Machine Learning 10-601

Tom M. Mitchell
Machine Learning Department
Carnegie Mellon University

February 4, 2015

Today:
- Generative – discriminative classifiers
- Linear regression
- Decomposition of error into bias, variance, unavoidable

Readings:
- Mitchell: "Naïve Bayes and Logistic Regression" (required)
- Ng and Jordan paper (optional)
- Bishop, Ch 9.1, 9.2 (optional)

Logistic Regression

- Consider learning f: X → Y, where
  - X is a vector of real-valued features, $< X_1 \ldots X_n >$
  - Y is boolean
  - assume all X_i are conditionally independent given Y
  - model $P(X_i \mid Y = y_k)$ as Gaussian $N(\mu_{ik},\sigma_i)$
  - model P(Y) as Bernoulli (π)

- Then P(Y|X) is of this form, and we can directly estimate W

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

- Furthermore, same holds if the X_i are boolean
  - trying proving that to yourself

- Train by gradient ascent estimation of w's (no assumptions!)

MLE vs MAP

- Maximum conditional likelihood estimate

$$W \leftarrow \arg\max_W \ \ln \prod_l P(Y^l | X^l, W)$$

$$w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))$$

- Maximum a posteriori estimate with prior W~N(0,σI)

$$W \leftarrow \arg\max_W \ \ln[P(W) \ \prod_l P(Y^l | X^l, W)]$$

$$w_i \leftarrow w_i - \eta \lambda w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))$$

MAP estimates and Regularization

- Maximum a posteriori estimate with prior W~N(0,σI)

$$W \leftarrow \arg\max_W \ln[P(W) \prod_l P(Y^l|X^l, W)]$$

$$w_i \leftarrow w_i - \eta\lambda w_i + \eta \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1|X^l, W))$$

called a "<u>regularization</u>" term

• helps reduce overfitting, especially when training data is sparse

• keep weights nearer to zero (if P(W) is zero mean Gaussian prior), or whatever the prior suggests

• used very frequently in Logistic Regression

Generative vs. Discriminative Classifiers

Training classifiers involves estimating f: X → Y, or P(Y|X)

Generative classifiers (e.g., Naïve Bayes)

- Assume some functional form for P(Y), P(X|Y)
- Estimate parameters of P(X|Y), P(Y) directly from training data
- Use Bayes rule to calculate P(Y=y |X= x)

Discriminative classifiers (e.g., Logistic regression)

- Assume some functional form for P(Y|X)
- Estimate parameters of P(Y|X) directly from training data

- NOTE: even though our derivation of the form of P(Y|X) made GNB-style assumptions, the *training procedure* for Logistic Regression does not!

Use Naïve Bayes or Logisitic Regression?

Consider

- Restrictiveness of modeling assumptions (how well can we learn with infinite data?)

- Rate of convergence (in amount of training data) toward asymptotic (infinite data) hypothesis
  - i.e., the learning curve

Naïve Bayes vs Logistic Regression

Consider Y boolean, X_i continuous, $X=<X_1 \ldots X_n>$

Number of parameters:

- NB: 4n +1
- LR: n+1

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

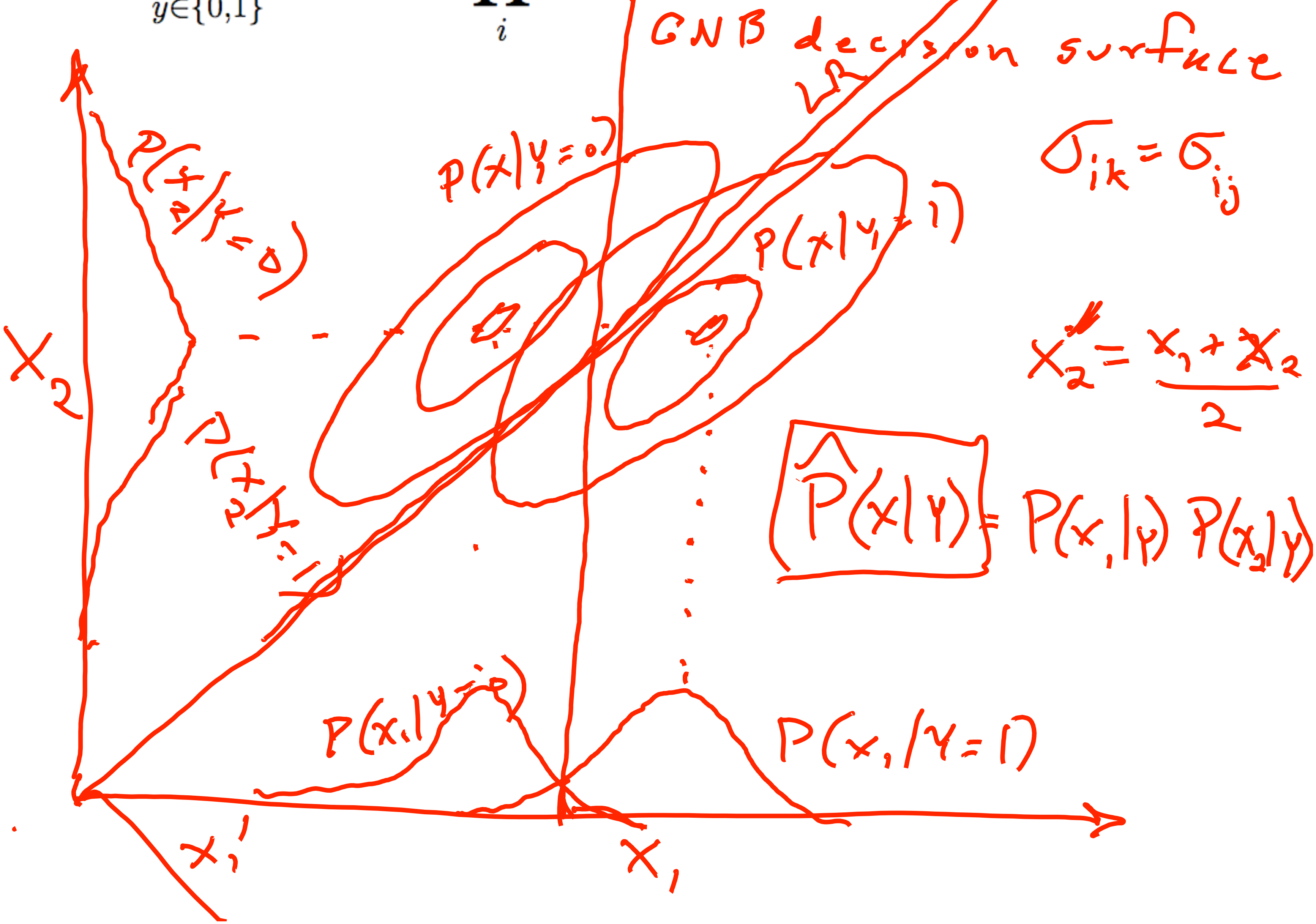
Estimation method:

- NB parameter estimates are uncoupled
- LR parameter estimates are coupled

Gaussian Naïve Bayes – Big Picture

$$Y^{new} \leftarrow \arg\max_{y \in \{0,1\}} P(Y=y) \prod_i P(X_i^{new} | Y=y)$$

assume P(Y=1) = 0.5



Gaussian Naïve Bayes – Big Picture

$$Y^{new} \leftarrow \arg\max_{y\in\{0,1\}} P(Y=y)\prod_i P(X_i^{new}|Y=y)$$

assume P(Y=1) = 0.5

G.Naïve Bayes vs. Logistic Regression

[Ng & Jordan, 2002]

Recall two assumptions deriving form of LR from GNBayes:

1. X_i conditionally independent of X_k given Y
2. $P(X_i \mid Y = y_k) = N(\mu_{ik},\sigma_i)$, ← not $N(\mu_{ik},\sigma_{ik})$

Consider three learning methods:

- GNB (assumption 1 only) – can be nonlinear
- GNB2 (assumption 1 and 2) – linear dec surf
- LR — linear " "

Which method works better if we have *infinite* training data, and...

- Both (1) and (2) are satisfied GNB = GNB2 = LR
- Neither (1) nor (2) is satisfied GNB > GNB2 LR > GNB2
- (1) is satisfied, but not (2) GNB > GNB2 GNB > LR

G.Naïve Bayes vs. Logistic Regression

[Ng & Jordan, 2002]

Recall two assumptions deriving form of LR from GNBayes:

1. X_i conditionally independent of X_k given Y
2. $P(X_i \mid Y = y_k) = N(\mu_{ik},\sigma_i)$, ← not $N(\mu_{ik},\sigma_{ik})$

Consider three learning methods:

- GNB (assumption 1 only) -- decision surface can be non-linear
- GNB2 (assumption 1 and 2) – decision surface linear
- LR -- decision surface linear, trained differently

Which method works better if we have <u>*infinite*</u> training data, and...

- Both (1) and (2) are satisfied: LR = GNB2 = GNB
- Neither (1) nor (2) is satisfied: LR > GNB2, GNB>GNB2
- (1) is satisfied, but not (2) : GNB > LR, LR > GNB2

G.Naïve Bayes vs. Logistic Regression

[Ng & Jordan, 2002]

What if we have only finite training data?

They converge at different rates to their asymptotic (∞ data) error

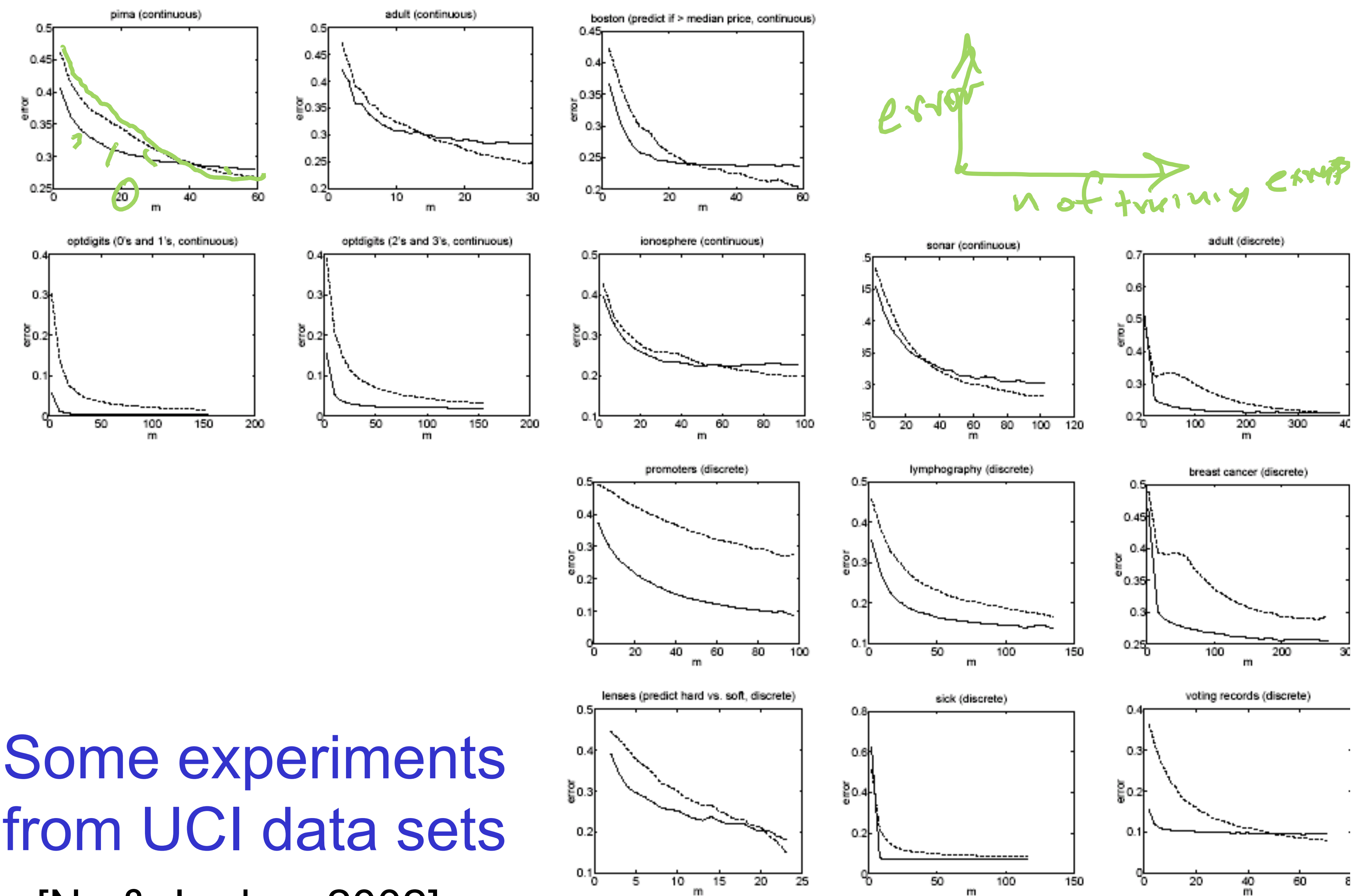
Let $\epsilon_{A,n}$ refer to expected error of learning algorithm A after n training examples

Let d be the number of features: <X_1 … X_d>

$$\epsilon_{LR,n} \leq \epsilon_{LR,\infty} + O\left(\sqrt{\frac{d}{n}}\right)$$

$$\epsilon_{GNB,n} \leq \epsilon_{LR,\infty} + O\left(\sqrt{\frac{\log d}{n}}\right)$$

So, GNB requires n = O(log d) to converge, but LR requires n = O(d)



Some experiments from UCI data sets

[Ng & Jordan, 2002]

Figure 1: Results of 15 experiments on datasets from the UCI Machine Learnin repository. Plots are of generalization error vs. m (averaged over 1000 randor train/test splits). Dashed line is logistic regression; solid line is naive Bayes.

Naïve Bayes vs. Logistic Regression

The bottom line:

GNB2 and LR both use linear decision surfaces, GNB need not

Given infinite data, LR is better than GNB2 because *training procedure* does not make assumptions 1 or 2 (though our derivation of the form of P(Y|X) did).

But GNB2 converges more quickly to its perhaps-less-accurate asymptotic error

And GNB is both more biased (assumption1) and less (no assumption 2) than LR, so either might beat the other

What you should know:

- Logistic regression
  - Functional form follows from Naïve Bayes assumptions
    - For Gaussian Naïve Bayes assuming variance $\sigma_{i,k} = \sigma_i$
    - For discrete-valued Naïve Bayes too
  - But training procedure picks parameters without the conditional independence assumption
  - MCLE training: pick W to maximize P(Y | X, W)
  - MAP training: pick W to maximize P(W | X,Y)
    - regularization: e.g., P(W) ~ N(0,σ)
    - helps reduce overfitting

- Gradient ascent/descent
  - General approach when closed-form solutions for MLE, MAP are unavailable

- Generative vs. Discriminative classifiers
  - Bias vs. variance tradeoff

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

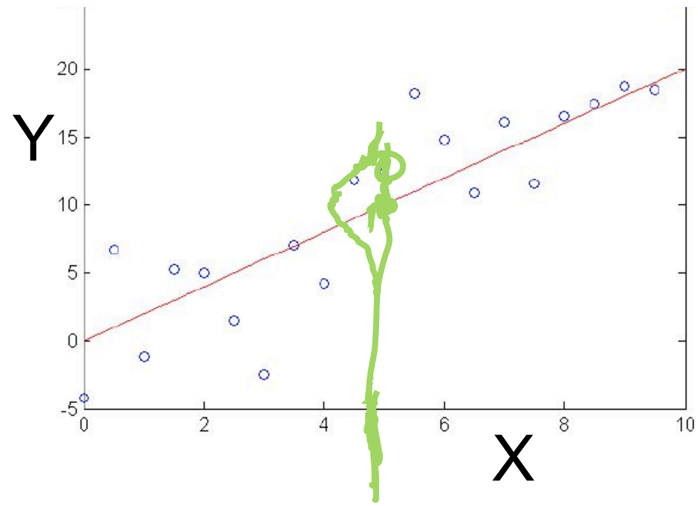
Wish to learn f:X→Y, where Y is real, given $\{<x^1,y^1>\ldots<x^n,y^n>\}$

Approach:

1. choose some parameterized form for $P(Y|X; \theta)$
   (θ is the vector of parameters)

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

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



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

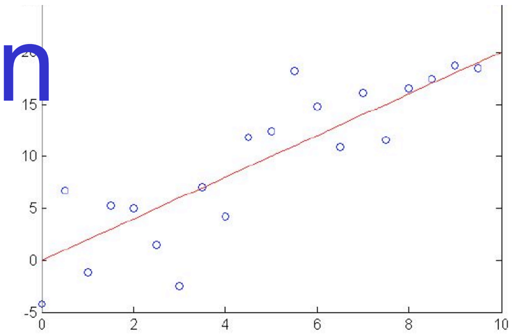
$$y = f(x) + \epsilon \quad \text{where} \quad \epsilon \sim N(0, \sigma)$$

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)

Consider Linear Regression



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

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

$$p(y|x) = N(w_0 + w_1 x, \sigma)$$

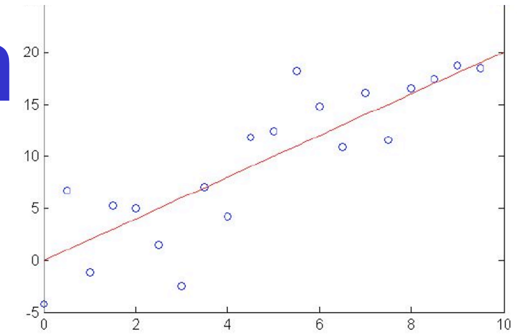
$$E[y|x] = w_0 + w_1 x$$

Notation: to make our parameters explicit, let's write

$$W = < w_0, w_1 >$$

$$p(y|x; W) = N(w_0 + w_1 x, \sigma)$$

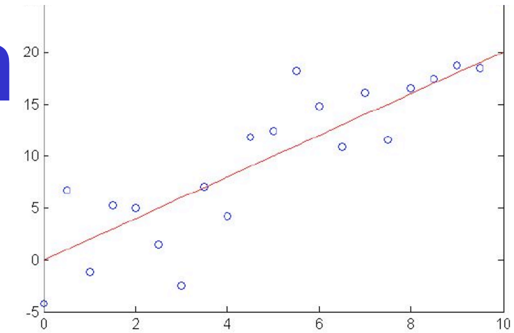
Training Linear Regression



$$p(y|x;W) = N(w_0 + w_1 x, \sigma)$$

How can we learn W from the training data?

Training Linear Regression



$$p(y|x;W) = N(w_0 + w_1 x, \sigma)$$

How can we learn W from the training data?

Learn Maximum Conditional Likelihood Estimate!

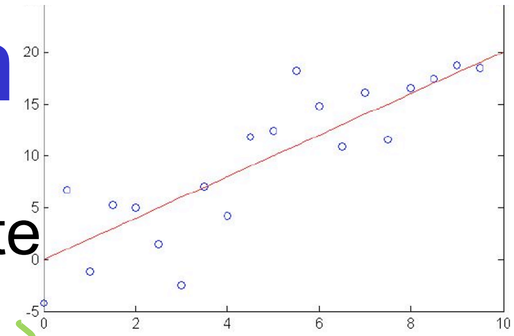
$$W_{MCLE} = \arg\max_{W} \prod_{l} p(y^l | x^l, W)$$

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

where

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

Training Linear Regression



Learn Maximum Conditional Likelihood Estimate

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

$\ell(w)$, $w_0 + w_1 x$

where

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

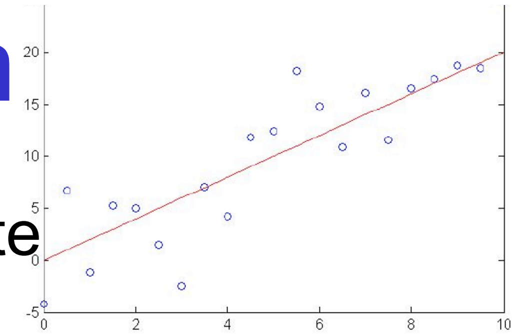
$$= \arg\max_w \sum_l \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} + \left(-\frac{1}{2} \left(\frac{y^l - (w_0 + w_1 x^l)}{\sigma}\right)^2\right)\right)$$

$$= \arg\max_w \sum_l \left(-\frac{1}{2}\left(\frac{y^l - (w_0 + w_1 x^l)}{\sigma}\right)^2\right)$$

$$= \arg\min_w \sum_l \frac{1}{\sigma^2} \left(y^l - (w_0 + w_1 x^l)\right)^2$$

$$\frac{\partial \ell(w)}{\partial w_1} = \sum_l \frac{1}{\sigma^2} 2\left(y^l - (w_0 + w_1 x^l)\right) \cdot x^l$$

Training Linear Regression



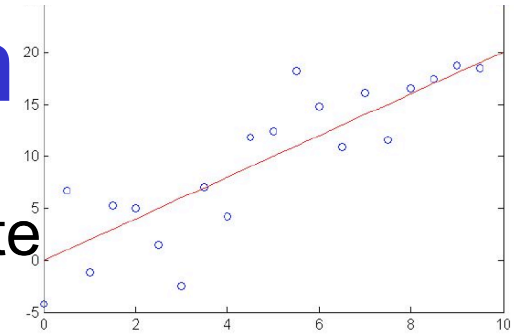
Learn Maximum Conditional Likelihood Estimate

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

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

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

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?

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

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

How about MAP instead of MLE estimate?

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

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

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

2. MAP estimate minimizes SSE plus sum of squared weights

3. Again, learning is an optimization problem once we choose our objective function
   - maximize data likelihood
   - maximize posterior prob of W

4. Again, we can use gradient descent as a general learning algorithm
   - as long as our objective fn is differentiable wrt W
   - though we might learn local optima ins

$f(x) = y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2$

5. Almost nothing we said here required that f(x) be linear in x

Bias/Variance Decomposition of Error

Bias and Variance

given some estimator Y for some parameter θ, we define

the bias of estimator Y = $E[Y] - \theta$

the variance of estimator Y = $E[(Y - E[Y])^2]$

e.g., define Y as the MLE estimator for probability of heads, based on n independent coin flips

biased or unbiased?

variance decreases as sqrt(1/n)

Bias – Variance decomposition of error

Reading: Bishop chapter 9.1, 9.2

- Consider simple regression problem f:X→Y

y = f(x) + ε

noise N(0,σ)

deterministic

What are sources of prediction error?

$$E_D\left[\int_y \int_x (h(x) - f(x))^2 p(y|x)p(x)dydx\right]$$

learned estimate of f(x)

Sources of error

- What if we have perfect learner, infinite data?
  - Our learned h(x) satisfies h(x)=f(x)
  - Still have remaining, *unavoidable error*

$$\sigma^2$$

Sources of error

- What if we have only n training examples?
- What is our expected error
  - Taken over random training sets of size n, drawn from distribution $D{=}p(x,y)$

$$E_D\left[\int_y \int_x (h(x) - f(x))^2 p(y|x)p(x)dydx\right]$$

Sources of error

$$E_D\left[\int_y \int_x (h(x) - f(x))^2 p(y|x)p(x)dydx\right]$$

$$= unavoidableError + bias^2 + variance$$

$$bias^2 = \int (E_D[h(x)] - f(x))^2 p(x)dx$$

$$variance = \int E_D[(h(x) - E_D[h(x)])^2]p(x)dx$$