Linear Regression

Machine Learning 10-601

Seyoung Kim

Many of these slides are derived from Tom Mitchell. Thanks!

Regression

- So far, we've been interested in learning P(Y|X) where Y has discrete values (called 'classification')

- What if Y is continuous? (called 'regression')
  - predict weight from gender, height, age, …

  - predict Google stock price today from Google, Yahoo, MSFT prices yesterday

  - predict each pixel intensity in robot's current camera image, from previous image and previous action

Supervised Learning

- Wish to learn $f: X \rightarrow Y$, given observations for both X and Y in training data - Supervised learning
  - Classification: Y is discrete
  - Regression: Y is continuous

Regression

- Wish to learn $f{:}X \rightarrow Y$, where Y is real, given $\{<x^1,y^1>\ldots<x^N,y^N>\}$

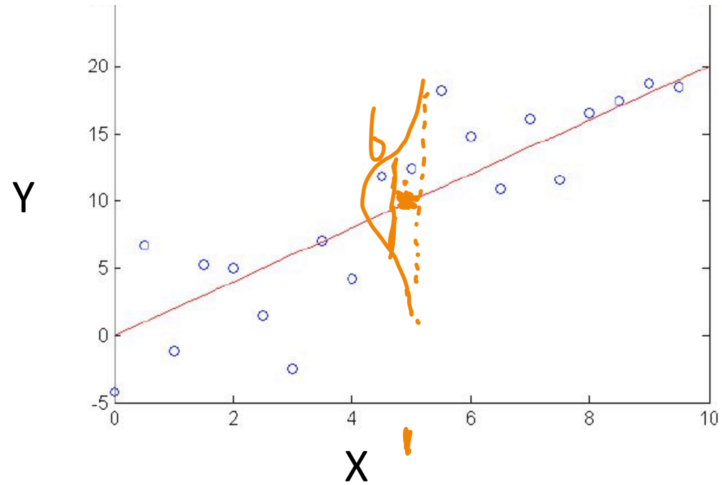
- Approach:

  1. choose some parameterized form for $P(Y|X; \theta)$

     (θ is the vector of parameters)

  2. derive learning algorithm as MLE or MAP estimate for θ

1. Choose parameterized form for P(Y|X; θ)



- Assume Y is some deterministic f(X), plus random noise ε

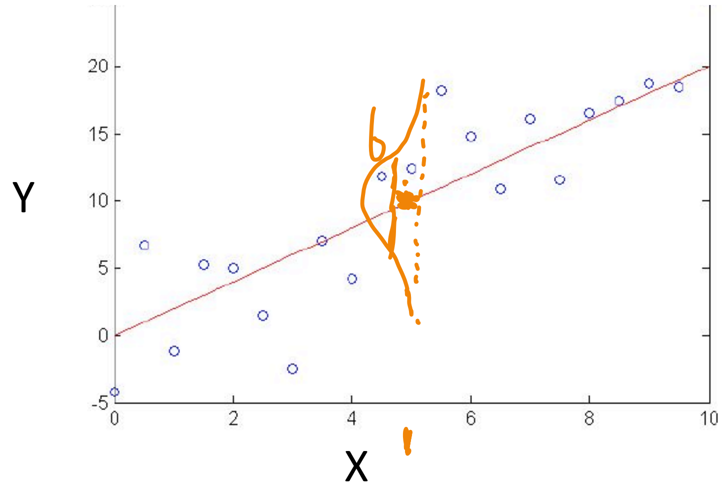
$$y = f(x) + \epsilon \qquad \text{where} \qquad \epsilon \sim N(0, \sigma)$$

- Therefore Y is a random variable that follows the distribution

$$p(y|x) = N(f(x), \sigma)$$

- The expected value of y for any given x is $E_{p(y|x)}[y] = f(x)$

1. Choose parameterized form for P(Y|X; θ)



- Assume Y is some deterministic f(X), plus random noise ε

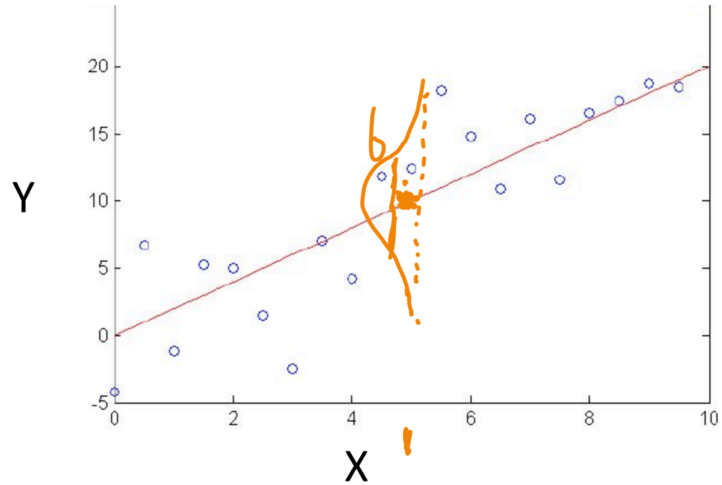
$$y = f(x) + \epsilon \qquad \text{where} \qquad \epsilon \sim N(0, \sigma)$$

- Assume a linear function for $f(x)$

$$f(x) = w_0 + \sum w_i x_i$$

$$p(y|x) = N(w_0 + \sum_i w_i x_i, \sigma)$$

1. Choose parameterized form for P(Y|X; θ)



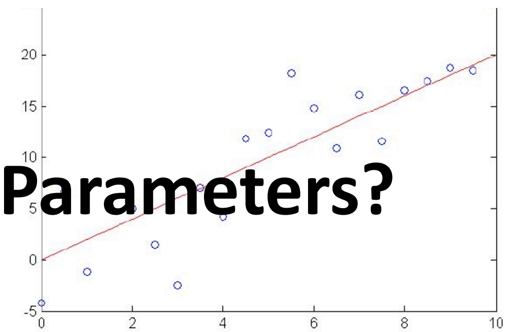
- Assume a linear function for $f(x)$

$$f(x) = w_0 + \sum w_i x_i$$

$$p(y|x) = N(w_0 + \sum_i w_i x_i, \sigma)$$

$$E_{p(x,y)}[y|x] = w_0 + \sum_i w_i x_i$$

2. How can We Learn Linear Regression Parameters?



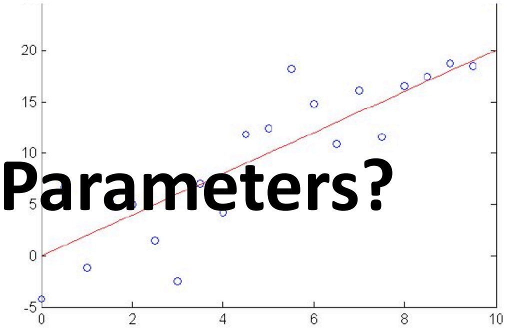
- Given the linear regression model

$$p(y|x) = N(w_0 + \sum_i w_i x_i, \sigma)$$

  – Notation: to make our parameters explicit, let's write using vector notation

$$W = \begin{pmatrix} \omega_0 \\ \omega_1 \\ \vdots \\ \omega_J \end{pmatrix}$$

- Given a training dataset of *N* samples $\{<x^1,y^1>\ldots<x^N,y^N>\}$
  – y^l: a univariate real value for the l-th sample
  – x^l: a vector of *J* features for the l-th sample
- How can we learn W from the training data?



2. How can We Learn Linear Regression Parameters?

$$p(y|x) = N(w_0 + \sum_i w_i x_i, \sigma)$$

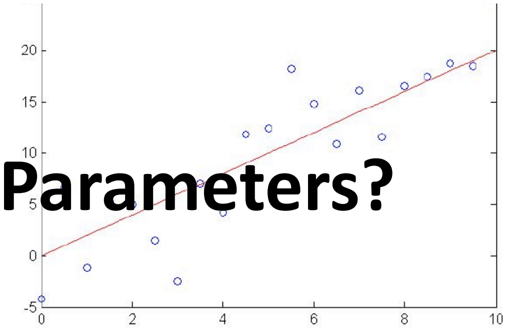
- How can we learn W from the training data (y^l, x^l), where l=1, …N for N samples? **Maximum Conditional Likelihood Estimate!**

$$W_{MCLE} = \arg\max_W \prod_l p(y^l | x^l, W)$$

$$W_{MCLE} = \arg\max_W \sum_l \ln p(y^l | x^l, W)$$

where

$$p(y|x;W) = \frac{1}{\sqrt{2\pi\sigma^2}} \; e^{-\frac{1}{2}(\frac{y-f(x;W)}{\sigma})^2}$$



2. How can We Learn Linear Regression Parameters?

- Learn Maximum Conditional Likelihood Estimate

$$W_{MCLE} = \arg\max_W \sum_l \ln p(y^l | x^l, W)$$

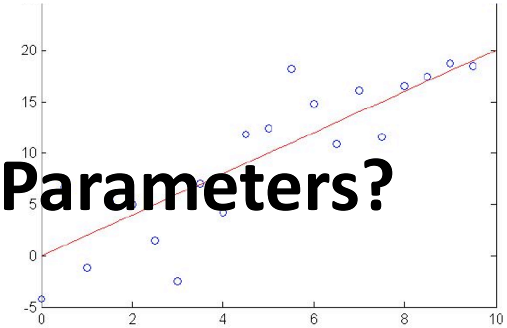
where

$$p(y|x;W) = \frac{1}{\sqrt{2\pi\sigma^2}} \; e^{-\frac{1}{2}(\frac{y-f(x;W)}{\sigma})^2}$$

- Thus, the conditional log-likelihood is given as

$$\sum_l \ln p(y^l|x^l;W) = \sum_l \left[\underbrace{\ln \frac{1}{\sqrt{2\pi\sigma^2}}}_{\text{Constant with respect to W}} - \frac{1}{2}\left(\frac{y^l - f(x^l;W)}{\sigma}\right)^2 \right]$$

2. How can We Learn Linear Regression Parameters?



- Learn Maximum Conditional Likelihood Estimate

$$W_{MCLE} = \arg\max_W \sum_l \ln p(y^l | x^l, W)$$

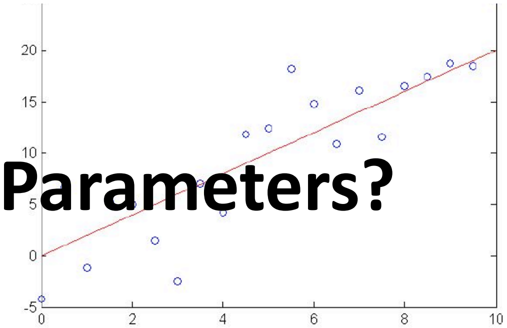
where

$$p(y|x;W) = \frac{1}{\sqrt{2\pi\sigma^2}} \; e^{-\frac{1}{2}(\frac{y-f(x;W)}{\sigma})^2}$$

- Thus, the conditional log-likelihood is given as

$$\sum_l \ln p(y^l | x^l; W) = \sum_l \left[\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2} \left(\frac{y^l - f(x^l; W)}{\sigma} \right)^2 \right]$$

$$\boxed{W_{MCLE} = \arg\max_W \sum_l -(y^l - f(x^l; W))^2}$$



2. How can We Learn Linear Regression Parameters?

- Learn Maximum Conditional Likelihood Estimate

$$W_{MCLE} = \arg\max_W \sum_l -(y^l - f(x^l; W))^2$$
$$= \arg\min_W \sum_l (y^l - f(x^l; W))^2$$

- Maximum conditional likelihood estimate is also called least squared-error estimate

- MLE provides a probabilistic interpretation of least squared-error estimate

Vector/Matrix Representation

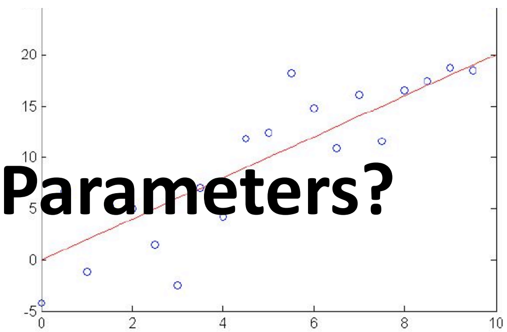
- Rewrite the linear regression model for training data using vector/matrix representation

$$\boldsymbol{y} = \boldsymbol{X}\,\boldsymbol{W} + \boldsymbol{\varepsilon}$$

Augmented input feature corresponding to w_0

$$\boldsymbol{y} = \begin{array}{cc} \begin{pmatrix} y^1 \\ \vdots \\ y^N \end{pmatrix} & \text{N samples} \end{array} \qquad \boldsymbol{X} = \begin{array}{cc} & \text{J input features} \\ & \begin{pmatrix} 1 & x_1^1 & \cdots & x_J^1 \\ \vdots & \vdots & & \vdots \\ 1 & x_1^N & \cdots & x_J^N \end{pmatrix} \end{array} \text{N samples} \qquad W = \begin{pmatrix} \omega_0 \\ \omega_1 \\ \vdots \\ \omega_J \end{pmatrix}$$

2. How can We Learn Linear Regression Parameters?



- Learn Maximum Conditional Likelihood Estimate

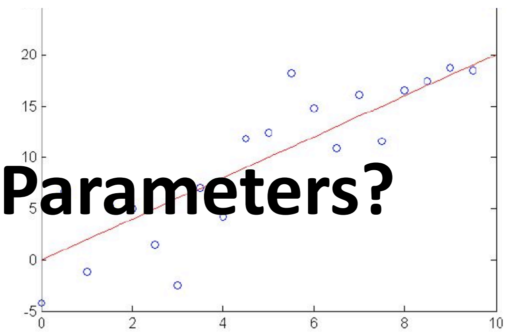
$$W_{MCLE} = \arg\max_{W} \sum_{l} -(y^l - f(x^l; W))^2$$

$$= \arg\min_{W} \sum_{l} (y^l - f(x^l; W))^2$$

$$= \arg\min\, (\boldsymbol{y} - \boldsymbol{X}\mathrm{W})^T (\boldsymbol{y} - \boldsymbol{X}\mathrm{W})$$

Re-write using vector representations of *N* samples in data

$$\boldsymbol{y} = \begin{array}{cc} \begin{pmatrix} y^1 \\ \vdots \\ y^N \end{pmatrix} & N \text{ samples} \end{array} \qquad \boldsymbol{X} = \begin{array}{c} J \text{ input features} \\ \begin{pmatrix} 1 & x_1^1 & \dots & x_J^1 \\ \vdots & \vdots & & \vdots \\ 1 & x_1^N & \dots & x_J^N \end{pmatrix} \end{array}$$



2. How can We Learn Linear Regression Parameters?

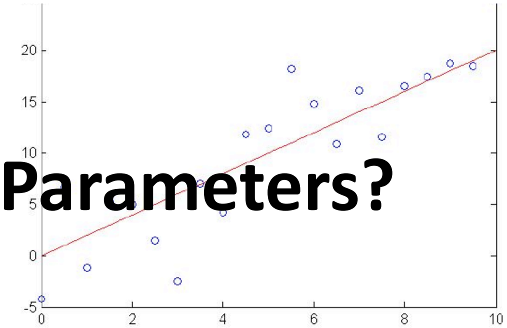
- Learn Maximum Conditional Likelihood Estimate

$$W_{MCLE} = \arg\min \, (\boldsymbol{y} - \boldsymbol{X}\mathrm{W})^T (\boldsymbol{y} - \boldsymbol{X}\mathrm{W})$$

Re-write using vector representations of *N* samples in data

$$\boldsymbol{y} = \begin{array}{cc} \begin{pmatrix} y^1 \\ \vdots \\ y^N \end{pmatrix} & N \text{ samples} \end{array} \qquad \boldsymbol{X} = \begin{array}{c} J \text{ input features} \\ \begin{pmatrix} 1 & x_1^1 & \dots & x_J^1 \\ \vdots & \vdots & & \vdots \\ 1 & x_1^N & \dots & x_J^N \end{pmatrix} \end{array}$$

$$\frac{\delta}{\delta W} (\boldsymbol{y} - \boldsymbol{X}\mathrm{W})^T (\boldsymbol{y} - \boldsymbol{X}\mathrm{W}) = 0$$



2. How can We Learn Linear Regression Parameters?

- Learn Maximum Conditional Likelihood Estimate

$$\frac{\delta}{\delta W}(\boldsymbol{y} - \boldsymbol{X}\mathrm{W})^T(\boldsymbol{y} - \boldsymbol{X}\mathrm{W})$$

$$= 2\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\mathrm{W}) = 0$$

$$\boxed{W_{MCLE} = (X^T X)^{-1} X^T y}$$

Comments on Training Linear Regression Models

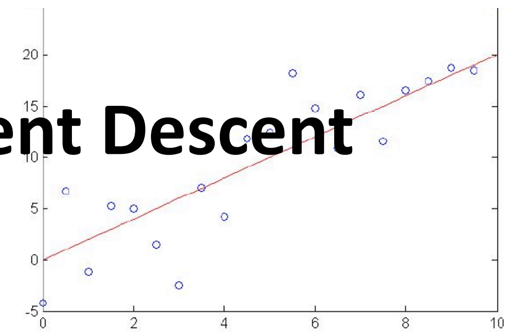
- Least squared error method

$$W_{MCLE} = (X^T X)^{-1} X^T y$$

  - A single equation for computing the estimate (i.e., a closed-form solution for MLE estimate)
  - When the dataset is extremely large, computing $X^T X$ and inverting it can be costly especially for streaming data

- Alternatively, gradient descent method
  - Works well on large datasets

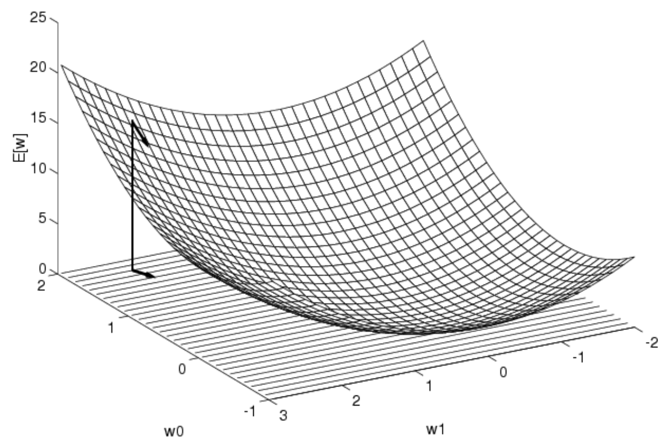
Training Linear Regression with Gradient Descent



Learn Maximum Conditional Likelihood Estimate

$$W_{MCLE} = \arg\min_W \sum_l (y - f(x;W))^2$$

Can we derive gradient descent rule for training?



Gradient

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

i.e.,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Gradient Descent:

Batch gradient: use error $E_D(\mathbf{w})$ over entire training set D

Do until satisfied:

1. Compute the gradient $\nabla E_D(\mathbf{w}) = \left[\frac{\partial E_D(\mathbf{w})}{\partial w_0} \ldots \frac{\partial E_D(\mathbf{w})}{\partial w_n}\right]$
2. Update the vector of parameters: $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla E_D(\mathbf{w})$

Stochastic gradient: use error $E_d(\mathbf{w})$ over single examples $d \in D$

Do until satisfied:

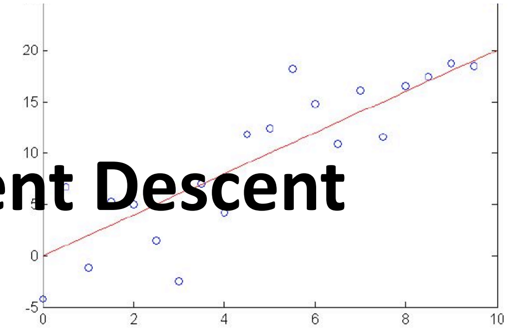
1. Choose (with replacement) a random training example $d \in D$
2. Compute the gradient just for d: $\nabla E_d(\mathbf{w}) = \left[\frac{\partial E_d(\mathbf{w})}{\partial w_0} \ldots \frac{\partial E_d(\mathbf{w})}{\partial w_n}\right]$
3. Update the vector of parameters: $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla E_d(\mathbf{w})$

Stochastic approximates Batch arbitrarily closely as $\eta \rightarrow 0$

Stochastic can be much faster when D is very large

Intermediate approach: use error over subsets of D

Training Linear Regression with Gradient Descent



- Learn Maximum Conditional Likelihood Estimate

$$W_{MCLE} = \arg\min_W \sum_l (y - f(x;W))^2$$

- Can we derive gradient descent rule for training?

$$\frac{\partial \sum_l (y - f(x;W))^2}{\partial w_i} = \sum_l 2(y - f(x;W)) \frac{\partial (y - f(x;W))}{\partial w_i}$$

$$= \sum_l -2(y - f(x;W)) \frac{\partial f(x;W)}{\partial w_i}$$

And if $f(x) = w_0 + \sum_i w_i x_i$ …

Gradient descent rule:

$$\boxed{w_i \leftarrow w_i + \eta \sum_l (y^l - f(x^l;W))\ x_i^l}$$

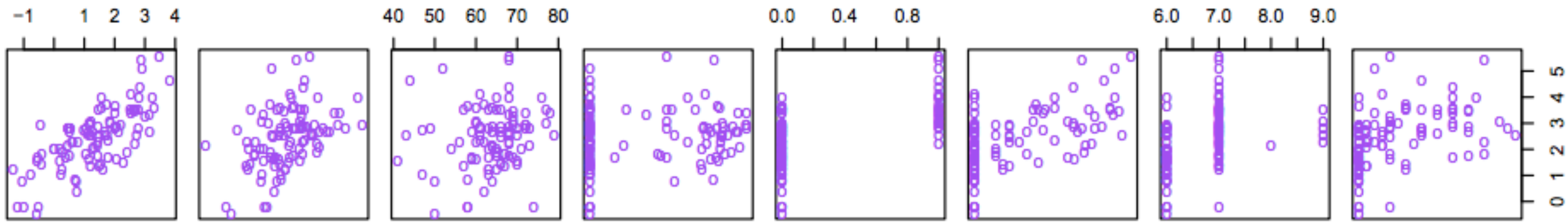
Example: Prostate Cancer

- Is there correlation between the level of prostate-specific antigen and a number of clinical measures in men who were about to receive a radical prostatectomy for 97 men
  - x : clnical measures
    - log cancer volume (lcavol)
    - log prostate weight (lweight)
    - age
    - log of the amount of benign prostatic hyperplasia (lbph)
    - seminal vesicle invasion (svi)
    - log of capsular penetration (lcp)
    - Gleason score (gleason)
    - percent of Gleason scores 4 or 5 (pgg45)
  - y: level of prostate-specific antigen

Hastie/Tibshirani/Friedman Elements of statistical learning

Example: Prostate Cancer

- Correlation between y and each input feature x_i

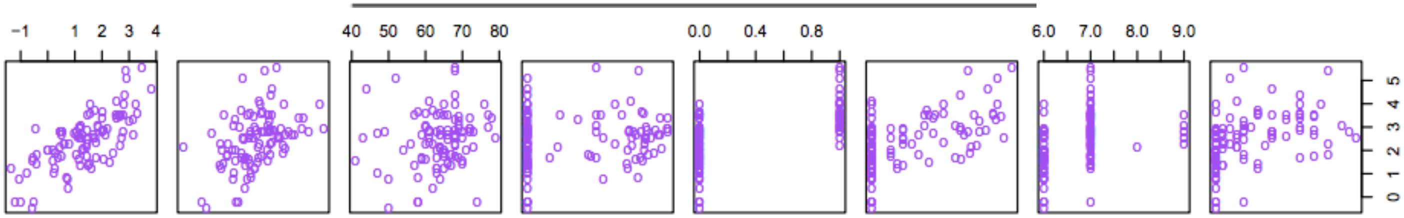


  – x : clnical measures
    - log cancer volume (lcavol)
    - log prostate weight (lweight)
    - age
    - log of the amount of benign prostatic hyperplasia (lbph)
    - seminal vesicle invasion (svi)
    - log of capsular penetration (lcp)
    - Gleason score (gleason)
    - percent of Gleason scores 4 or 5 (pgg45)

Example: Prostate Cancer

- Estimated regression coefficients

X Term	W Coefficient
Intercept	2.46
lcavol	0.68
lweight	0.26
age	−0.14
lbph	0.21
svi	0.31
lcp	−0.29
gleason	−0.02
pgg45	0.27



Comments on Least Squared Error Estimate

$$\boxed{W_{MCLE} = (X^T X)^{-1} X^T y}$$

- In many problems of practical interest, *N>J* (i.e., the number of data points *N* is larger than the dimensionality *J* of the input space and the matrix X is of full column rank.)
- When *N>J*, 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 X has less than full column rank? *N<J*
  - MAP estimate

How about MAP instead of MLE estimate?

- Let's assume Gaussian prior: each $w_i \sim N(0, \sigma_0)$ for i=1,…,J

$$p(w_i) = \frac{1}{Z}\exp\left(-\frac{(w_i - 0)^2}{2\sigma^2{}_0}\right)$$

  – Note we do not place a prior on w_0. Why?
  – We assume a model without an intercept (W=(w_1, …, w_J)) after mean-centering data y and X. Why? See Hastie/Tibshirani/Friedman Ex 3.5 page 95.

- MAP estimate is given as

$$\arg\max \ln P(W \mid X, y) = \arg\max \ln(X, y \mid W) + \ln P(W)$$

How about MAP instead of MLE estimate?

- Let's assume Gaussian prior: each $w_i \sim N(0, \sigma_0^2)$

$$p(w_i) = \frac{1}{Z} \exp\left(-\frac{(w_i - 0)^2}{2\sigma_0^2}\right)$$

- Then MAP estimate is given as

$$W = \arg\max_W -\frac{1}{2\sigma_0^2} \sum_{w_i \in W} w_i^2 + \sum_{l \in training\ \ data} \ln P(Y^l | X^l; W)$$

$$= \arg\min_W \frac{1}{2\sigma_0^2} \sum_{w_i \in W} w_i^2 + \sum_{l \in training\ \ data} (y^l - f(x^l; W))^2$$

$$= \arg\min (\boldsymbol{y} - \boldsymbol{X}\mathrm{W})^T (\boldsymbol{y} - \boldsymbol{X}\mathrm{W}) + (1/2\sigma_0^2)\ \mathrm{W}^{\mathrm{T}}\mathrm{W}$$

How about MAP instead of MLE estimate?

- Then MAP estimate is

$$\arg\min\ (\boldsymbol{y} - \boldsymbol{X}\mathrm{W})^T(\boldsymbol{y} - \boldsymbol{X}\mathrm{W}) + (1/2\sigma_0^2)\ \mathrm{W}^\mathrm{T}\mathrm{W}$$

$$\frac{\delta}{\delta W}(\boldsymbol{y} - \boldsymbol{X}\mathrm{W})^T(\boldsymbol{y} - \boldsymbol{X}\mathrm{W}) + (1/2\sigma_0^2)\ \mathrm{W}^\mathrm{T}\mathrm{W} = 0$$

Invertible, even if N<J

$$\boxed{W_{MAP} = (X^T X + \frac{1}{2\sigma_0^2} I)^{-1} X^T y}$$

Small σ_0^2 value means strong prior belief

MAP Estimate and Regularization

- MAP estimation

$$\arg\min\, (\boldsymbol{y} - \boldsymbol{X}\mathrm{W})^T(\boldsymbol{y} - \boldsymbol{X}\mathrm{W}) + (1/2\sigma_0^2)\ \mathrm{W}^{\mathrm{T}}\mathrm{W}$$

with prior $w_i \sim N(0, \sigma_0^2)$

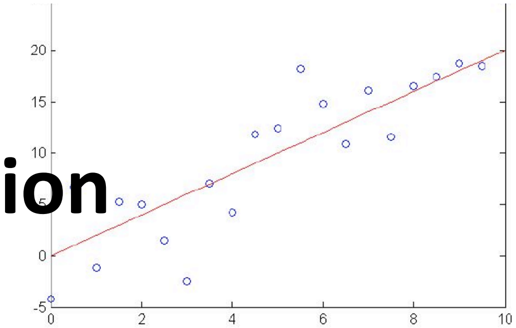
- More generally, this can be viewed as a regularization

$$\arg\min\, (\boldsymbol{y} - \boldsymbol{X}\mathrm{W})^T(\boldsymbol{y} - \boldsymbol{X}\mathrm{W}) + \lambda\ \mathrm{W}^{\mathrm{T}}\mathrm{W}$$

with regularization parameter λ

$$\boxed{\text{Equivalently } ||\boldsymbol{y} - \boldsymbol{X}\mathrm{W}||_2^2 + \lambda\ ||\mathrm{W}||_2^2}$$

Generalizing Linear Regression



$$p(y|x) = N(f(x), \sigma)$$

- E.g., assume f(x) is linear function of x

$$f(x) = w_0 + \sum_i w_i x_i$$

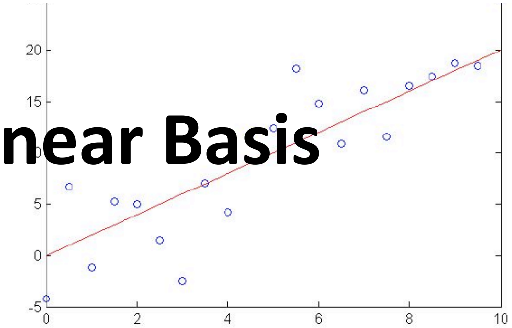
$$p(y|x) = N(w_0 + \sum_i w_i x_i, \sigma)$$

- f(x) is linear in x_i's and also linear in w_i's

Generalizing Linear Regression: Nonlinear Basis Function

- linear in w_i's
  - Widely-used assumption because of the mathematical convenience and easy estimation

- linear in x_i's
  - We can relax this by choosing arbitrary non-linear basis function $\phi(x_i)$
    - So far, we assumed $\phi_i(x) = x$
    - We can also use $\phi_i(x) = (1, x, x^2, x^3)$



Generalizing Linear Regression: Nonlinear Basis Function

$$p(y|x) = N(f(x), \sigma)$$

- E.g., assume f(x) is linear function of x

$$f(x) = \sum_i w_i \phi_i(x)$$

$$p(y|x) = N\left(\sum_i w_i \phi_i(x),\ \sigma\right)$$

Generalizing Linear Regression: Nonlinear Basis Function

- Different basis functions can be used
  - 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

Regression – What you should know

Under general assumption $p(y|x;W) = N(f(x;W), \sigma)$

1. MLE corresponds to minimizing Sum of Squared prediction Errors (SSE)
2. MAP estimate minimizes SSE plus sum of squared weights

3. Again, learning is an optimization problem once we choose our objective function
   - MLE: maximize data likelihood
   - MAP: maximize posterior probability, P(W | data)

4. Again, we can use gradient descent as a general learning algorithm
   - as long as our objective f is differentiable wrt W

5. Nothing we said here required that f(x) be linear in x -- just linear in W
6. Gradient descent is just one algorithm – linear algebra solutions too

Logistic Regression as Regression

$$P(Y=1|X=<X_1,...X_n>) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

implies

$$P(Y=0|X=<X_1,...X_n>) = \frac{exp(w_0+\sum_i w_i X_i)}{1+exp(w_0+\sum_i w_i X_i)}$$

implies

$$\frac{P(Y=0|X)}{P(Y=1|X)} = exp(w_0+\sum_i w_i X_i)$$

linear classification rule!

implies

$$\ln \frac{P(Y=0|X)}{P(Y=1|X)} = w_0+\sum_i w_i X_i$$