Cmput 466 / 551

Linear Regression

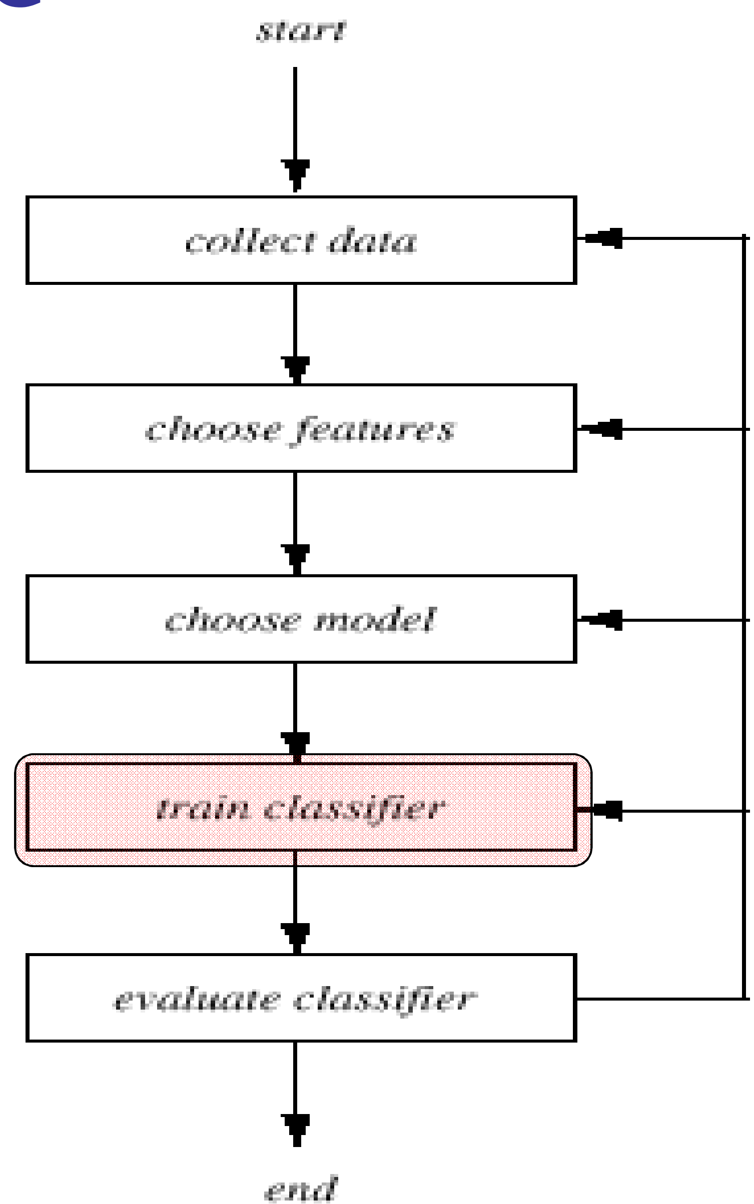
Covering chapters *HTF: Ch3, 7*

R Greiner
Department of Computing Science
University of Alberta

Thanks to: C Guestrin, T Dietterich, R Parr, H L Størvold, R Salakhutdinov

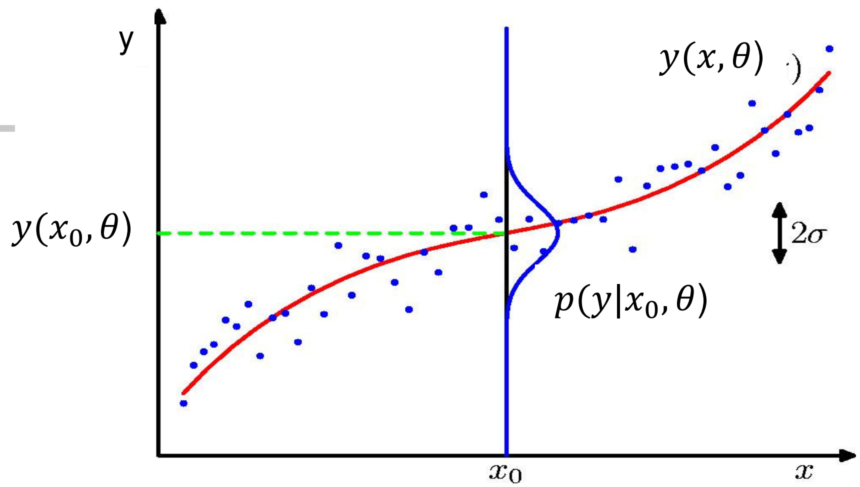
The Design Cycle

Most of ML course(s) focus on…



Outline

- Linear Regression
  - Iterative approach
    - Gradient Descent
  - Exact computation
  - Basis functions

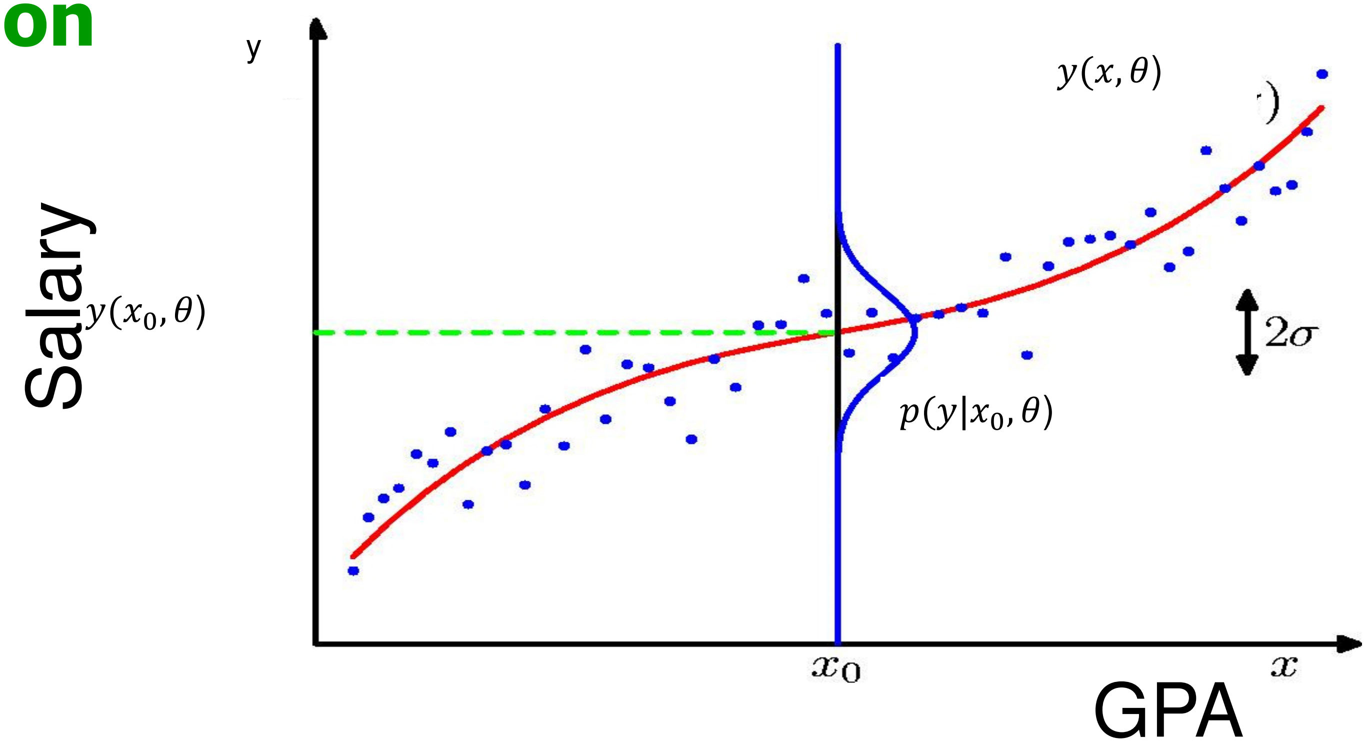


Prediction Problems …

- Predict *housing price* from:
  - House size, lot size, #rooms, neighborhood, location, location, location, …
- Predict person's *weight* from:
  - Gender, height, ethnicity, ...
- Predict *life expectancy increase* from:
  - Medication, disease state, ...
- Predict *salary* from:
  - GPA, age, skill-set, ...
- …

Prediction of (Continuous) Output

- Predict a continuous output
  based on set of discrete / continuous inputs:
  - Eg, predict Salary from GPA
  - **Regression**



Training a Regressor

Age	GPA	TroMk	…	Eye	Salary
35	95	Y	…	Pale	100
22	110	N	…	Clear	58
:	:			:	:
10	87	N	…	Pale	53

Learner

Age	GPA	TroMk	…	Eye
32	90	N	…	Pale

Regressor

Salary
74

The **Linear** Regression Task

- **Given set of labeled Instances:** { [$\mathbf{x}_j$, y_j] }

  GPA, Age, TroubleMaker, ShoeSize → Salary

Eg:	[(97,	34,	1,	8);	150]
	[(93,	24,	0,	12);	200]
	[(88,	20,	0,	9);	45]

- Intuition: Evidence Adds, or Subtracts
  - Base salary := θ_0
  - Salary += $\theta_1 \times$ GPA
  - Salary += $\theta_2 \times$ TroubleMaker
  - Salary += $0 \times$ ShoeSize

Just allow $\theta_2 < 0$

Sometimes… $\theta_3 = 0$

The Linear Regression Task

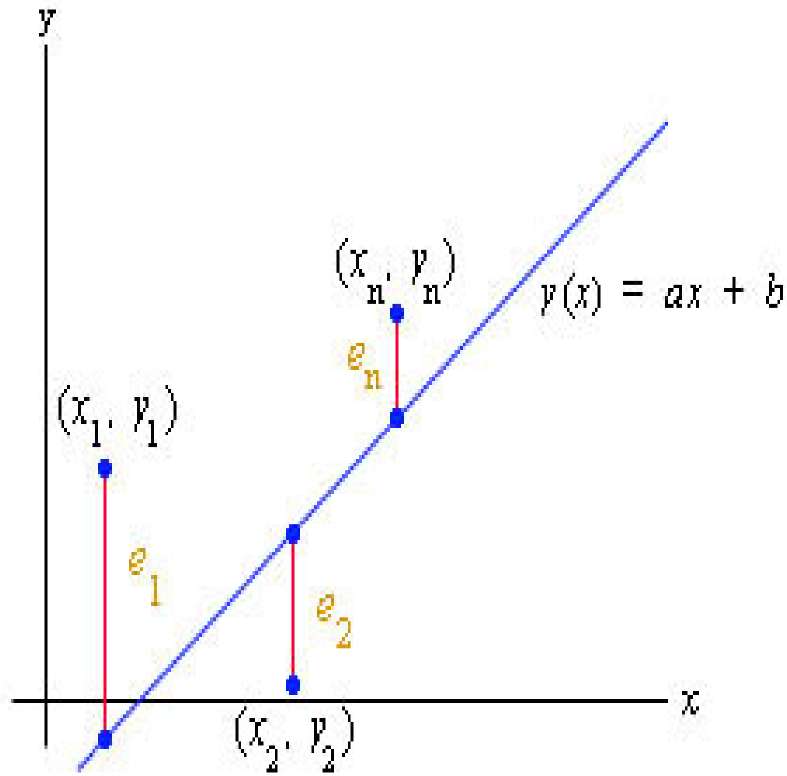
- **Given set of labeled Instances:** $\{ [\mathbf{x}_j, y_j] \}$

  GPA, Age, TroubleMaker, ShoeSize $\rightarrow$ Salary

  Eg: [(97, 34, 1, 8); 150]
  [(93, 24, 0, 12); 200]
  [(88, 20, 0, 9); 45]

- **Learn:** Mapping from $\mathbf{x}$ to $y(\mathbf{x})$
  - Direct linear mapping: $y(\mathbf{x}) \approx \theta_0 + \sum_j \theta_j x_j$
  - Find coeff's $\theta = (\theta_0, \theta_1, \ldots, \theta_k)$
- **Model**: Observed value $y(x) = \theta_0 + \sum_j \theta_j x_j + \varepsilon$ where $\varepsilon \sim N(0, \sigma^2)$

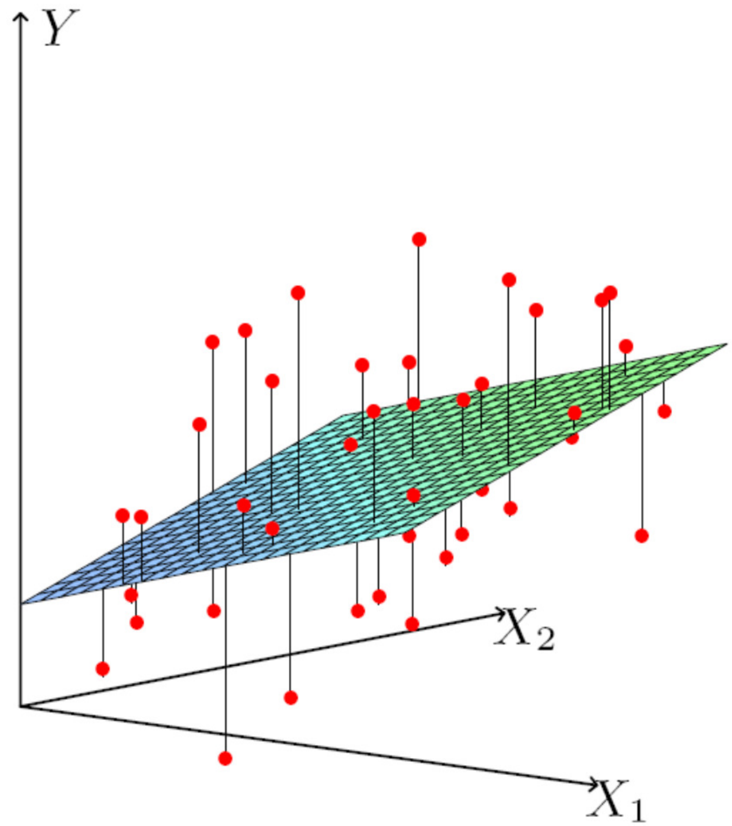
Best LINEAR Fit



- *Finding LINEAR fit*
  - *Find* $(\theta_0, \theta_1, \ldots, \theta_k)$
    $y = \theta_0 + \theta_1 x_1 + \ldots + \theta_k x_k$
- *Linear least squares fitting on* $X \in \Re$
- *Seek the linear function of* X *that minimizes the sum of squared residuals from* Y

$$\arg\min_{\theta} \left[\sum_i (y^{(i)} - \theta_0 - \sum_j \theta_j x_j^{(i)})^2 \right]$$

Best LINEAR Fit

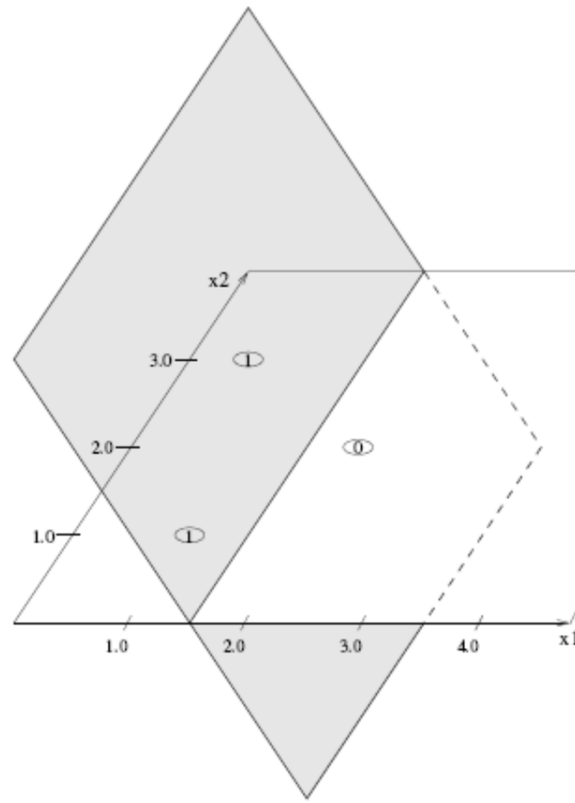


- *Finding LINEAR fit*
  - *Find* $(\theta_0, \theta_1, \ldots, \theta_k)$
    $y = \theta_0 + \theta_1 x_1 + \ldots + \theta_k x_k$
- *Linear least squares fitting on* $X \in \Re^2$
- *Seek the linear function of* X *that minimizes the sum of squared residuals from* Y

$$\arg\min_{\theta}\left[\sum_i \left(y^{(i)} - \theta_0 - \sum\nolimits_j \theta_j x_j^{(i)}\right)^2\right]$$

Linear Equation is Hyperplane

- Equation $\sum_i \theta_i x_i$ is a (hyper)plane



Linear Regression Model

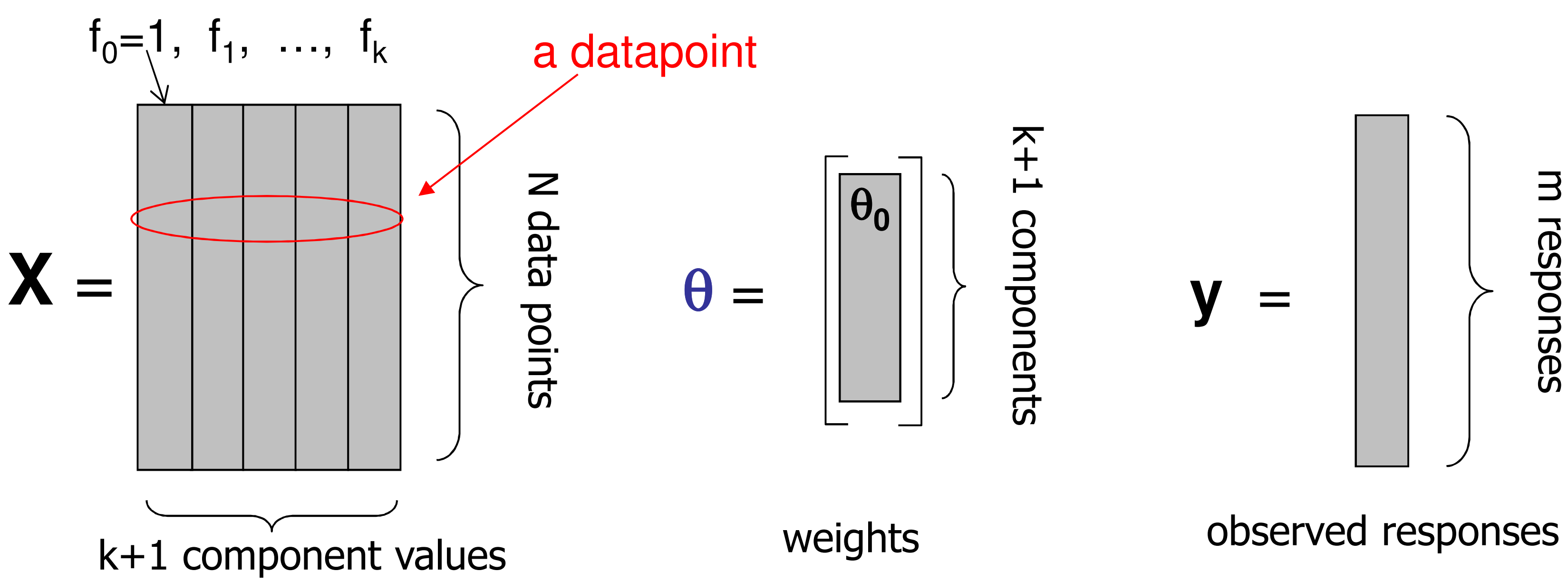
- Assumes that the regression function $f(\boldsymbol{x})$ is linear

$$f(\boldsymbol{x}) = \theta_0 + \sum_{i=1}^{k} \theta_i \, x_i$$

- *Linear models* are old tools but …
  - Still very useful
  - Simple
  - Allow an easy interpretation of regressor's effects
  - Useful to understand as foundation for many other methods
  - Very general as X_i's used can be any function of direct variables (quantitative or qualitative)
    - Basis functions

Dealing with Offset

- *Actually want k+1 values* $[\theta_0, \theta_1, \ldots, \theta_k]^T$
  $y = \theta_0 + \theta_1 x_1 + \ldots + \theta_k x_k$
- So view each k-tuple $\mathbf{x}^{(i)}$ as k**+1** tuple $[1, \mathbf{x}^{(i)}]$



Matrix Notation

- **X** is m × (k+1) of input vectors

$$X \equiv \begin{bmatrix} 1 & x_1^T \\ 1 & x_2^T \\ & \dots \\ 1 & x_m^T \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1k} \\ 1 & x_{21} & x_{22} & \dots & x_{2k} \\ & & \dots & & \\ 1 & x_{m1} & x_{m2} & \dots & x_{mk} \end{bmatrix}$$

- **y** is the m-vector of outputs (labels)

$$\mathbf{y} \equiv \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{bmatrix}$$

- θ is the (k+1)-vector of parameters

$$\boldsymbol{\theta} \equiv \begin{bmatrix} \theta_0 \\ \theta_1 \\ \dots \\ \theta_k \end{bmatrix}$$

How to make predictions …

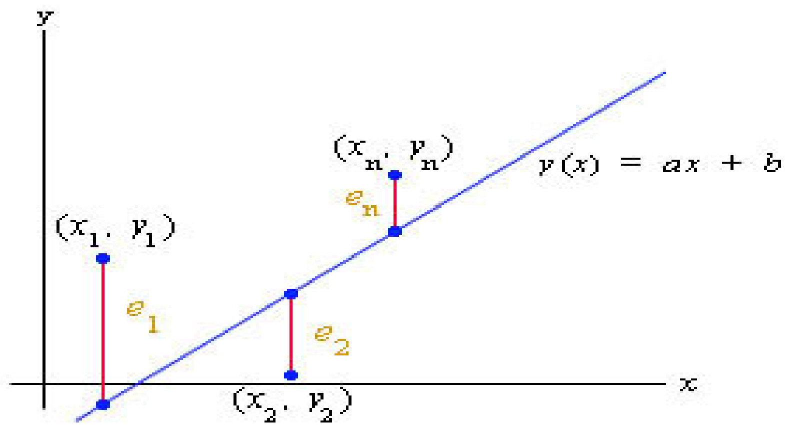
- The linear model is characterized by k+1 parameters $\boldsymbol{\theta}^*$

- For each instance $\boldsymbol{x}$, the prediction is

$$y(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{\theta}^*$$

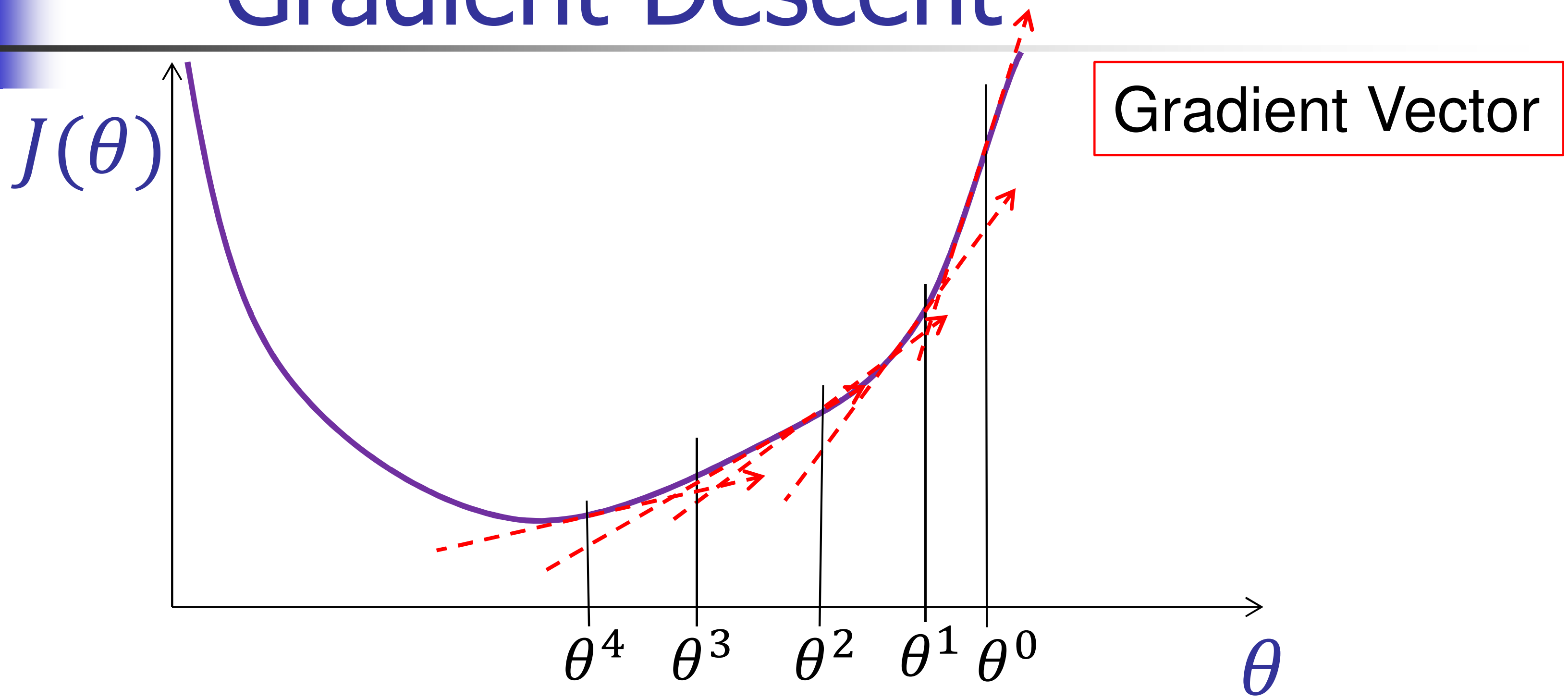
Gradient Descent

- Goal: Find θ^* that minimize squared error
  - $J(\theta) = \frac{1}{m}\sum_i [\, y^{(i)} - \theta_0 - \theta^T \mathbf{x}^{(i)}]^2$
  - $\ldots = \frac{1}{m}\sum_i e_i^2$



- Can use Gradient Descent!
  - aka Delta Rule, Adaline Rule, Widrow-Hoff Rule, LMS Rule, Classical Conditioning

Local Search via Gradient Descent



- Start w/ (random) weight vector θ^0
- Repeat until Converged (or bored):
  - Compute Gradient $\nabla J(\theta^t) = \left[\frac{\partial J(\theta^t)}{\partial \theta_0}, \dots, \frac{\partial J(\theta^t)}{\partial \theta_n}\right]$
  - Let $\theta^{t+1} = \theta^t + \eta \nabla J(\theta^t)$
- When CONVERGED: Return(θ^t)

Computing the Gradient

$$J(\theta) = \frac{1}{m} \sum_i [\, y^{(i)} - \theta^T x^{(i)} \,]^2$$

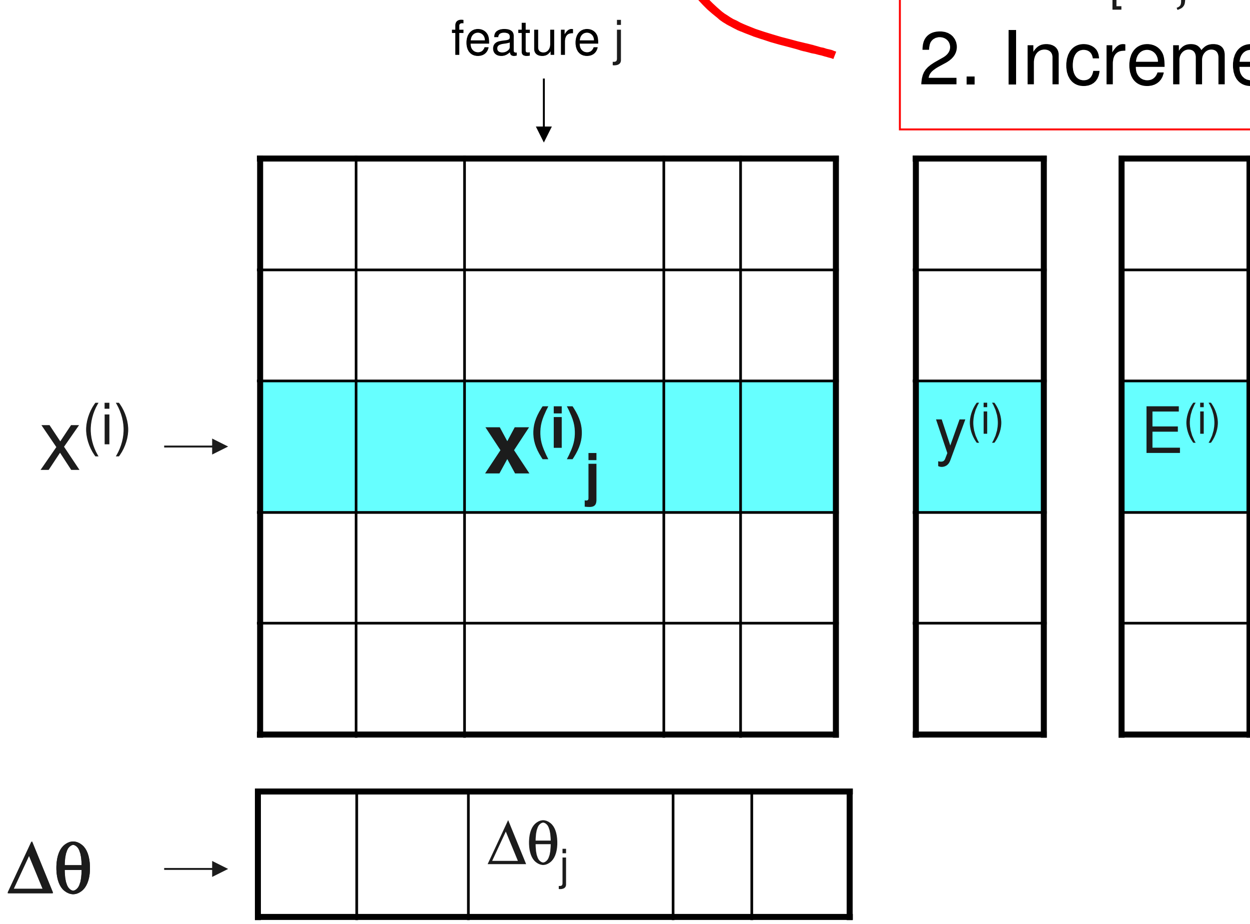
$$\Delta\boldsymbol{\Theta}_j = \frac{\partial J(\boldsymbol{\Theta})}{\partial \theta_j} = \frac{\partial}{\partial \theta_j}\left(\frac{1}{m}\sum_{i=1}^{m} err_i(\boldsymbol{\Theta})\right) = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial \theta_j} err_i(\boldsymbol{\Theta})$$

$$\begin{aligned}
\frac{\partial\, err_i(\theta)}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j}\left[\left(\sum_{i=1}^{m}\theta_j x_j^{(i)}\right) - y^{(i)}\right]^2 \\
&= 2\cdot\left[\left(\sum_{i=1}^{m}\theta_j x_j^{(i)}\right) - y^{(i)}\right]\cdot\frac{\partial}{\partial \theta_j}\left[\left(\sum_{i=1}^{m}\theta_j x_j^{(i)}\right) - y^{(i)}\right] \\
&= 2\cdot\left[\left(\sum_{i=1}^{m}\theta_j x_j^{(i)}\right) - y^{(i)}\right]\cdot x_j^{(i)}
\end{aligned}$$

Then descend a distance η along gradient $\left[\frac{\partial J(\theta)}{\partial \theta_0}, \frac{\partial J(\theta)}{\partial \theta_1}, \ldots, \frac{\partial J(\theta)}{\partial \theta_k}\right]$

LMS Alg

0. New θ
   $\Delta\theta := 0$
1. For each row i, compute
   a. $E^{(i)} = y^{(i)} - \theta^T \mathbf{x}^{(i)}$
   b. $\Delta\theta \mathrel{+}= E^{(i)} \mathbf{x}^{(i)}$
      $[\ \forall j \quad \Delta\theta_j \mathrel{+}= E^{(i)} x^{(i)}_j\ ]$
2. Increment $\theta \mathrel{+}= \eta_t \Delta\theta$

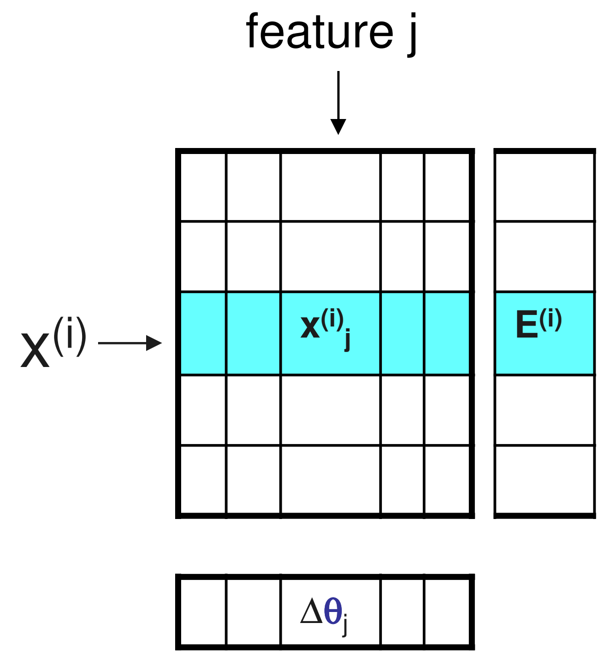


Gradient Descent Algorithm

Gradient-Descent(S : training examples; $\eta \in \Re^+$)

% S = { [$\mathbf{x}^{(i)}$, $y^{(i)}$] },…
% x = vector of input values; t is target output value
% η is learning rate (eg, 0.05)

- Initialize each θ_j to small random value
  - Typically $\in$ [–0.05, +0.05]
- Until termination condition is met, do
  - Initialize each $\Delta\theta_j \leftarrow 0$
  - For each [$\mathbf{x}^{(i)}$, $y^{(i)}$] $\in$ S, do
    - Set $\boxed{E^{(i)} \leftarrow y^{(i)} - \theta^{\mathbf{T}} \mathbf{x}^{(i)}}$
    - For each j, do
      - $\Delta\theta_j \leftarrow \Delta\theta_j + E^{(i)} x^{(i)}_j$
  - For each j do
    - $\theta_j \leftarrow \theta_j + \eta_t \Delta\theta_j$
- Return θ



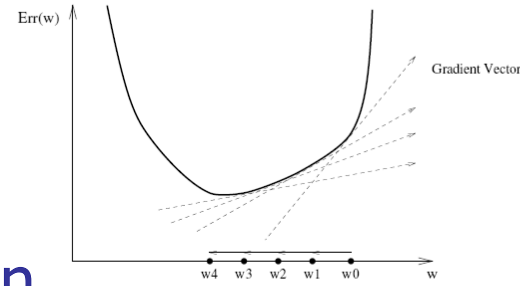
$$E^{(i)} \leftarrow y^{(i)} - \theta^{\mathbf{T}} x^{(i)}$$

Batch vs On-Line

- Batch:
  - do entire "epoch" (all instances),
  - then update weights
- On-line:
  - Update weights after each instances
  - … do multiple epochs…

  aka "Stochastic Gradient Descent",
  "Robbins-Munro algorithm"
- In gen'l…
  - Batch is smoother, better model of training data
  - But on-line may avoid local minima as "noisier"

Correctness



- Rule is intuitive: Climbs in correct direction...
- Theorem: Converges to correct answer, if ...
  - sufficiently small η
- Proof: Weight space has **EXACTLY 1 minimum**!
  (no non-global minima)

  $\Rightarrow$ with enough steps, finds correct function!
- Explains early popularity
- If η too large, may overshoot
  If η too small, takes too long
- Can use η_t … which decays with # of iterations, t

Learning Rates and Convergence

- Learning rate $\eta \equiv$ "step size"
- Convergence whenever…
  - $\lim_{t\to\infty} \eta_t = 0$
  - $\sum_t \eta_t = \infty$
  - $\sum_t \eta_t^2 < \infty$
- $\exists$ sophisticated alg's

  (Newton's method; Line Search; …)

  that choose step size automatically, converge faster.
- $\exists$ only one "basin" for linear threshold units

  $\Rightarrow$ local minimum is global minimum!
- Good starting point $\Rightarrow$ algorithm converges faster

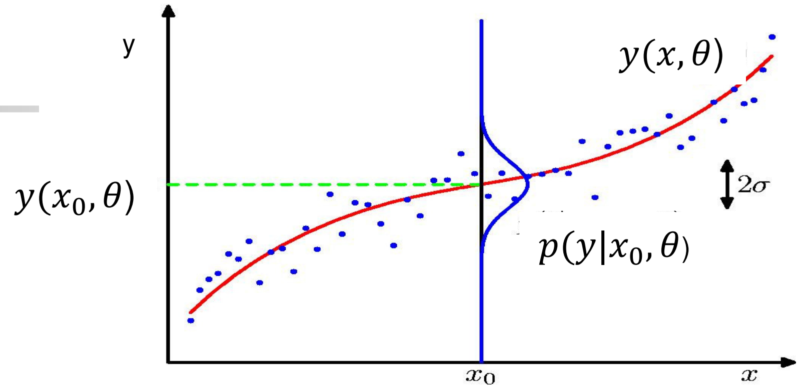
Results wrt Gradient Descent

Gradient descent (Delta training rule)

- guaranteed to converge to hypothesis with minimum squared error (eventually!) if…
  - Sufficiently small learning rate η
- ... even when training data…
  - contains noise
  - not separable!

Outline

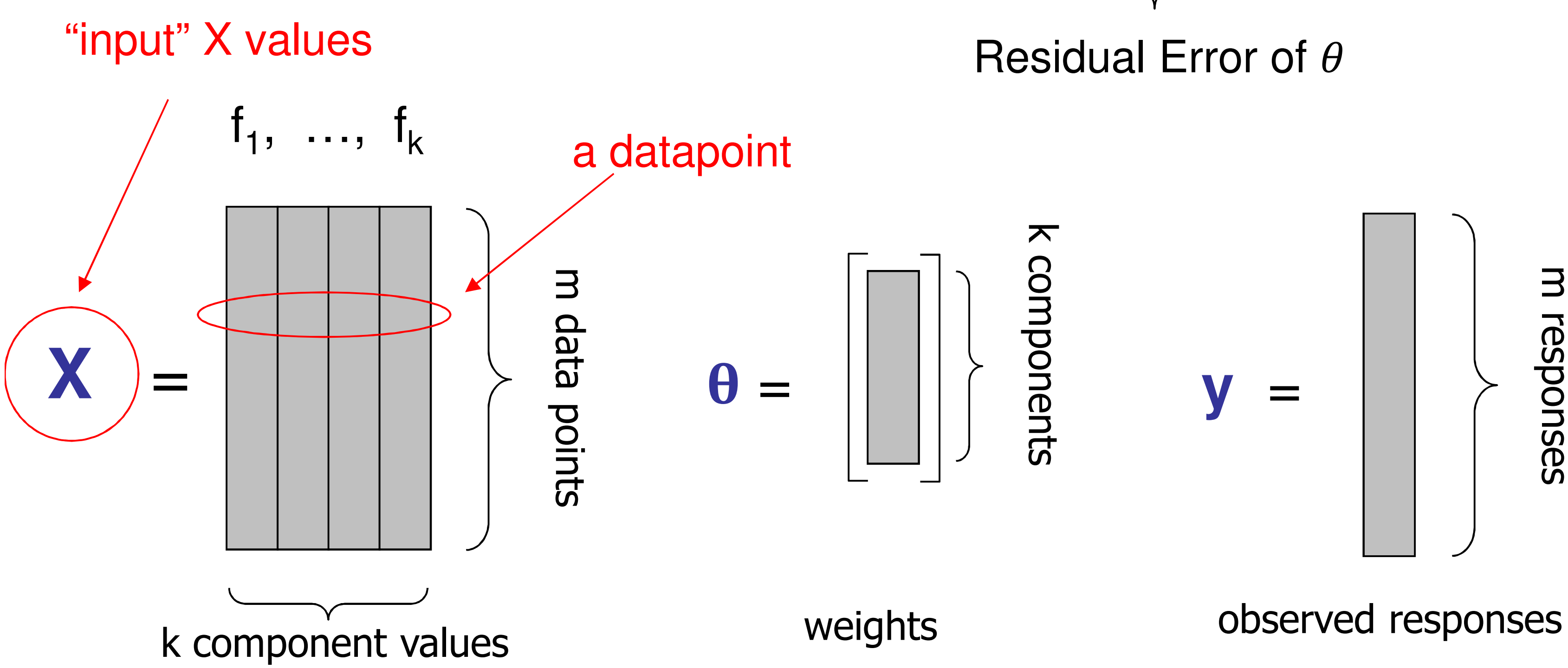
- Linear Regression
  - Iterative approach
  - Exact computation
    - Matrix operation
    - Least Square = MLE if Gaussian noise
  - Basis functions



Regression in Matrix Notation

$$\boldsymbol{\theta}^* = \underset{\theta}{argmin} \sum_{i=1}^{m} \left[y^{(i)} - \sum_{j=1}^{k} \theta_j \; x_j^{(i)} \right]^2$$

$$= \underset{\theta}{argmin} \; \underbrace{(\mathbf{y} - \mathbf{X}\,\boldsymbol{\theta})^{T}(\mathbf{y} - \mathbf{X}\,\boldsymbol{\theta})}_{\text{Residual Error of } \theta}$$



Optimal $\boldsymbol{\theta}^*$ Values

$$J(\theta) = (\mathbf{X}\theta - \mathbf{y})^T(\mathbf{X}\theta - \mathbf{y})$$

$$= (\mathbf{X}\theta)^T\mathbf{X}\theta - \mathbf{y}^T\mathbf{X}\theta - (\mathbf{X}\theta)^T\mathbf{y} + \mathbf{y}^T\mathbf{y}$$

$$= \theta^T\mathbf{X}^T\mathbf{X}\theta - 2\mathbf{y}^T\mathbf{X}\theta + \mathbf{y}^T\mathbf{y}$$

$$J'(\theta) = 2\theta\mathbf{X}^T\mathbf{X} - 2\mathbf{X}^T\mathbf{y}$$

$$J'(\theta) = 0$$

$$\Rightarrow \theta\mathbf{X}^T\mathbf{X} = \mathbf{X}^T\mathbf{y}$$

$$\Rightarrow \theta^* = (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{y})$$

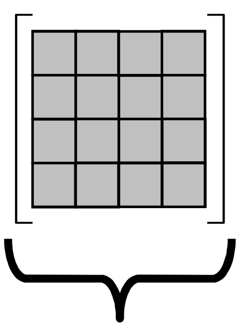
Regression solution = simple matrix operations

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

Setting derivative to 0 yields:

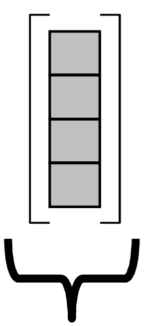
$$Solution: \ \boldsymbol{\theta}^* = \underbrace{(\mathbf{X}^T\mathbf{X})}_{\mathbf{A}^{-1}}{}^{-1} \underbrace{\mathbf{X}^T\mathbf{y}}_{\mathbf{b}} = \mathbf{A}^{-1}\mathbf{b}$$

where $\mathbf{A} = \mathbf{X}^T\mathbf{X} =$



k×k matrix over k features

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



k×1 vector

If X square & invertible, then $(X^T X)^{-1} X^T = X^{-1}$

Gradient-Descent vs Matrix-Inversion

Pro: Gradient Descent advantages

- ≈Biologically plausible
- Each iteration costs only $O(km)$
- If uses $< m$ iterations, faster than Matrix Inversion!
- More easily parallelizable

Con: Gradient Descent disadvantages

- It's moronic... essentially a slow way to build X^TX matrix, then solve a set of linear equations
- If m is small, it's especially outrageous

  If m is large then direct matrix inversion method can be problematic

  but not impossible if you want to be efficient
- Need to choose a good learning rate η_t -- how?
- Matrix inversion takes predictable time.

  You can't be sure when gradient descent will stop.

Likelihood of Data, given θ

- **Model**: Observed value is $y(\mathbf{x}) = \theta^T x_j + \varepsilon$ where $\varepsilon \sim N(0, \sigma^2)$

$$\boxed{P(y \mid \mathbf{x}, \boldsymbol{\theta}, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{[y - \boldsymbol{\theta}^T \mathbf{x}]^2}{2\sigma^2}}}$$

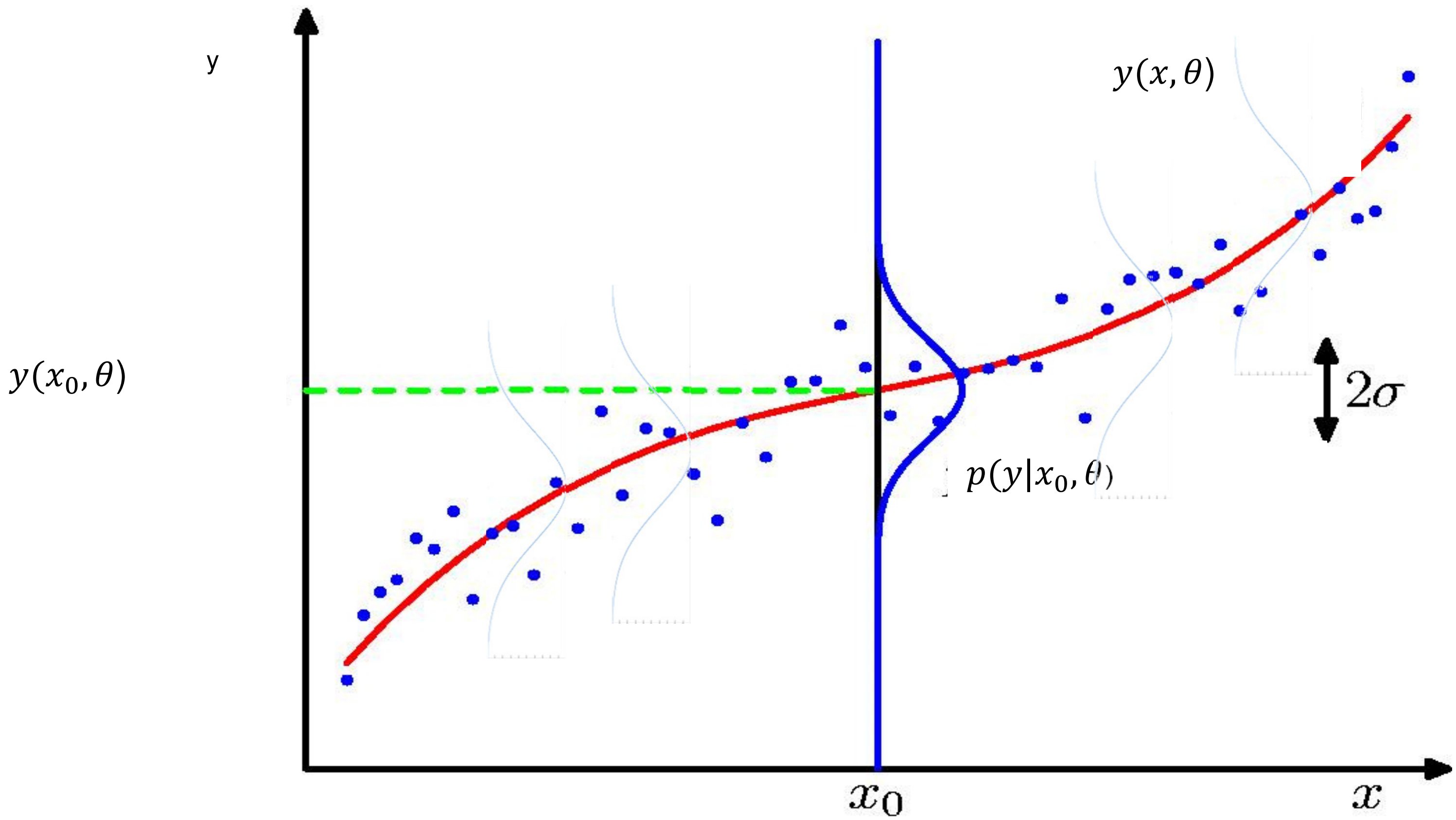
- Given $\boldsymbol{\theta}, \sigma$: y_1 given $\mathbf{x}_1$ is independent of y_2 given $\mathbf{x}_2$

$$P(y_1, y_2 \mid \mathbf{x_1}, \mathbf{x_2}, \boldsymbol{\theta}, \sigma) = P(y_1 \mid \mathbf{x_1}, \boldsymbol{\theta}, \sigma)\, P(y_2 \mid \mathbf{x_2}, \boldsymbol{\theta}, \sigma)$$

$$= \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{[y_1 - \boldsymbol{\theta}^T \mathbf{x_1}]^2}{2\sigma^2}} \times \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{[y_2 - \boldsymbol{\theta}^T \mathbf{x_2}]^2}{2\sigma^2}}$$

$$= \left(\frac{1}{\sigma\sqrt{2\pi}} \right)^2 \; e^{-\frac{[y_1 - \boldsymbol{\theta}^T \mathbf{x_1}]^2 + [y_2 - \boldsymbol{\theta}^T \mathbf{x_2}]^2}{2\sigma^2}}$$

Distribution p(y|x), at each x



Likelihood of Data, given θ

- Likelihood from MANY labeled instances, given θ

m {

Width	Heigh	Eyes	…	Light	size
35	95	Y	…	Pale	22
22	110	N	…	Clear	18
:	:			:	:
10	87	N	…	Pale	33

- $\ln P\left(\left[\mathrm{y}^{(1)}, \ldots, \mathrm{y}^{(m)} \right] \mid \left[\mathrm{x}^{(1)}, \ldots, \mathrm{x}^{(m)} \right], \theta, \sigma \right)$

$$= m \ln \left(\frac{1}{\sigma \sqrt{2\pi}} \right) - \frac{1}{2\sigma^2} \sum_i \left[\mathrm{y}^{(i)} - \Sigma_{\mathrm{j}} \theta_{\mathrm{j}} \mathrm{x}_{\mathrm{j}}^{(i)} \right]^2$$

Max Likely Estimate

- Find most likely value of θ from MANY labeled instances… (MLE)

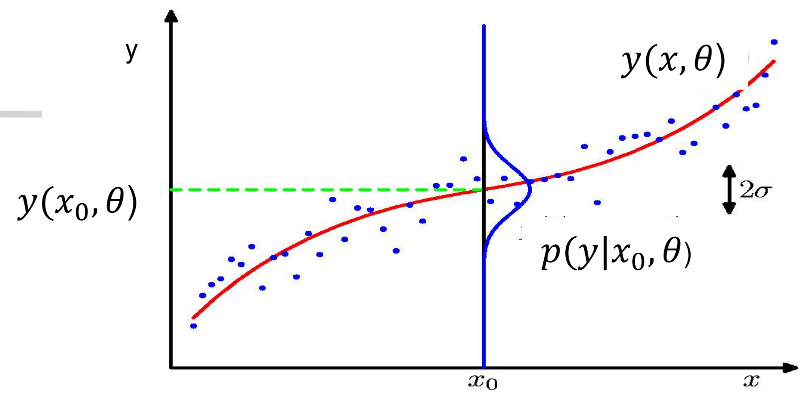
$$\ln P(\mathbf{y} \mid \mathbf{x}, \theta, \sigma) = m \ln\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{1}{2\sigma^2}\sum_i \left(y^{(i)} - \sum_j \theta_j x_j^{(i)}\right)^2$$

$$\operatorname*{argmax}_\theta \ln P(\mathbf{y} \mid \mathbf{x}, \theta, \sigma) = \boxed{\operatorname*{argmin}_\theta \sum_i \left(y^{(i)} - \sum_j \theta_j x_j^{(i)}\right)^2}$$

- **Least-squares Linear Regression is MLE for Gaussians !!!**

Outline

- Linear Regression
  - Iterative approach
  - Exact computation
  - Basis functions

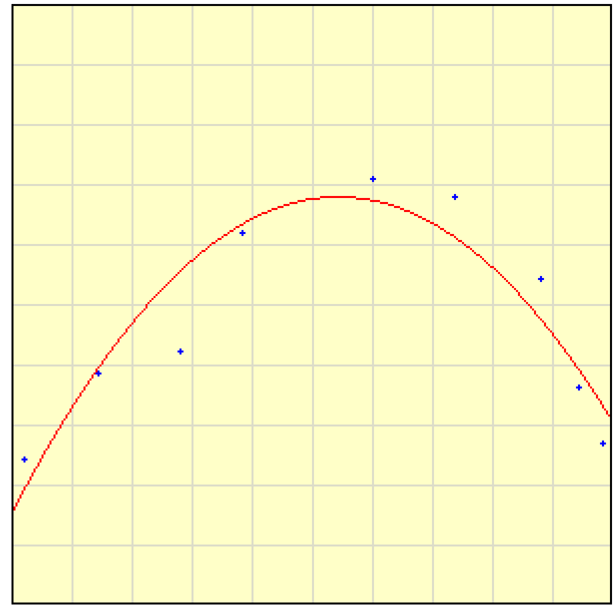


What about other features?

- Data:
  - Not linear!!
  - Perhaps

$$f(x) = \alpha_2 x^2 + \alpha_1 x + \alpha_0$$

- How to fit ???



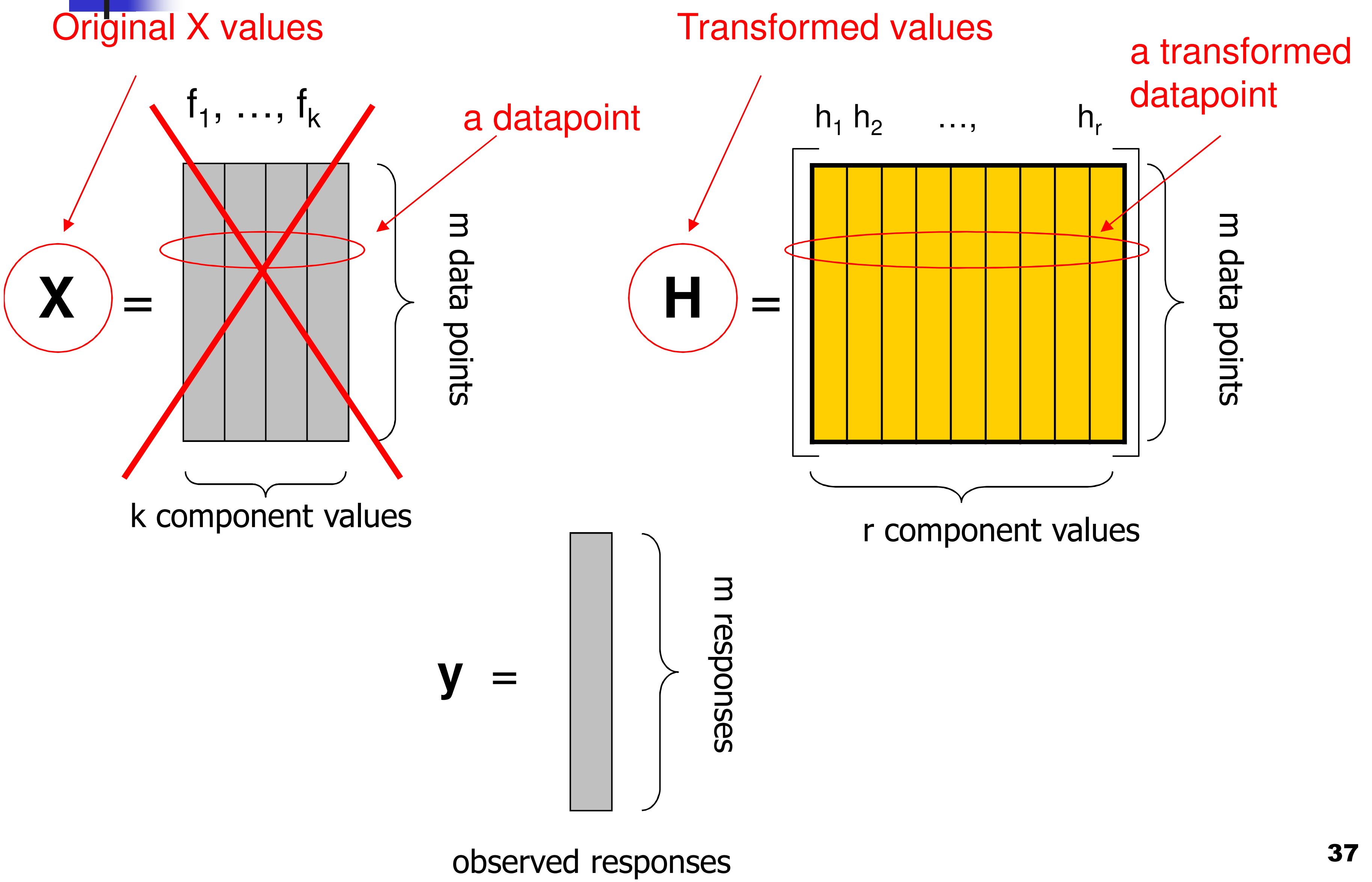
y
18
11
38
6

~~x~~
~~5~~
~~-2~~
~~7~~
~~3~~

x^2	x	1
25	5	1
4	-2	1
49	7	1
9	3	1

$\alpha_2 = -0.179$
$\alpha_1 = 1.938$
$\alpha_0 = 1.543$

General Approach



General Linear Regression Task

- **Given set of labeled Instances:** $\{ [\mathbf{x}_j, y_j] \}$
- **Learn:** Mapping from x to $y(\mathbf{x})$
- Can use **BASIS** functions: $H = \{ h_1(\mathbf{x}), \ldots, h_r(\mathbf{x}) \}$
  - Eg: x_i^2, x_i^3, $(x_1\ x_3)$, $x_i \sin(x_i)$, …
  - (Basis) linear mapping: $y(\mathbf{x}) \approx \sum_j \theta_j\, h_j(\mathbf{x})$
  - Find coeffs $\theta = (\theta_1, \ldots, \theta_r)$
- **Model**: Observed value $y^*(x) = \sum_j \theta_j\, h_j(\mathbf{x}) + \varepsilon$ where $\varepsilon \sim N(0, \sigma^2)$

Model is LINEAR in these bases… even if bases are NOT linear

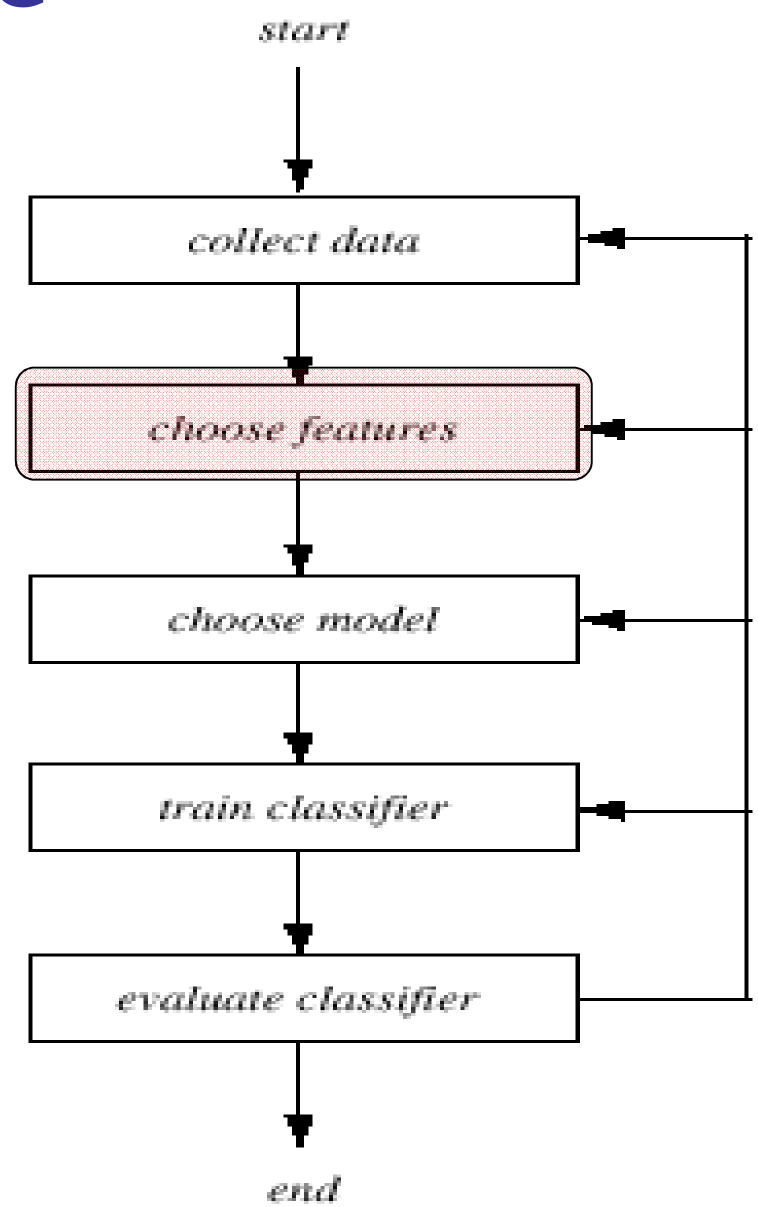
Features/Basis Functions

- Polynomials
  - $1, x, x^2, x^3, x^4, ..$
- Gaussian densities
  - Indicators
- Sigmoids
  - Step functions
- Sinusoids (Fourier basis)
- Wavelets
- Anything you can imagine...

Fun Demo
http://mste.illinois.edu/users/exner/java.f/leastsquares/

The Design Cycle

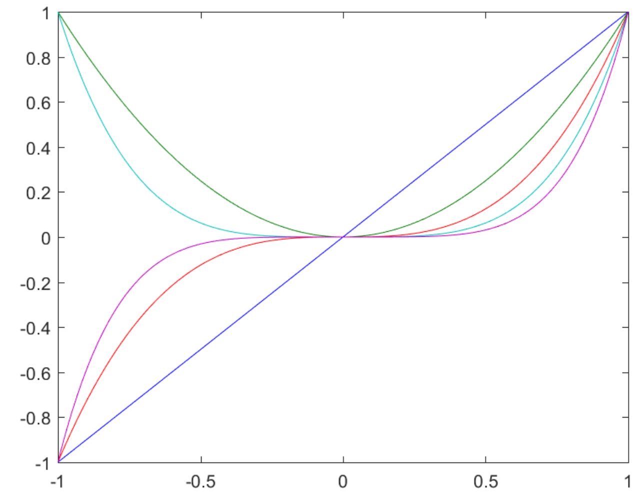
Features: basis functions…



Common Basis Functions

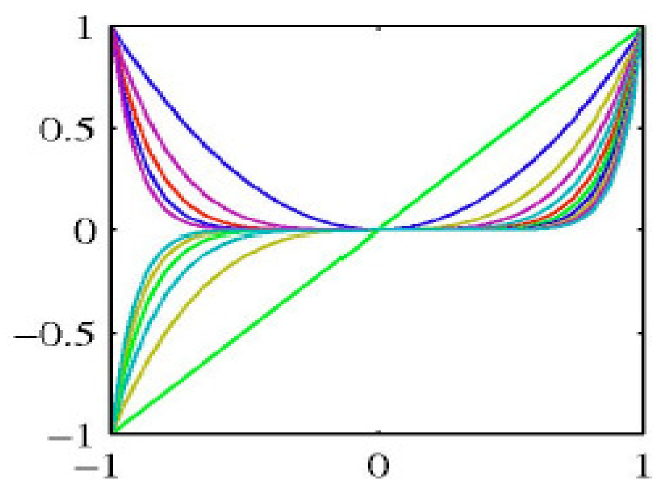
1. Polynomial basis functions:

$$\varphi_j(\boldsymbol{x}) = \boldsymbol{x}^j$$



- Global:

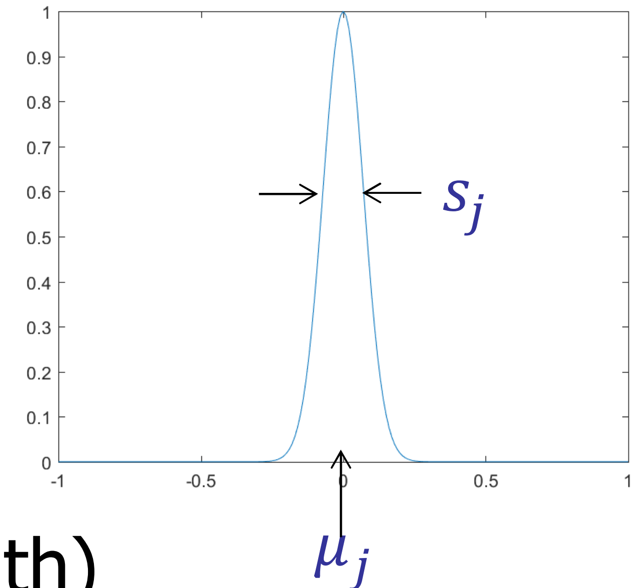
  small changes in $\boldsymbol{x}$ affect all basis functions



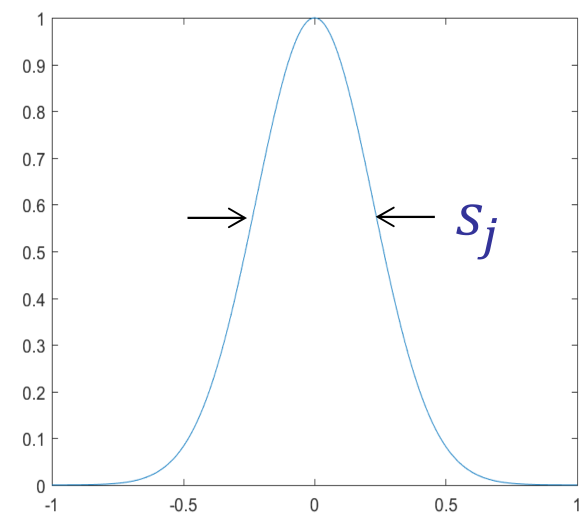
Common Basis Functions

2. Gaussian Basis functions:

$$\varphi_j(\boldsymbol{x}) = \exp\left(-\frac{(\boldsymbol{x}-\mu_j)^2}{2{s_j}^2}\right)$$



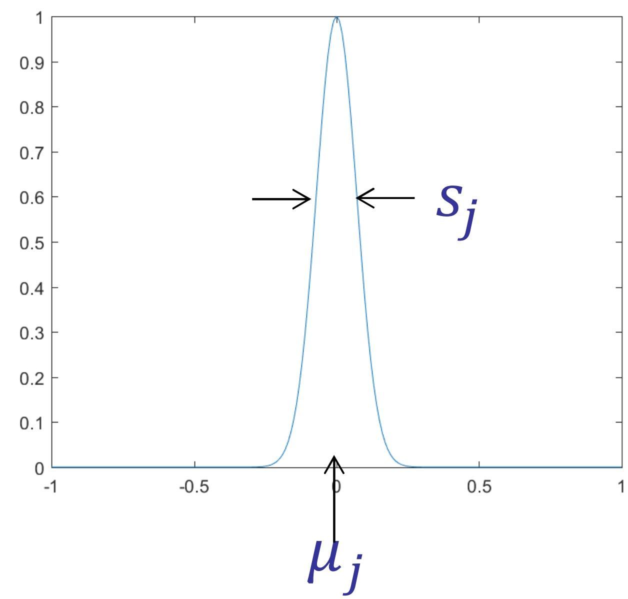
- μ_j , s_j control location, scale (width)
- $\varphi_j(\boldsymbol{x})$ is ≈ 1 when $\boldsymbol{x}$ is $\pm\alpha\, s_j$ of μ_j
  … else ≈ 0
- Wider range, for larger s_j



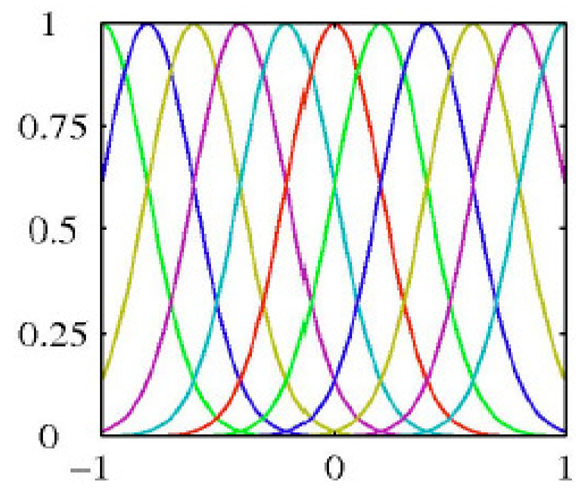
Common Basis Functions

2. Gaussian Basis functions:

$$\varphi_j(\boldsymbol{x}) = \exp\left(-\frac{(\boldsymbol{x}-\mu_j\,)^2}{2{s_j}^2}\right)$$

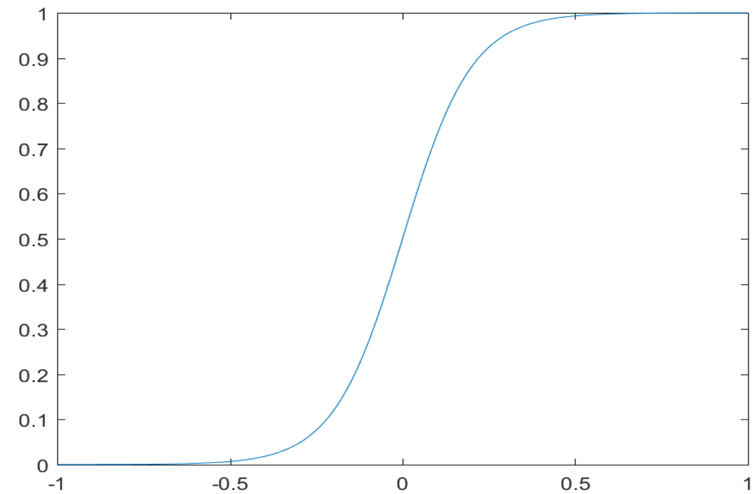


- Suite of basis functions …
  - Range of $\{\,\mu_j\,\}$, fixed $s_j = s$



- local:
  small changes in $\boldsymbol{x}$ only
  affect nearby basis functions

Common Basis Functions, …

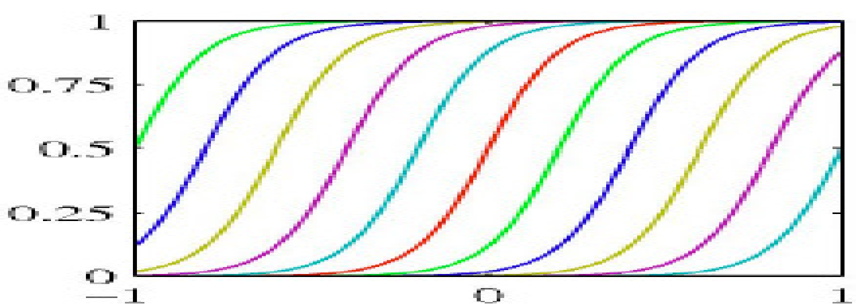


3. Sigmoidal basis functions:

$$\varphi_j(\boldsymbol{x}) = \sigma\left(k_j(x - \mu_j)\right)$$

where $\sigma(a) = \frac{1}{1+\exp(-a)}$

- μ_j and k_j control location and slope
- Suite



- local:
  small changes in $\boldsymbol{x}$ only affect nearby basis functions

Why use Basis Functions?

- Other basis functions
  can involve >1 variables
- Labels that are …
  - NOT linear combination of
    the original input space x,
  - might be linear in the feature space $\varphi_j(x)$
- … or at least, approximately so…

What if response $\mathbf{y}^{(i)}$ is a vector?

- Want to have linear model**s** for predicting both [height, weight] ?
- Nothing changes!
- Scalar prediction:

$$Solution: \ \boldsymbol{\theta}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- Vector prediction:

$$Solution: \ \underbrace{\boldsymbol{\theta}^*}_{\text{Weight MATRIX}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\underbrace{\mathbf{Y}}_{\text{Target MATRIX}}$$

  - Why?
  - … parameters are independent of each other…

Properties of Least Squares estimators

- If X fixed, Y_i are independent and $Var(Y_i) = \sigma^2$ constant, then

$$E(\hat{\boldsymbol{\theta}}) = \boldsymbol{\theta}, \quad Var(\hat{\boldsymbol{\theta}}) = \sigma^2 (X^T X)^{-1}$$

$$\text{and} \quad E(\hat{\sigma}^2) = \sigma^2 \quad \text{where} \quad \hat{\sigma}^2 = \frac{1}{m-k-1} \sum (y_i - \hat{f}(\mathbf{x}_i))^2$$

- If, in addition $Y_i = f(X_i) + \varepsilon$ with $\varepsilon \sim N(0, \sigma^2)$, then

$$\hat{\boldsymbol{\theta}} \sim N(\boldsymbol{\theta}, X^T X \sigma^2) \quad where \quad (m-k-1)\hat{\sigma}^2 \sim \sigma^2 \chi^2_{m-k-1}$$

Warning

$$\hat{\boldsymbol{\theta}} = (X^T X)^{-1} X^T \mathbf{y}$$

- If **X** square & invertible, then $(X^T X)^{-1} X^T = X^{-1}$
- When $\mathbf{X}^T\mathbf{X}$ is singular,
  the least squares coefficients $\boldsymbol{\theta}$ are not well defined
- Need alternative strategy to obtain a solution:
  - Recoding and/or dropping redundant columns
  - Filtering
  - Control fit by regularization

Sparse models: LATER!

Summary

- Finding linear regression model

  $$y = \theta^T x$$

  is very common
- Many ways to find this effective
  - Iterative approach: Gradient descent
  - Exact computation (Matrix operation)
    - Least Square = MLE if Gaussian noise
- Can handle complex relations by using basis functions