

Gaussian Naive Bayes and Linear Regression

Instructor: Alan Ritter

Many Slides from Tom Mitchell

What if we have continuous X_i ?

Eg., image classification: X_i is real-valued i^{th} pixel



What if we have continuous X_i ?

Eg., image classification: X_i is real-valued i^{th} pixel

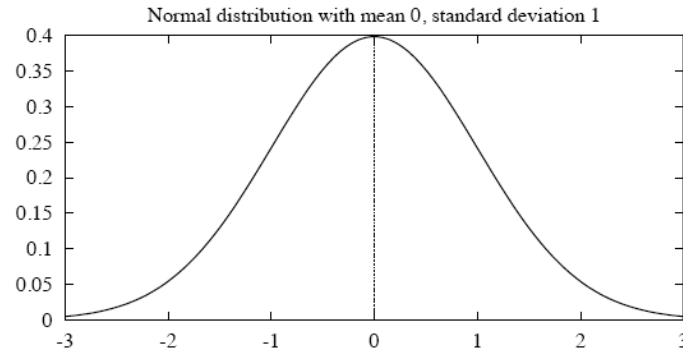
Naïve Bayes requires $P(X_i | Y=y_k)$, but X_i is real (continuous)

$$P(Y = y_k | X_1 \dots X_n) = \frac{P(Y = y_k) \prod_i P(X_i | Y = y_k)}{\sum_j P(Y = y_j) \prod_i P(X_i | Y = y_j)}$$

Common approach: assume $P(X_i | Y=y_k)$ follows a Normal (Gaussian) distribution

Gaussian Distribution (also called “Normal”)

$p(x)$ is a *probability density function*, whose integral (not sum) is 1



$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

The probability that X will fall into the interval (a, b) is given by

$$\int_a^b p(x) dx$$

- Expected, or mean value of X , $E[X]$, is

$$E[X] = \mu$$

- Variance of X is

$$Var(X) = \sigma^2$$

- Standard deviation of X , σ_X , is

$$\sigma_X = \sigma$$

Gaussian Naïve Bayes Algorithm – continuous X_i (but still discrete Y)

- Train Naïve Bayes (examples)

for each value y_k

estimate* $\pi_k \equiv P(Y = y_k)$

for each attribute X_i estimate $P(X_i|Y = y_k)$

- class conditional mean μ_{ik} , variance σ_{ik}

- Classify (X^{new})

$$Y^{new} \leftarrow \arg \max_{y_k} P(Y = y_k) \prod_i P(X_i^{new} | Y = y_k)$$

$$Y^{new} \leftarrow \arg \max_{y_k} \pi_k \prod_i \mathcal{N}(X_i^{new}; \mu_{ik}, \sigma_{ik})$$

* probabilities must sum to 1, so need estimate only n-1 parameters...

Estimating Parameters: Y discrete, X_i continuous

Maximum likelihood estimates:

$$\hat{\mu}_{ik} = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j X_i^j \delta(Y^j = y_k)$$

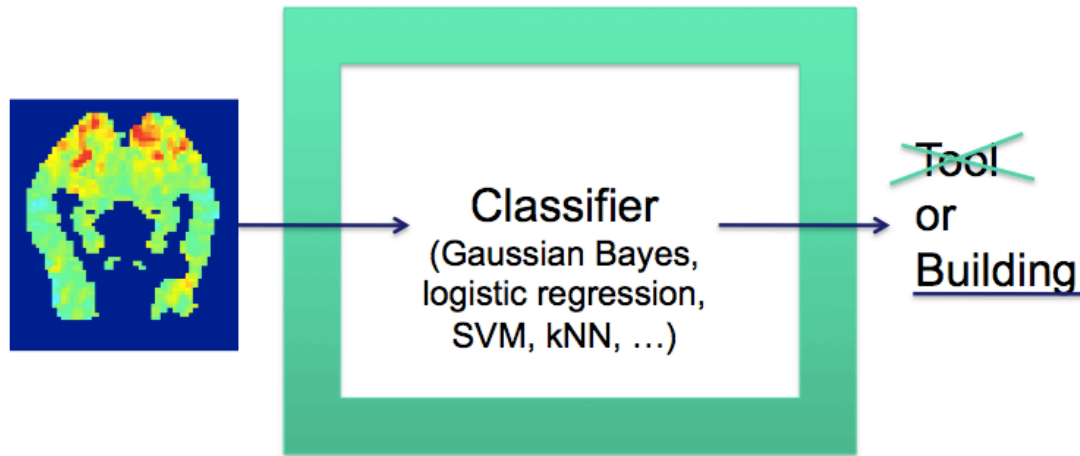
Diagram annotations:

- $\hat{\mu}_{ik}$: ith feature, kth class
- X_i^j : jth training example
- $\delta()$: $\delta()=1$ if $(Y^j=y_k)$ else 0

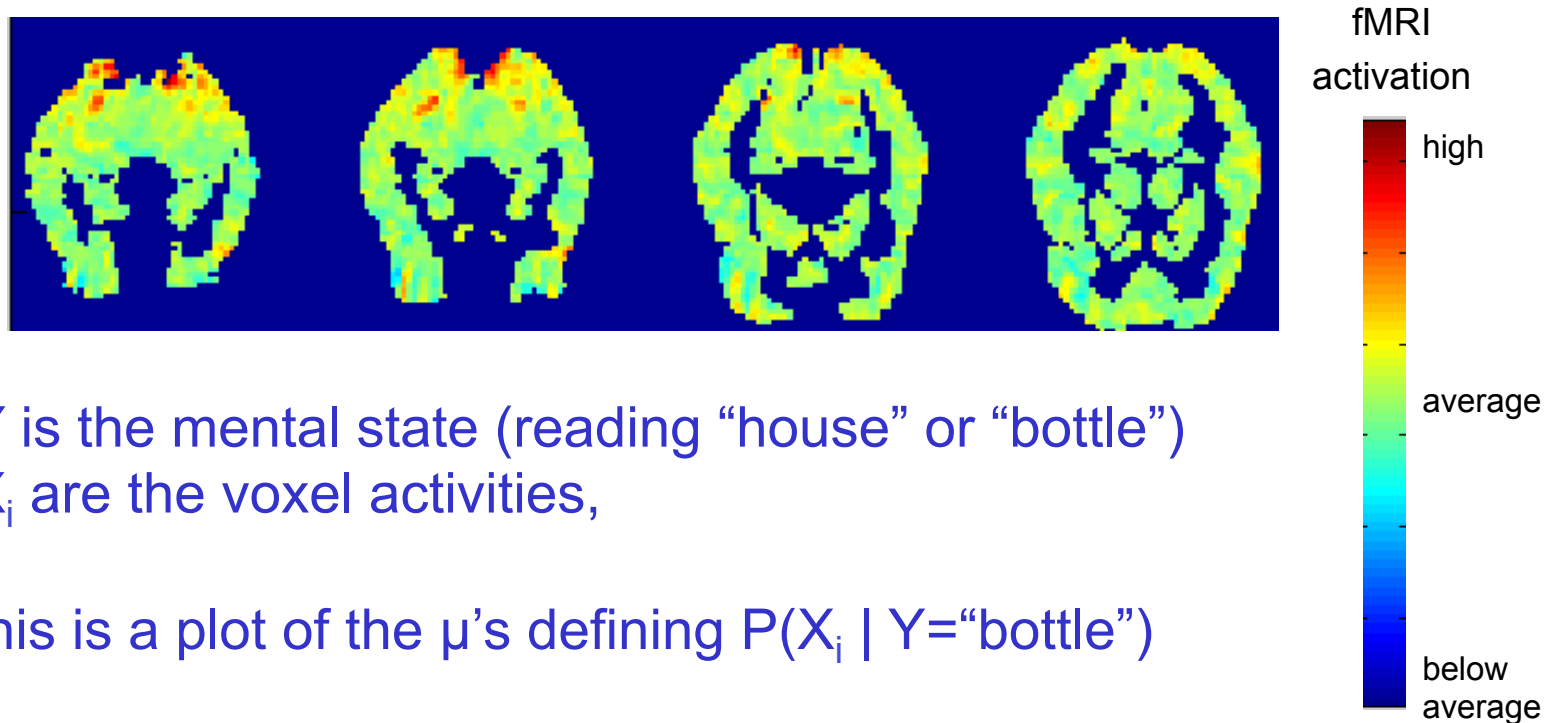
$$\hat{\sigma}_{ik}^2 = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j (X_i^j - \hat{\mu}_{ik})^2 \delta(Y^j = y_k)$$

GNB Example: Classify a person's cognitive state, based on brain image

- reading a sentence or viewing a picture?
- reading the word describing a “Tool” or “Building”?
- answering the question, or getting confused?



Mean activations over all training examples for $Y=\text{"bottle"}$

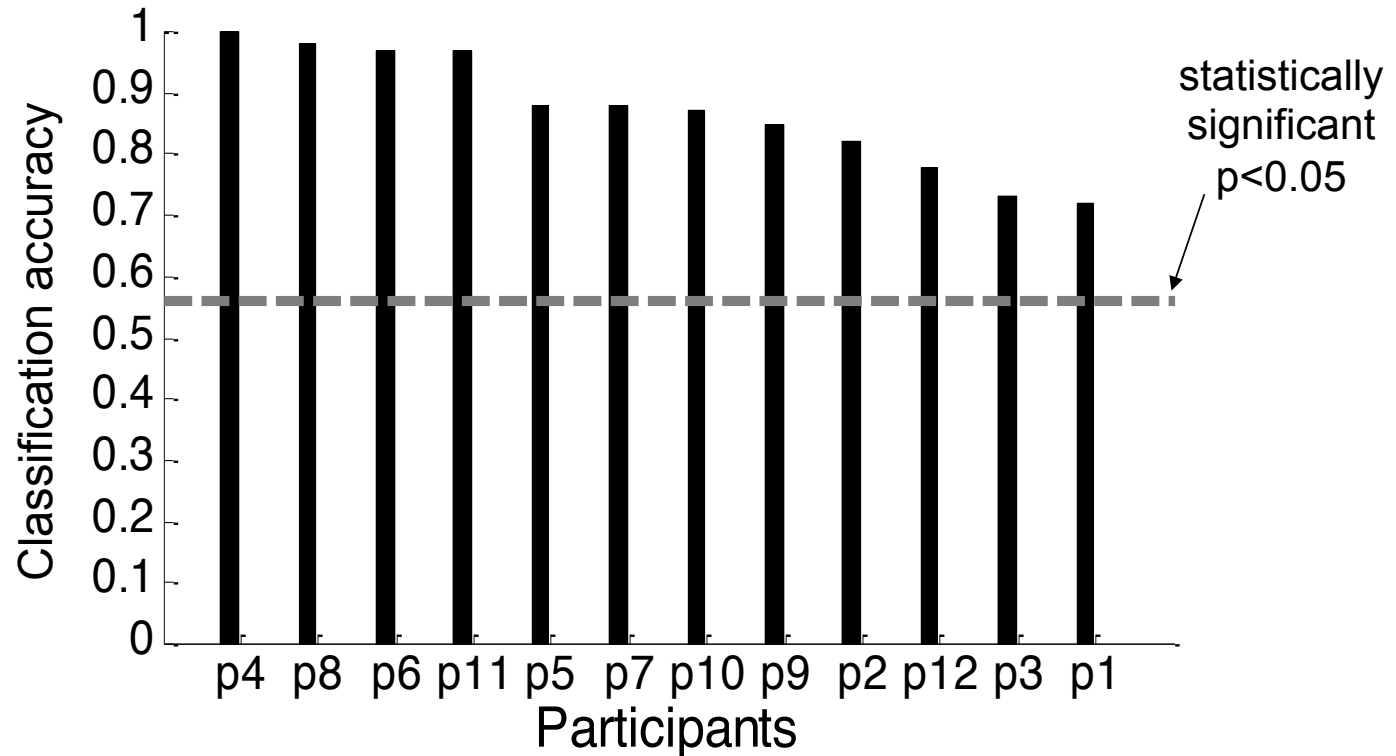


Y is the mental state (reading "house" or "bottle")

X_i are the voxel activities,

this is a plot of the μ 's defining $P(X_i | Y=\text{"bottle"})$

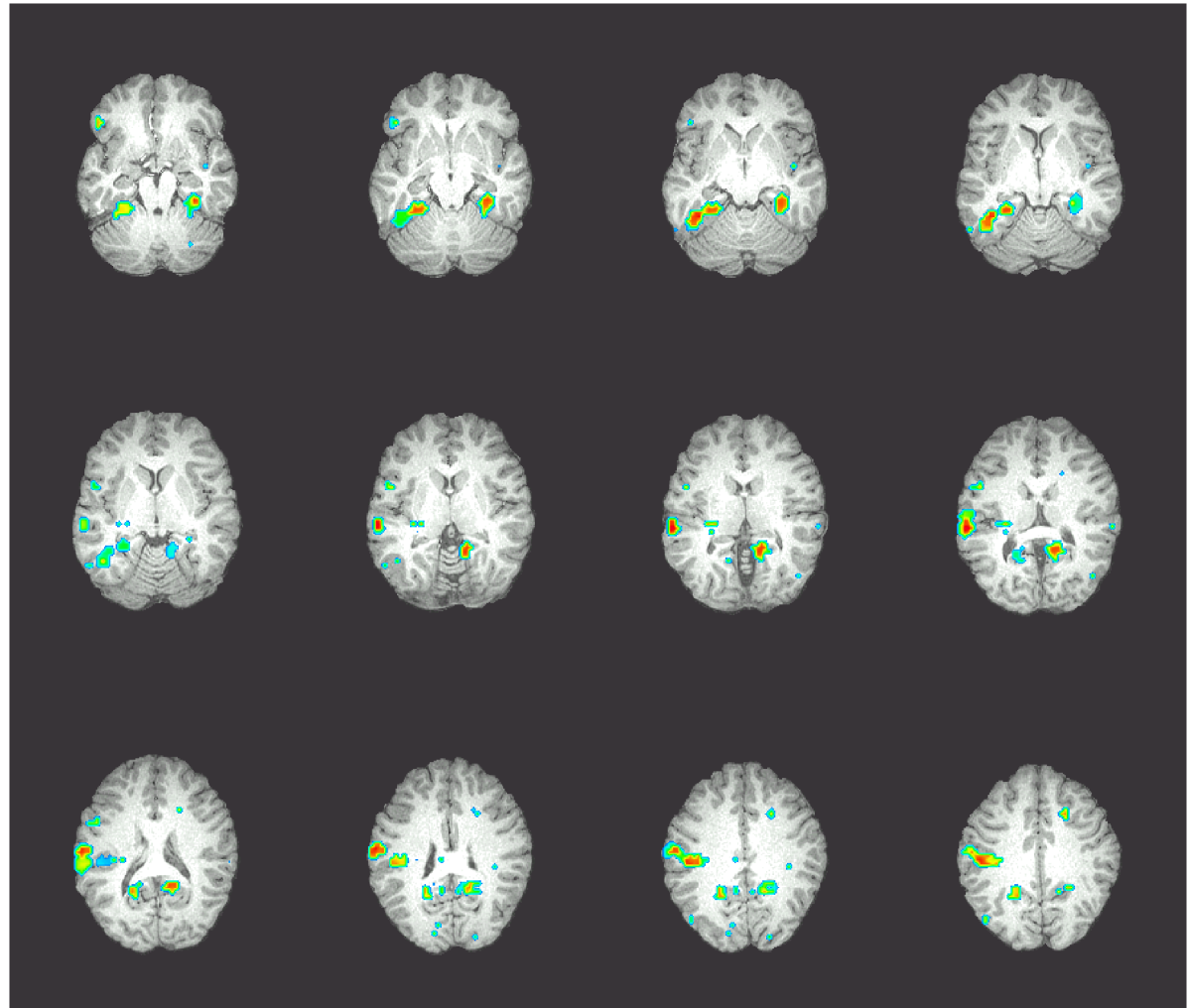
Classification task: is person viewing a “tool” or “building”?



Where is information encoded in the brain?

Accuracies of
cubical
27-voxel
classifiers
centered at
each significant
voxel

[0.7-0.8]



Naïve Bayes with Log Probabilities

$$\begin{aligned}c_{MAP} &= \operatorname{argmax}_c P(c|x_1, \dots, x_n) \\&= \operatorname{argmax}_c P(c) \prod_{i=1}^n P(x_i|c) \\&= \operatorname{argmax}_c \log \left(P(c) \prod_{i=1}^n P(x_i|c) \right) \\&= \operatorname{argmax}_c \log P(c) + \sum_{i=1}^n \log P(x_i|c)\end{aligned}$$

What if we want to calculate posterior log-probabilities?

$$P(c|x_1, \dots, x_n) = \frac{P(c) \prod_{i=1}^n P(x_i|c)}{\sum_{c'} P(c') \prod_{i=1}^n P(x_i|c')}$$

What if we want to calculate posterior log-probabilities?

$$P(c|x_1, \dots, x_n) = \frac{P(c) \prod_{i=1}^n P(x_i|c)}{\sum_{c'} P(c') \prod_{i=1}^n P(x_i|c')}$$

$$\log P(c|x_1, \dots, x_n) = \log \frac{P(c) \prod_{i=1}^n P(x_i|c)}{\sum_{c'} P(c') \prod_{i=1}^n P(x_i|c')}$$

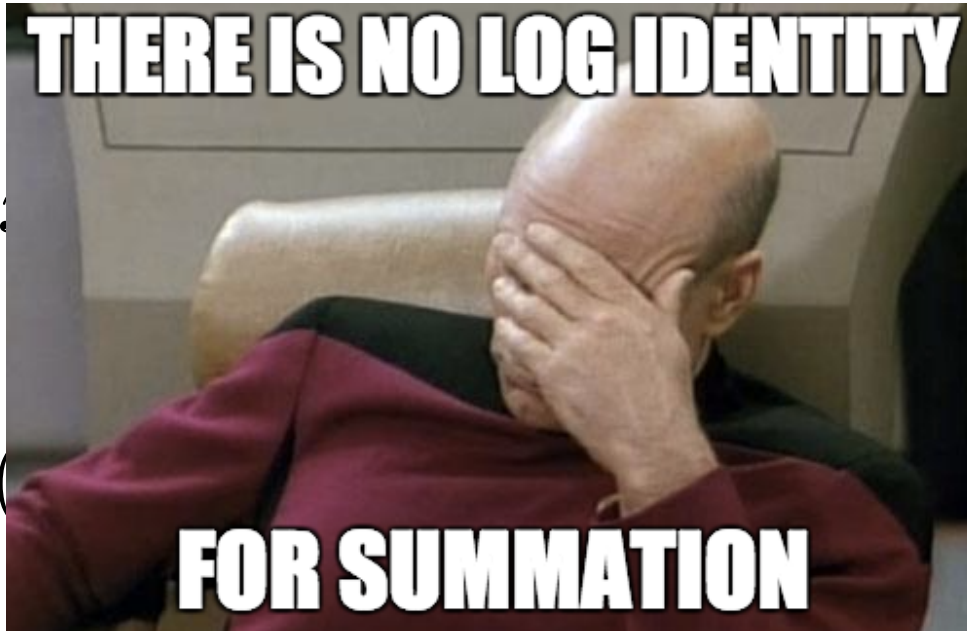
What if we want to calculate posterior log-probabilities?

$$P(c|x_1, \dots, x_n) = \frac{P(c) \prod_{i=1}^n P(x_i|c)}{\sum_{c'} P(c') \prod_{i=1}^n P(x_i|c')}$$

$$\log P(c|x_1, \dots, x_n) = \log \frac{P(c) \prod_{i=1}^n P(x_i|c)}{\sum_{c'} P(c') \prod_{i=1}^n P(x_i|c')}$$

$$= \log P(c) + \sum_{i=1}^n \log P(x_i|c) - \log \left[\sum_{c'} P(c') \prod_{i=1}^n P(x_i|c') \right]$$

What if we want to calculate posterior log-probabilities?



$$P(c|s)$$

$$\log P(c)$$

$$\frac{\prod_{i=1}^n P(x_i|c)}{\prod_{i=1}^n P(x_i|c')}$$

$$\frac{\prod_{i=1}^n P(x_i|c)}{c') \prod_{i=1}^n P(x_i|c')}$$

$$= \log P(c) + \sum_{i=1}^n \log P(x_i|c) - \log \left[\sum_{c'} P(c') \prod_{i=1}^n P(x_i|c') \right]$$

Log Exp Sum Trick

- We have: a bunch of log probabilities.
 - $\log(p_1)$, $\log(p_2)$, $\log(p_3)$, ... $\log(p_n)$
- We want: $\log(p_1 + p_2 + p_3 + \dots p_n)$

Log Exp Sum Trick:

$$\log\left[\sum_i \exp(x_i)\right] = x_{max} + \log\left[\sum_i \exp(x_i - x_{max})\right]$$

Naïve Bayes: What you should know

- Designing classifiers based on Bayes rule
- Conditional independence
 - What it is
 - Why it's important
- Naïve Bayes assumption and its consequences
 - Which (and how many) parameters must be estimated under different generative models (different forms for $P(X|Y)$)
 - and why this matters
- How to train Naïve Bayes classifiers
 - MLE and MAP estimates
 - with discrete and/or continuous inputs X_i
- Log-Exp-Sum Trick

Linear Regression

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

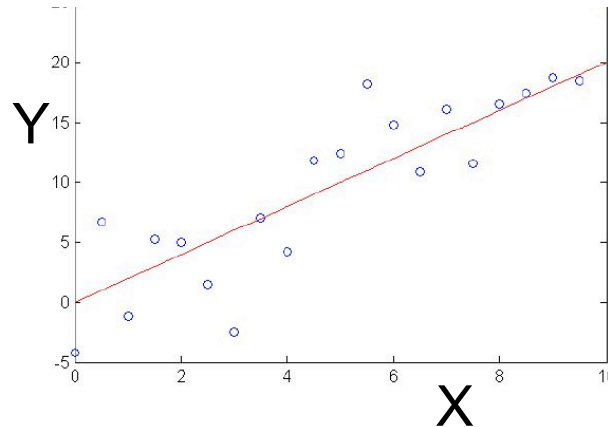
Regression

Wish to learn $f: X \rightarrow Y$, where Y is real, given $\{ \langle x^1, y^1 \rangle \dots \langle x^n, y^n \rangle \}$

Approach:

1. choose some parameterized form for $P(Y|X; \theta)$
(θ is the vector of parameters)
2. derive learning algorithm as MCLE or MAP estimate for θ

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



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

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

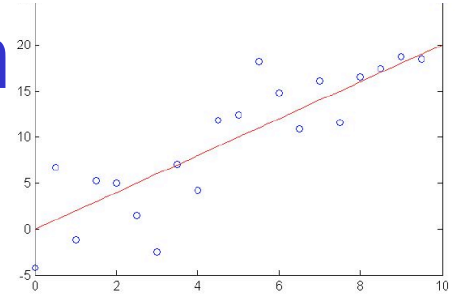
Therefore Y is a random variable that follows the distribution

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

and the expected value of y for any given x is $f(x)$

Training Linear Regression

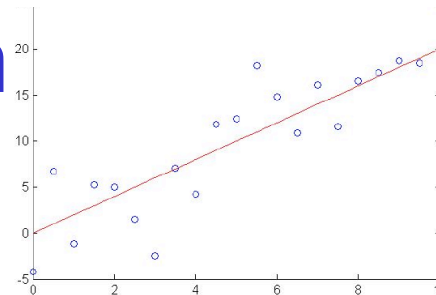
$$p(y|x; W) = N(w_0 + w_1x, \sigma)$$



How can we learn W from the training data?

Training Linear Regression

$$p(y|x; W) = N(w_0 + w_1x, \sigma)$$



How can we learn W from the training data?

Learn 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

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

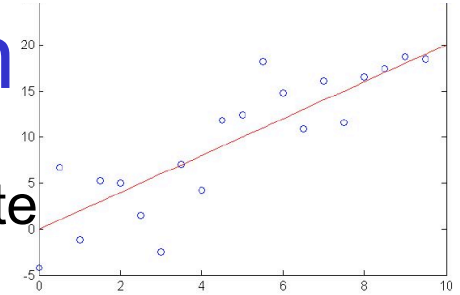
Training Linear Regression

Learn Maximum Conditional Likelihood Estimate

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

where

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



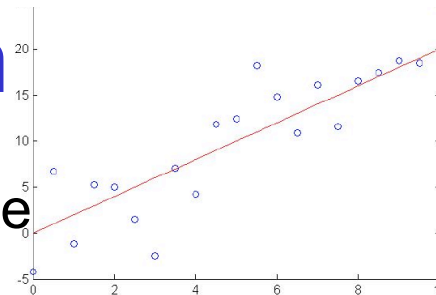
Training Linear Regression

Learn Maximum Conditional Likelihood Estimate

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

where

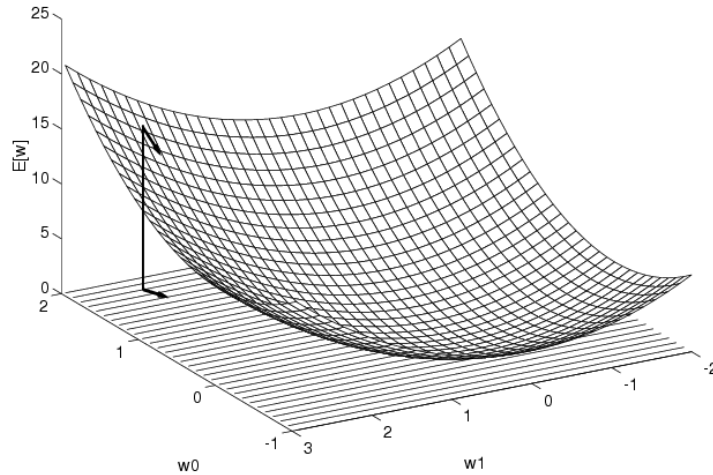
$$p(y|x; W) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{y-f(x;W)}{\sigma}\right)^2}$$



so:

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

Gradient Descent



Gradient

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

Training rule:

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

i.e.,

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

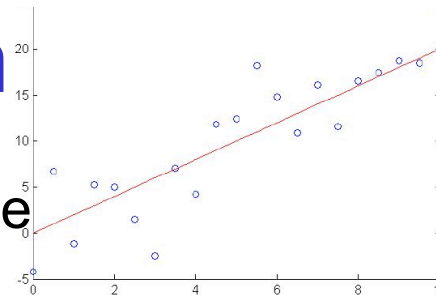
Training Linear Regression

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?

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



Normal Equation

$$w^* = (X^T X)^{-1} X^T y$$

How About MAP instead of MLE?

$$w^* = \arg \max_W \sum_l \ln P(Y^l | X^l; W) + \ln N(W | 0, I)$$

$$= \arg \max_W \sum_l \ln P(Y^l | X^l; W) - c \sum_i w_i^2$$

Regression - What you should know

- MLE -> Sum of Squared Errors
- MAP -> Sum of Squared Errors minus sum of squared weights
- Learning is an optimization problem once we choose objective function
 - Maximize Data Likelihood
 - Maximize Posterior Prob of Weights
- Can use Gradient Descent as General Learning Algorithm