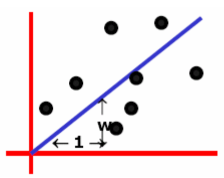
Machine Learning

10-701/15-781, Fall 2006

Introduction to Regression

Eric Xing



Lecture 3, September 19, 2006

Reading: Chap. 3, CB

Inference with the Joint

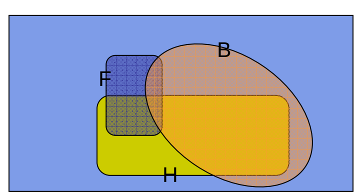
- Compute Conditionals

$$P(\text{Flu}|\text{Headhead}) = \frac{P(\text{Flu} \wedge \text{Headhead})}{P(\text{Headhead})}$$

$$=$$

¬F	¬B	¬H	0.4
¬F	¬B	H	0.1
¬F	B	¬H	0.17
¬F	B	H	0.2
F	¬B	¬H	0.05
F	¬B	H	0.05
F	B	¬H	0.015
F	B	H	0.015

- General idea: compute distribution on query variable by **fixing** **evidence variables** and **summing** over **hidden variables**



Conditional independence

- Write out full joint distribution using chain rule:

P(Headache;Flu;Virus;DrinkBeer)

= P(Headache | Flu;Virus;DrinkBeer) P(Flu;Virus;DrinkBeer)

= P(Headache | Flu;Virus;DrinkBeer) P(Flu | Virus;DrinkBeer) P(Virus | DrinkBeer) P(DrinkBeer)

Assume independence and conditional independence

= **P(Headache|Flu;DrinkBeer) P(Flu|Virus) P(Virus) P(DrinkBeer)**

I.e., 4 ? independent parameters 2 1 1

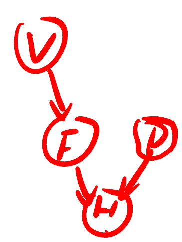
- In most cases, the use of conditional independence reduces the size of the representation of the joint distribution from **exponential** in *n* to **linear** in *n*.
- Conditional independence is our most basic and robust form of knowledge about uncertain environments.

Rules of Independence --- by examples

- P(Virus | DrinkBeer) = **P(Virus)**
  iff Virus is independent of DrinkBeer

- P(Flu | Virus;DrinkBeer) = **P(Flu|Virus)**
  iff Flu is independent of DrinkBeer, given Virus

- P(Headache | Flu;Virus;DrinkBeer) = **P(Headache|Flu;DrinkBeer)**
  iff Headache is independent of Virus, given Flu and DrinkBeer



Marginal and Conditional Independence

- Recall that for events E (i.e. $X=x$) and H (say, $Y=y$), the conditional probability of E given H, written as $P(E|H)$, is

  $$P(E \text{ and } H)/P(H)$$

  (= the probability of both E and H are true, given H is true)

- E and H are (statistically) independent if

  $$P(E) = P(E|H)$$

  (i.e., prob. E is true doesn't depend on whether H is true); or equivalently

  $$P(E \text{ and } H)=P(E)P(H).$$

- E and F are *conditionally* independent given H if

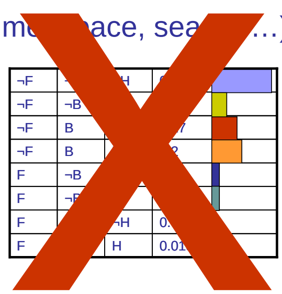
  $$P(E|H,F) = P(E|H)$$

  or equivalently

  $$P(E,F|H) = P(E|H)P(F|H)$$

Why knowledge of Independence is useful

- Lower complexity (time, space, search …)



- Motivates efficient inference for all kinds of queries

  Stay tuned !!

- Structured knowledge about the domain
  - easy to learning (both from expert and from data)
  - easy to grow

Where do probability distributions come from?

- Idea One: Human, Domain Experts
- Idea Two: Simpler probability facts and some algebra

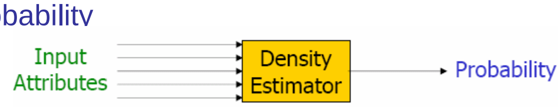
e.g., P(F)
P(B)
P(H|¬F,B)
P(H|F,¬B)
…

¬F	¬B	¬H	0.4
¬F	¬B	H	0.1
¬F	B	¬H	0.17
¬F	B	H	0.2
F	¬B	¬H	0.05
F	¬B	H	0.05
F	B	¬H	0.015
F	B	H	0.015

- Idea Three: Learn them from data!
  - A good chunk of this course is essentially about various ways of learning various forms of them!

Density Estimation

- A Density Estimator learns a mapping from a set of attributes to a Probability



- Often known as parameter estimation if the distribution form is specified
  - Binomial, Gaussian …
- Three important issues:
  - Nature of the data (iid, correlated, …)
  - Objective function (MLE, MAP, …)
  - Algorithm (simple algebra, gradient methods, EM, …)
  - Evaluation scheme (likelihood on test data, predictability, consistency, …)

Parameter Learning from iid data

- Goal: estimate distribution parameters θ 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 assumptions, 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 1: Bernoulli model

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

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

- Model: $P(x) = \begin{cases} 1-p & \text{for } x=0 \\ p & \text{for } x=1 \end{cases} \Rightarrow P(x) = \theta^x (1-\theta)^{1-x}$
- 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 $D=\{x_1, \ldots, x_N\}$:

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

MLE

- 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

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

Frequency as sample mean

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

MLE for discrete (joint) distributions

- More generally, it is easy to show that

$$P(\text{event}_i) = \frac{\#\text{records in which event}_i \text{ is true}}{\text{total number of records}}$$

- This is an important (but sometimes not so effective) learning algorithm!

¬F	¬B	¬H	0.4
¬F	¬B	H	0.1
¬F	B	¬H	0.17
¬F	B	H	0.2
F	¬B	¬H	0.05
F	¬B	H	0.05
F	B	¬H	0.015
F	B	H	0.015

Example 2: univariate normal

- 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:

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

$$\Rightarrow$$

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

Overfitting

- Recall that for Bernoulli Distribution, we have

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

- What if we tossed too few times so that we saw zero head?
  We have $\widehat{\theta}^{head}_{ML} = 0,$ and we will predict that the probability of seeing a head next is zero!!!
- The rescue:
  - Where n' is know as the pseudo- (imaginary) count

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

  - But can we make this more formal?

The Bayesian Theory

- The Bayesian Theory: (e.g., for date D and model M)

$$P(M|D) = P(D|M)P(M)/P(D)$$

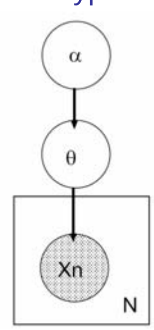
  - the **posterior** equals to the **likelihood** times the **prior**, up to a constant.

- This allows us to capture uncertainty about the model in a principled way

Hierarchical Bayesian Models

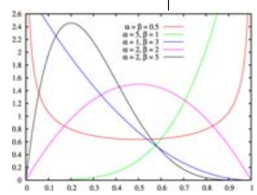
- θ are the parameters for the likelihood $p(x|\theta)$
- α are the parameters for the prior $p(\theta|\alpha)$.
- We can have hyper-hyper-parameters, etc.
- We stop when the choice of hyper-parameters makes no difference to the marginal likelihood; typically make hyper-parameters constants.
- Where do we get the prior?
  - Intelligent guesses
  - Empirical Bayes (Type-II maximum likelihood)
    → computing point estimates of α :

$$\hat{\vec{\alpha}}_{MLE} = \arg\max_{\vec{\alpha}} = p(\vec{n} \mid \vec{\alpha})$$



Bayesian estimation for Bernoulli

- Beta distribution:



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

- 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**

Bayesian estimation for Bernoulli, con'd

- Posterior distribution of θ:

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

- Maximum *a posteriori* (MAP) estimation:

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

- 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 ($\alpha=\beta=1/2\times$A), and we observe $\vec{n}=(n_h=2, n_t=8)$
- Weak prior A = 2. Posterior prediction:

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

- Strong prior A = 20. Posterior prediction:

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

- 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

Bayesian estimation for normal distribution

- Normal Prior:

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

- Joint probability:

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

- Posterior:

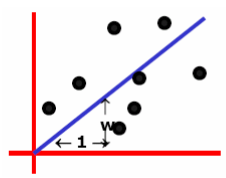
Homework!!!

Machine Learning

10-701/15-781, Fall 2006

Introduction to Regression

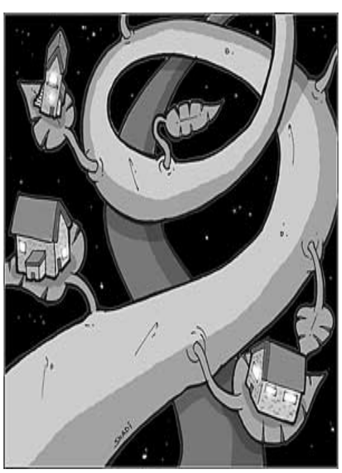
Eric Xing



Lecture 3, September 19, 2006

Reading: Chap. 3, CB

Machine learning for apartment hunting



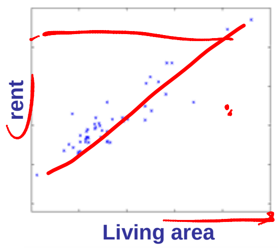
- Now you've moved to Pittsburgh!!

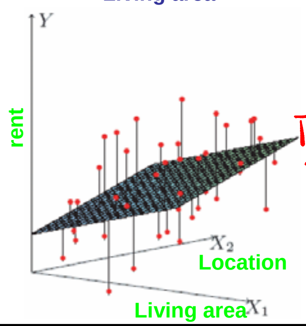
  And you want to find the **most reasonably priced** apartment satisfying your **needs:**

  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



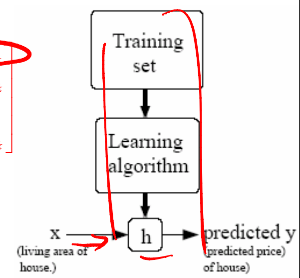


- Features:
  - Living area, distance to campus, # bedroom …
  - Denote as $\mathbf{x}=[x_1, x_2, \dots x_k]$
- Target:
  - Rent
  - Denoted as y
- Training set:

$$\mathbf{X} = \begin{bmatrix} -- & \mathbf{x}_1 & -- \\ -- & \mathbf{x}_2 & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_n & -- \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,k} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,k} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,k} \end{bmatrix}$$

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

Our goal:



Linear Regression

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

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

  - let's assume a vacuous "feature" X_0=1 (this is the intercept term, why?), and define the feature vector to be:

$$X = [1, x_1, x_2]^T$$

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

$$\hat{y} = x^T \theta$$

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

$$\vec{x}$$

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

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:

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

$x_i^T\theta$

$= \sum_k x_{i,k}\theta_k$

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

$= \theta_j^t - \alpha \sum_{i=1}^{n}(x_i^T\theta - y_i)\, x_{ij}$.

The Least-Mean-Square (LMS) method

- Now we have the following descent rule:

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

- For a single training point, we have:

$\theta_j^{t+1} = \theta_j^t + \alpha(y_i - x_i^T\theta^t)\, x_{i,j}$.

  - 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 an **on-line** algorithm

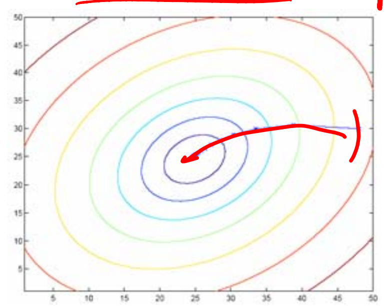
The Least-Mean-Square (LMS) method

- Steepest descent
  - Note that:

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

$x_{i,j}$

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



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

Some matrix derivatives

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

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

- Some fact of matrix derivatives (without proof)

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

The normal equations

- Write the cost function in matrix form:

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

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

- To minimize *J(θ)*, take derivative and set to zero:

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

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

The normal equations

$$\Downarrow$$

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

A recap:

- LMS update rule

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

  - Pros: on-line, low per-step cost
  - Cons: coordinate, maybe slow-converging

- Steepest descent

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

  - Pros: fast-converging, easy to implement
  - Cons: a batch,

- 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 $(X^TX)^{-1}$, expensive, numerical issues (e.g., matrix is singular ..)

Geometric Interpretation of LMS

- The predictions on the training data are:

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

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

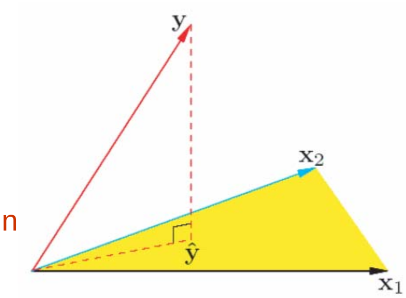
- Note that

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

and

$$X^T\left(\hat{\vec{y}} - \vec{y}\right) = X^T\left(X\left(X^T X\right)^{-1} X^T - I\right)\vec{y}$$
$$= \left(X^T X\left(X^T X\right)^{-1} X^T - X^T\right)\vec{y}$$
$$= 0 \; !$$

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

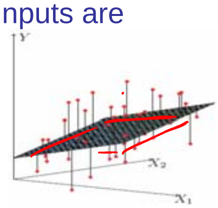


Probabilistic Interpretation of LMS

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

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

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:

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

- 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 !

Beyond basic LR

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

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

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

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

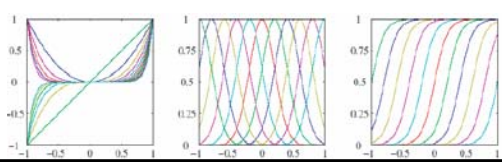
- Example: polynomial regression:

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

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

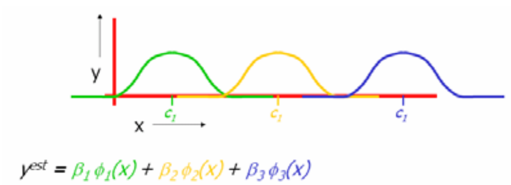
Basis functions

- 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

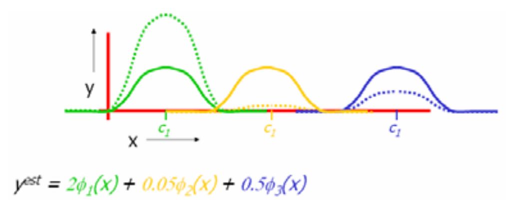


1D and 2D RBFs

- 1D RBF

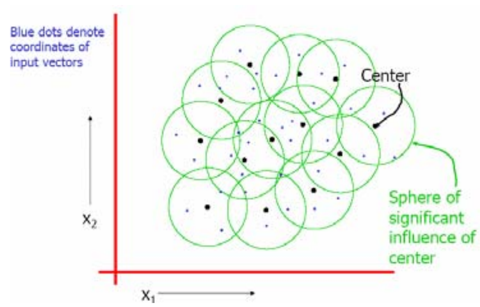


- After fit:

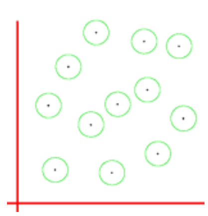


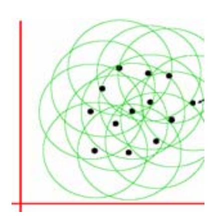
Good and Bad RBFs

- A good 2D RBF



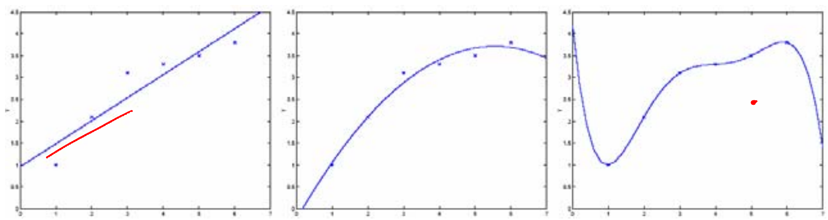
- Two bad 2D RBFs





Locally weighted linear regression

- Overfitting and underfitting



$y = \theta_0 + \theta_1 x$ $\quad$ $y = \theta_0 + \theta_1 x + \theta_2 x^2$ $\quad$ $y = \sum_{j=0}^{5} \theta_j x^j$

Locally weighted linear regression

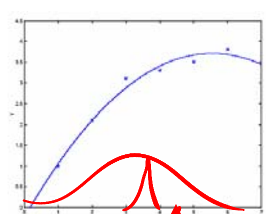
- The algorithm:

  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)

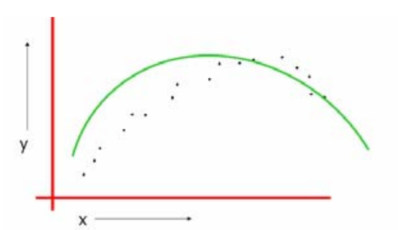
  - Do we also have a probabilistic interpretation here (as we did for LR)?

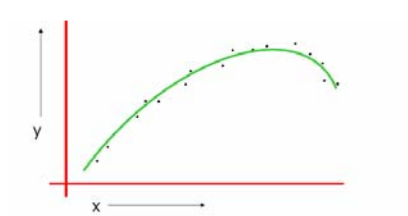
Parametric vs. non-parametric

- Locally weighted linear regression is the first example we are running into of a **non-parametric** algorithm.
- 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 …

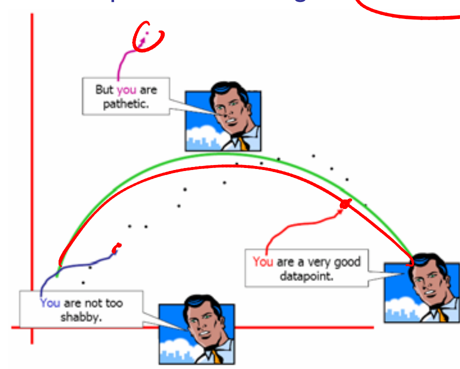




How can we do this?

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"

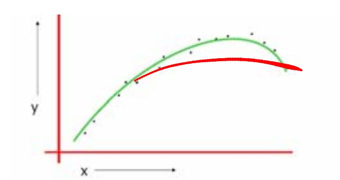


(Courtesy to Andrew Moor)

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:

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



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

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

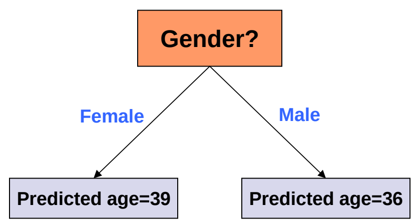
  Computational task is to find the Maximum Likelihood estimates of θ, p, μ and σ_{huge}.

- 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

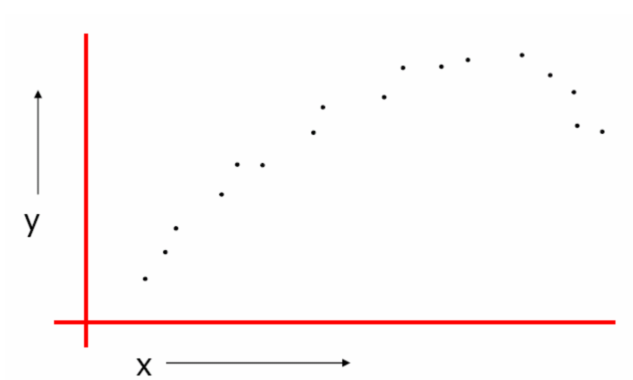
- Decision tree for regression

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



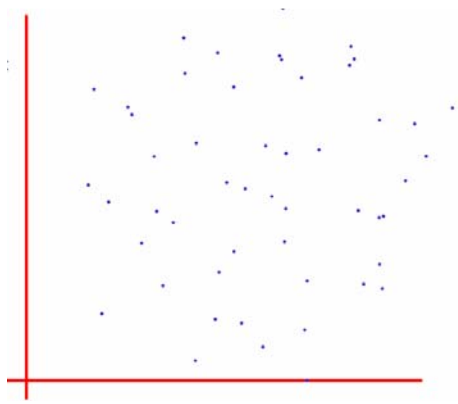
A conceptual picture

- 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
- Equivalence of LMS and MLE
- 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