?

$$P(x_i) = \theta^{x_i}(1-\theta)^{1-x_i}$$

- The likelihood of dataset *D*={*x*₁, …,*x*_N}:

$$P(x_1, x_2, \ldots, x_N \mid \theta) = \prod_{i=1}^{N} P(x_i \mid \theta) = \prod_{i=1}^{N} \left(\theta^{x_i}(1-\theta)^{1-x_i}\right) = \theta^{\sum_{i=1}^{N} x_i} (1-\theta)^{\sum_{i=1}^{N} 1-x_i} = \theta^{\#\text{head}}(1-\theta)^{\#\text{tails}}$$

Maximum Likelihood Estimation

- Objective function:

$$\ell(\theta; D) = \log P(D \mid \theta) = \log \theta^{n_h} (1-\theta)^{n_t} = n_h \log \theta + (N - n_h) \log(1-\theta)$$

- We need to maximize this w.r.t. θ

- Take derivatives wrt θ

$$\frac{\partial \ell}{\partial \theta} = \frac{n_h}{\theta} - \frac{N-n_h}{1-\theta} = 0 \quad \Rightarrow \quad \widehat{\theta}_{MLE} = \frac{n_h}{N} \quad \textbf{or} \quad \widehat{\theta}_{MLE} = \frac{1}{N}\sum_i x_i$$

Frequency as sample mean

- Sufficient statistics
  - The counts, n_h, where $n_k = \sum_i x_i$, are **sufficient statistics** of data *D*

Overfitting

- Recall that for Bernoulli Distribution, we have

$$\widehat{\theta}_{ML}^{head} = \frac{n^{head}}{n^{head} + n^{tail}}$$

- What if we tossed too few times so that we saw zero head?

  We have $\widehat{\theta}_{ML}^{head} = 0,$ and we will predict that the probability of seeing a head next is zero!!!

- The rescue: *"smoothing"*
  - Where n' is know as the pseudo- (imaginary) count

$$\widehat{\theta}_{ML}^{head} = \frac{n^{head} + n'}{n^{head} + n^{tail} + n'}$$

  - But can we make this more formal?

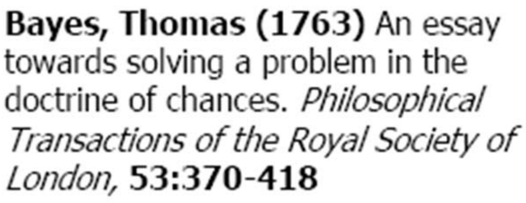
Bayesian Parameter Estimation

- Treat the distribution parameters *θ* also as a *random variable*
- The *a posteriori* distribution of *θ* after seem the data is:

$$p(\theta \mid D) = \frac{p(D \mid \theta)\, p(\theta)}{p(D)} = \frac{p(D \mid \theta)\, p(\theta)}{\int p(D \mid \theta)\, p(\theta) d\theta}$$

This is Bayes Rule

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}$$



Bayes, Thomas (1763) An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, **53:370-418**

The prior p(.) encodes our prior knowledge about the domain

Frequentist Parameter Estimation

Two people with different priors $p(\theta)$ will end up with different estimates $p(\theta|D)$.

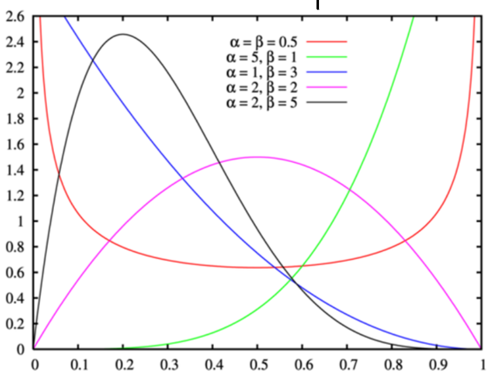
- Frequentists dislike this “subjectivity”.
- Frequentists think of the parameter as a fixed, unknown constant, not a random variable.
- Hence they have to come up with different "objective" **estimators** (ways of computing from data), instead of using Bayes’ rule.
  - These estimators have different properties, such as being “unbiased”, “minimum variance”, etc.
  - The maximum likelihood estimator, is one such estimator.

Discussion

θ or $p(\theta)$, this is the problem!

Bayesian estimation for Bernoulli

- Beta distribution:



$$P(\theta;\alpha,\beta)=\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}=B(\alpha,\beta)\theta^{\alpha-1}(1-\theta)^{\beta-1}$$

  - When x is discrete $\Gamma(x+1)=x\Gamma(x)=x!$

- Posterior distribution of θ :

$$P(\theta\,|\,x_1,\ldots,x_N)=\frac{p(x_1,\ldots,x_N\,|\,\theta)\,p(\theta)}{p(x_1,\ldots,x_N)}\propto\theta^{n_h}(1-\theta)^{n_t}\times\theta^{\alpha-1}(1-\theta)^{\beta-1}=\theta^{n_h+\alpha-1}(1-\theta)^{n_t+\beta-1}$$

  - Notice the isomorphism of the posterior to the prior,
  - such a prior is called a **conjugate prior**
  - α *and* β are hyperparameters (parameters of the prior) and correspond to the number of “virtual” heads/tails (pseudo counts)

Bayesian estimation for Bernoulli, con'd

- Posterior distribution of θ:

$$P(\theta | x_1,\dots,x_N) = \frac{p(x_1,\dots,x_N | \theta)\, p(\theta)}{p(x_1,\dots,x_N)} \propto \theta^{n_h}(1-\theta)^{n_t} \times \theta^{\alpha-1}(1-\theta)^{\beta-1} = \theta^{n_h+\alpha-1}(1-\theta)^{n_t+\beta-1}$$

- Maximum *a posteriori* (MAP) estimation:

$$\theta_{MAP} = \arg\max_{\theta} \log P(\theta | x_1,\dots,x_N)$$

- Posterior mean estimation:

Bata parameters can be understood as pseudo-counts

$$\theta_{Bayes} = \int \theta p(\theta | D) d\theta = C \int \theta \times \theta^{n_h+\alpha-1}(1-\theta)^{n_t+\beta-1} d\theta = \frac{n_h + \alpha}{N + \alpha + \beta}$$

- Prior strength: A=α+β
  - A can be interoperated as the size of an imaginary data set from which we obtain the **pseudo-counts**

Effect of Prior Strength

- Suppose we have a uniform prior (α=β=1/2), and we observe $\vec{n} = (n_h = 2, n_t = 8)$
- Weak prior A = 2. Posterior prediction:

$$p(x = h \mid n_h = 2, n_t = 8, \vec{\alpha} = \vec{\alpha}' \times 2) = \frac{1+2}{2+10} = 0.25$$

- Strong prior A = 20. Posterior prediction:

$$p(x = h \mid n_h = 2, n_t = 8, \vec{\alpha} = \vec{\alpha}' \times 20) = \frac{10+2}{20+10} = 0.40$$

- However, if we have enough data, it washes away the prior. e.g., $\vec{n} = (n_h = 200, n_t = 800)$. Then the estimates under weak and strong prior are $\frac{1+200}{2+1000}$ and $\frac{10+200}{20+1000}$, respectively, both of which are close to 0.2

Example 2: Gaussian density

- Data:
  - We observed *N* ***iid*** real samples:
    D={-0.1, 10, 1, -5.2, …, 3}
- Model: $P(x) = \left(2\pi\sigma^2\right)^{-1/2} \exp\left\{-(x-\mu)^2 / 2\sigma^2\right\}$
- Log likelihood:

$$\ell(\theta; D) = \log P(D \mid \theta) = -\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2}\sum_{n=1}^{N}\frac{(x_n-\mu)^2}{\sigma^2}$$

- MLE: take derivative and set to zero:

$$\frac{\partial \ell}{\partial \mu} = (1/\sigma^2)\sum_n (x_n - \mu)$$

$$\frac{\partial \ell}{\partial \sigma^2} = -\frac{N}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_n (x_n-\mu)^2$$

$$\Rightarrow$$

$$\mu_{MLE} = \frac{1}{N}\sum_n (x_n)$$

$$\sigma^2_{MLE} = \frac{1}{N}\sum_n (x_n - \mu_{ML})^2$$

MLE for a multivariate-Gaussian

- It can be shown that the MLE for *μ and* Σ is

$$x_n = \begin{pmatrix} x_n^1 \\ x_n^2 \\ \vdots \\ x_n^K \end{pmatrix}$$

$$\mu_{MLE} = \frac{1}{N}\sum_n (x_n)$$

$$\Sigma_{MLE} = \frac{1}{N}\sum_n (x_n - \mu_{ML})(x_n - \mu_{ML})^T = \frac{1}{N} S$$

$$X = \begin{pmatrix} --- x_1^T --- \\ --- x_2^T --- \\ \vdots \\ --- x_N^T --- \end{pmatrix}$$

where the scatter matrix is

$$S = \sum_n (x_n - \mu_{ML})(x_n - \mu_{ML})^T = \left(\sum_n x_n x_{n\;n}^T\right) - N\mu_{ML}\mu_{ML}^T$$

- The sufficient statistics are $\Sigma_n x_n$ and $\Sigma_n x_n x_n^T$.
- Note that $X^T X = \Sigma_n x_n x_n^T$ may not be full rank (eg. if $N < D$), in which case Σ_{ML} is not invertible

Bayesian estimation

- Normal Prior:

$$P(\mu) = \left(2\pi\sigma_0^2\right)^{-1/2} \exp\left\{-(\mu-\mu_0)^2 / 2\sigma_0^2\right\}$$

- Joint probability:

$$P(x,\mu) = \left(2\pi\sigma^2\right)^{-N/2} \exp\left\{-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2\right\}$$
$$\times\left(2\pi\sigma_0^2\right)^{-1/2} \exp\left\{-(\mu-\mu_0)^2 / 2\sigma_0^2\right\}$$

- Posterior:

$$P(\mu\,|\,x) = \left(2\pi\tilde{\sigma}^2\right)^{-1/2} \exp\left\{-(\mu-\tilde{\mu})^2 / 2\tilde{\sigma}^2\right\}$$

$$\text{where} \quad \tilde{\mu} = \frac{N/\sigma^2}{N/\sigma^2 + 1/\sigma_0^2}\bar{x} + \frac{1/\sigma_0^2}{N/\sigma^2 + 1/\sigma_0^2}\mu_0\,, \quad \text{and} \quad \tilde{\sigma}^2 = \left(\frac{N}{\sigma^2} + \frac{1}{\sigma_0^2}\right)^{-1}$$

Sample mean

Bayesian estimation: unknown μ, known σ

$$\mu_N = \frac{N/\sigma^2}{N/\sigma^2 + 1/\sigma_0^2}\bar{x} + \frac{1/\sigma_0^2}{N/\sigma^2 + 1/\sigma_0^2}\mu_0 , \qquad \tilde{\sigma}^2 = \left(\frac{N}{\sigma^2} + \frac{1}{\sigma_0^2}\right)^{-1}$$

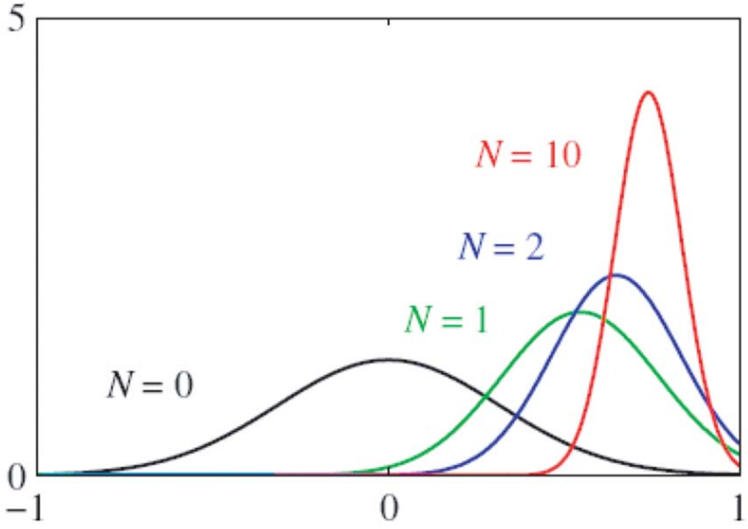
- The posterior mean is a convex combination of the prior and the MLE, with weights proportional to the relative noise levels.
- The precision of the posterior $1/\sigma^2_N$ is the precision of the prior $1/\sigma^2_0$ plus one contribution of data precision $1/\sigma^2$ for each observed data point.

- Sequentially updating the mean
  - $\mu*$ = 0.8 (unknown), $(\sigma^2)*$ = 0.1 (known)
  - Effect of single data point

$$\mu_1 = \mu_0 + (x-\mu_0)\frac{\sigma_0^2}{\sigma^2+\sigma_0^2} = x - (x-\mu_0)\frac{\sigma_0^2}{\sigma^2+\sigma_0^2}$$

  - Uninformative (vague/ flat) prior, $\sigma^2_0 \rightarrow \infty$

$$\mu_N \rightarrow \mu_0$$


5
0
−1
0
1
N = 10
N = 2
N = 1
N = 0