CSCI 5090/7090- Machine Learning

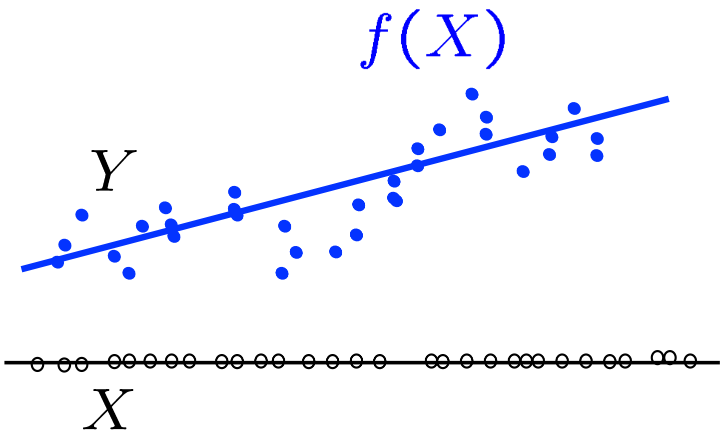
Spring 2018

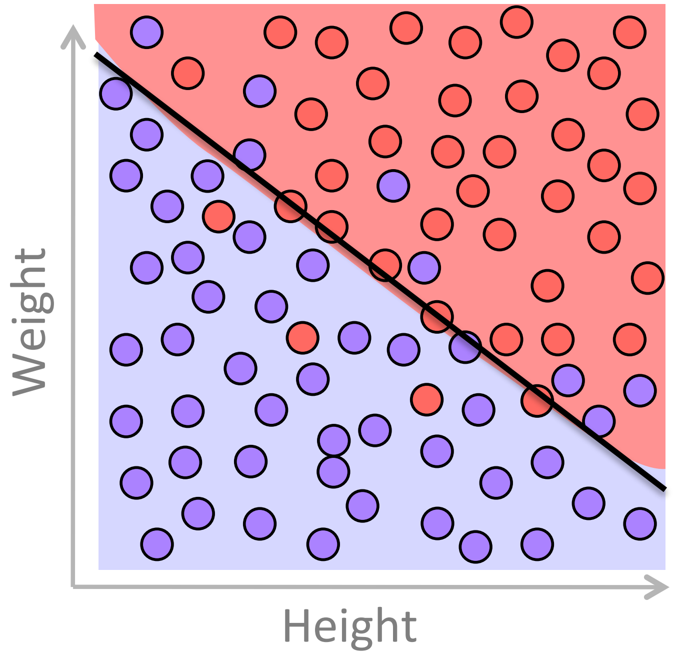
Mehdi Allahyari
Georgia Southern University

Logistic Regression

(slides borrowed from Tom Mitchell, Barnabás Póczos & Aarti Singh

Linear Regression & Linear Classification





Linear fit

Linear decision boundary

Naïve Bayes Recap…

- NB Assumption: $$P(X_1 ... X_d | Y) = \prod_{i=1}^{d} P(X_i | Y)$$

- NB Classifier:

$$f_{NB}(x) = \arg\max_y \prod_{i=1}^{d} P(x_i | y) P(y)$$

- Assume parametric form for $P(X_i | Y)$ and $P(Y)$
  - Estimate parameters using MLE/MAP and plug in

Generative vs. Discriminative Classifiers

Generative classifiers (e.g. Naïve Bayes)

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

But arg max_Y P(X|Y) P(Y) = arg max_Y P(Y|X)

Why not learn P(Y|X) directly? Or better yet, why not learn the decision boundary directly?

Discriminative classifiers (e.g. Logistic Regression)

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

Logistic Regression

Idea:

- Naïve Bayes allows computing P(Y|X) by learning P(Y) and P(X|Y)

- Why not learn P(Y|X) directly?

GNB with equal variance is a linear classifier

- 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)

- What does that imply about the form of P(Y|X)?

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

Derive form for P(Y|X) for Gaussian $P(X_i|Y=y_k)$ assuming $\sigma_{ik} = \sigma_i$

$$P(Y=1|X) = \frac{P(Y=1)P(X|Y=1)}{P(Y=1)P(X|Y=1) + P(Y=0)P(X|Y=0)}$$

$$= \frac{1}{1 + \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}}$$

$$= \frac{1}{1 + \exp(\ln \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)})}$$

$$= \frac{1}{1 + \exp(\ (\ln \frac{1-\pi}{\pi}) + \boxed{\sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)})}}$$

$$P(x \mid y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} \ e^{\frac{-(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$

$$\boxed{\sum_i \left(\frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} X_i + \frac{\mu_{i1}^2 - \mu_{i0}^2)}{2\sigma_i^2} \right)}$$

$$P(Y=1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^n w_i X_i)}$$

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

implies

$$P(Y=0|X=<X_1,...X_n>) =$$

implies

$$\frac{P(Y=0|X)}{P(Y=1|X)} =$$

implies

$$\ln \frac{P(Y=0|X)}{P(Y=1|X)} =$$

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

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

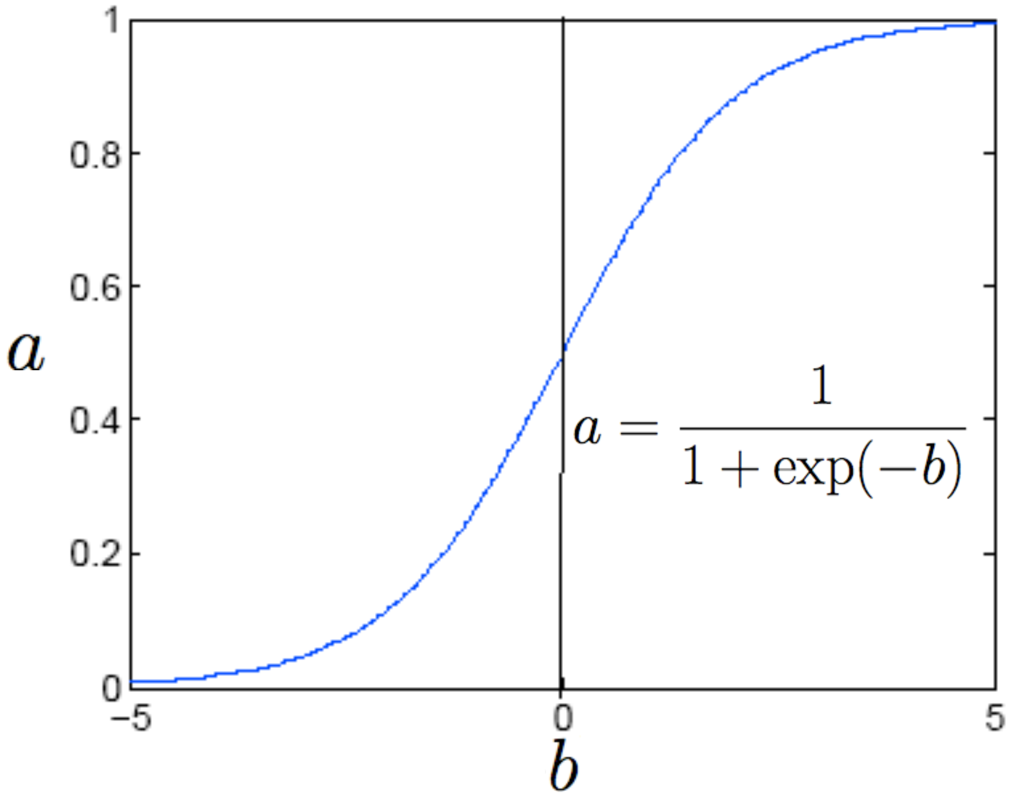
$$\frac{P(Y=0|X)}{P(Y=1|X)} = exp(w_0+\sum_i w_i X_i)$$

linear classification rule!

implies

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

Logistic Function



$$P(Y=1|X) = \frac{1}{1+\exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

Logistic regression more generally

- Logistic regression when Y not boolean (but still discrete-valued).
- Now $y \in \{y_1 \ldots y_R\}$: learn R-1 sets of weights

for $k<R$ $$P(Y = y_k|X) = \frac{\exp(w_{k0} + \sum_{i=1}^{n} w_{ki} X_i)}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^{n} w_{ji} X_i)}$$

for $k=R$ $$P(Y = y_R|X) = \frac{1}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^{n} w_{ji} X_i)}$$

Training Logistic Regression: MCLE

We'll focus on binary classification:

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

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

- we have L training examples: $\{\langle X^1, Y^1\rangle, \ldots \langle X^L, Y^L\rangle\}$

- maximum likelihood estimate for parameters W

$$W_{MLE} = \arg\max_W P(<X^1,Y^1> \ldots <X^L,Y^L> |W)$$
$$= \arg\max_W \prod_l P(<X^l,Y^l> |W)$$

But there is a problem …

Don't have a model for P(X) or P(X|Y) – only for P(Y|X)

Training Logistic Regression: MCLE

We'll focus on binary classification:

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

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

- we have L training examples: $\{\langle X^1, Y^1\rangle, \ldots \langle X^L, Y^L\rangle\}$
- maximum likelihood estimate for parameters W

$$W_{MLE} = \arg\max_W P(<X^1,Y^1>\ldots<X^L,Y^L>|W)$$
$$= \arg\max_W \prod_l P(<X^l,Y^l>|W)$$

- maximum <u>conditional</u> likelihood estimate

Training Logistic Regression: MCLE

- Choose parameters W=<w_0, ... w_n> to maximize conditional likelihood of training data

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

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

- Training data D = $\{\langle X^1, Y^1\rangle, \ldots \langle X^L, Y^L\rangle\}$
- Data likelihood = $\prod_l P(X^l, Y^l|W)$
- Data conditional likelihood = $\prod_l P(Y^l|X^l, W)$

$$W_{MCLE} = \arg\max_W \prod_l P(Y^l|W, X^l)$$

Expressing Conditional Log Likelihood

$$l(W) \equiv \ln \prod_l P(Y^l|X^l, W) = \sum_l \ln P(Y^l|X^l, W)$$

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

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

$$\begin{aligned} l(W) &= \sum_l Y^l \ln P(Y^l = 1|X^l, W) + (1 - Y^l) \ln P(Y^l = 0|X^l, W) \\ &= \sum_l Y^l \ln \frac{P(Y^l = 1|X^l, W)}{P(Y^l = 0|X^l, W)} + \ln P(Y^l = 0|X^l, W) \\ &= \sum_l Y^l (w_0 + \sum_i^n w_i X_i^l) - \ln(1 + exp(w_0 + \sum_i^n w_i X_i^l)) \end{aligned}$$

Maximizing Conditional Log Likelihood

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

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

$$\begin{aligned} l(W) &\equiv \ln \prod_l P(Y^l|X^l, W) \\ &= \sum_l Y^l (w_0 + \sum_i^n w_i X_i^l) - \ln(1 + exp(w_0 + \sum_i^n w_i X_i^l)) \end{aligned}$$

Good news: $l(\mathbf{w})$ is a concave function of **w**

→ no locally optimal solutions!

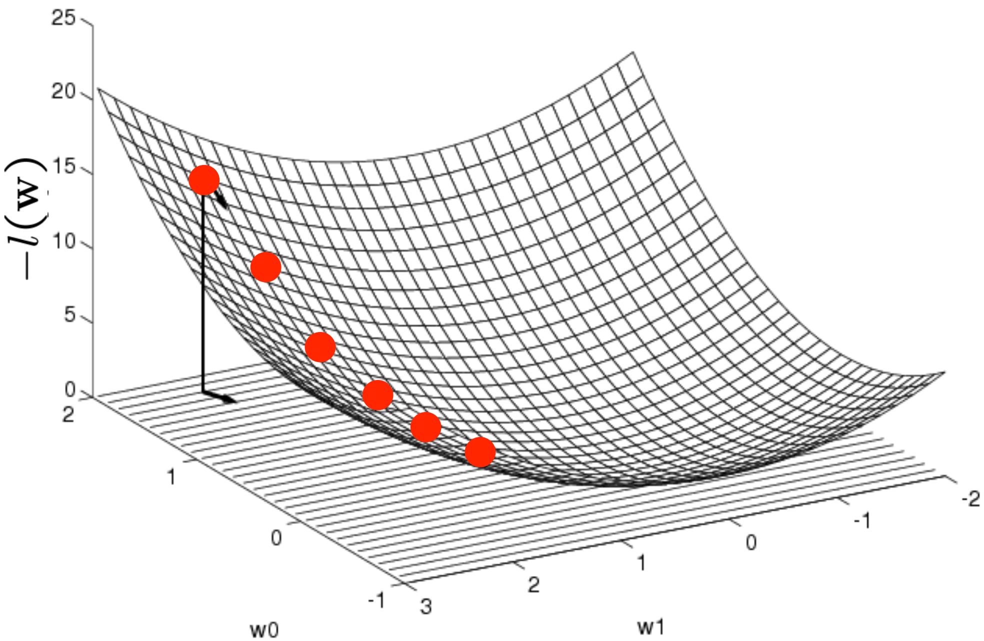
Bad news: no closed-form solution to maximize $l(\mathbf{w})$

Good news: concave functions "easy" to optimize

Optimizing concave/convex functions

- Conditional likelihood for Logistic Regression is concave
- Maximum of a concave function = minimum of a convex function

Gradient Ascent (concave)/ Gradient Descent (convex)



Gradient:

$$\nabla_{\mathbf{w}} l(\mathbf{w}) = [\frac{\partial l(\mathbf{w})}{\partial w_0}, \ldots, \frac{\partial l(\mathbf{w})}{\partial w_n}]'$$

Update rule:

Learning rate, η>0

$$\Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} l(\mathbf{w})$$

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left. \frac{\partial l(\mathbf{w})}{\partial w_i} \right|_t$$

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

Do until satisfied:

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

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

Stochastic can be much faster when D is very large

Intermediate approach: use error over subsets of D

Maximize Conditional Log Likelihood: Gradient Ascent

$$
\begin{aligned}
l(W) &\equiv \ln \prod_l P(Y^l | X^l, W) \\
&= \sum_l Y^l (w_0 + \sum_i^n w_i X_i^l) - \ln(1 + exp(w_0 + \sum_i^n w_i X_i^l))
\end{aligned}
$$

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

Maximize Conditional Log Likelihood: Gradient Ascent

$$
\begin{aligned}
l(W) &\equiv \ln \prod_l P(Y^l | X^l, W) \\
&= \sum_l Y^l (w_0 + \sum_i^n w_i X_i^l) - \ln(1 + exp(w_0 + \sum_i^n w_i X_i^l))
\end{aligned}
$$

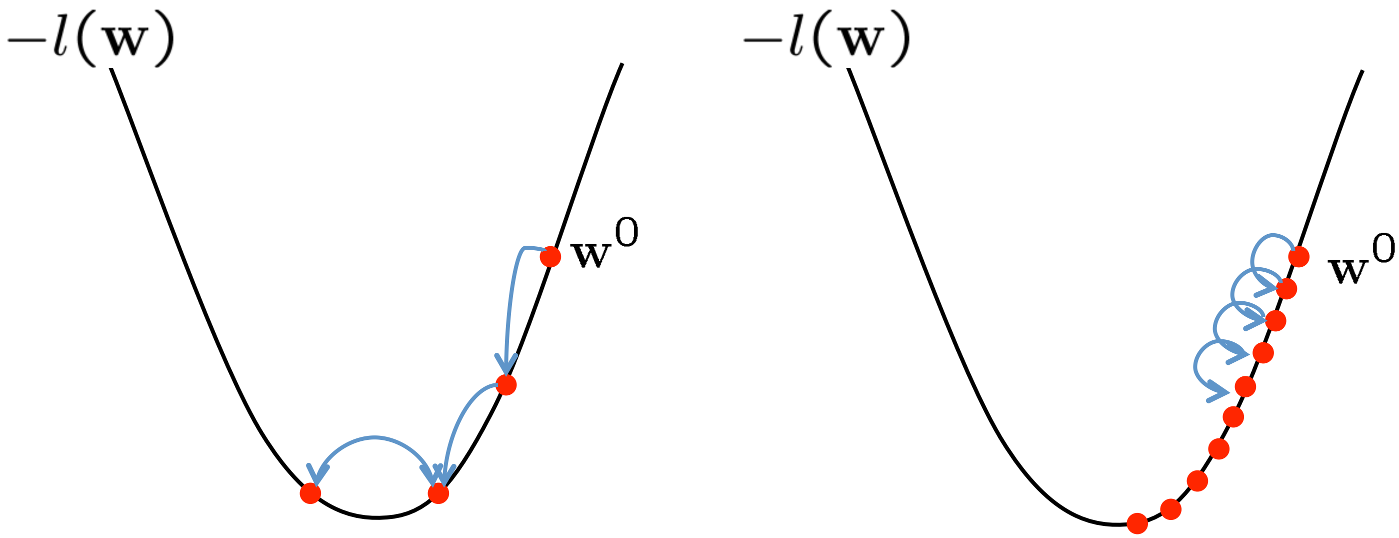
$$
\frac{\partial l(W)}{\partial w_i} = \sum_l X_i^l (Y^l - \hat{P}(Y^l = 1 | X^l, W))
$$

Gradient ascent algorithm: iterate until change < ε

For all i, repeat

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

Effect of step size η



Large η => Fast convergence but larger residual error
Also possible oscillations

Small η => Slow convergence but small residual error

That's all for M(C)LE. How about MAP?

- One common approach is to define priors on W
  - Normal distribution, zero mean, identity covariance
- Helps avoid very large weights and overfitting
- MAP estimate

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

- let's assume Gaussian prior: W ~ N(0, σ)

MLE vs. MAP

- Maximum conditional likelihood estimate

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

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

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

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

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

MAP estimates and Regularization

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

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

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

called a "<u>regularization</u>" term
- helps reduce overfitting
- keep weights nearer to zero (if P(W) is zero mean Gaussian prior), or whatever the prior suggests
- used very frequently in Logistic Regression

The Bottom Line

- 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

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(X|Y), P(X)
- Estimate parameters of P(X|Y), P(X) directly from training data
- Use Bayes rule to calculate P(Y|X= x_i)

Discriminative classifiers (e.g., Logistic regression)

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

Use Naïve Bayes or Logisitic Regression?

Consider

- Restrictiveness of modeling assumptions

- Rate of convergence (in amount of training data) toward asymptotic hypothesis

Use Naïve Bayes or Logisitic Regression?

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

Number of parameters to estimate:

- NB:

- LR:

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

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

Use Naïve Bayes or Logisitic Regression?

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

Number of parameters:

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

Estimation method:

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

G.Naïve Bayes vs. Logistic Regression

Recall two assumptions deriving form of LR from GNBayes:

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

Consider three learning methods:

- GNB (assumption 1 only)
- GNB2 (assumption 1 and 2)
- LR

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

- Both (1) and (2) are satisfied
- Neither (1) nor (2) is satisfied
- (1) is satisfied, but not (2)

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. $P(X_i \mid Y = y_k) = N(\mu_{ik},\sigma_i)$, $\leftarrow$ 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 without assumption 1.

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

- Both (1) and (2) are satisfied: LR = GNB2 = GNB

- (1) is satisfied, but not (2) : GNB > GNB2, GNB > LR, LR > GNB2

- Neither (1) nor (2) is satisfied: GNB>GNB2, LR > GNB2, LR><GNB

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

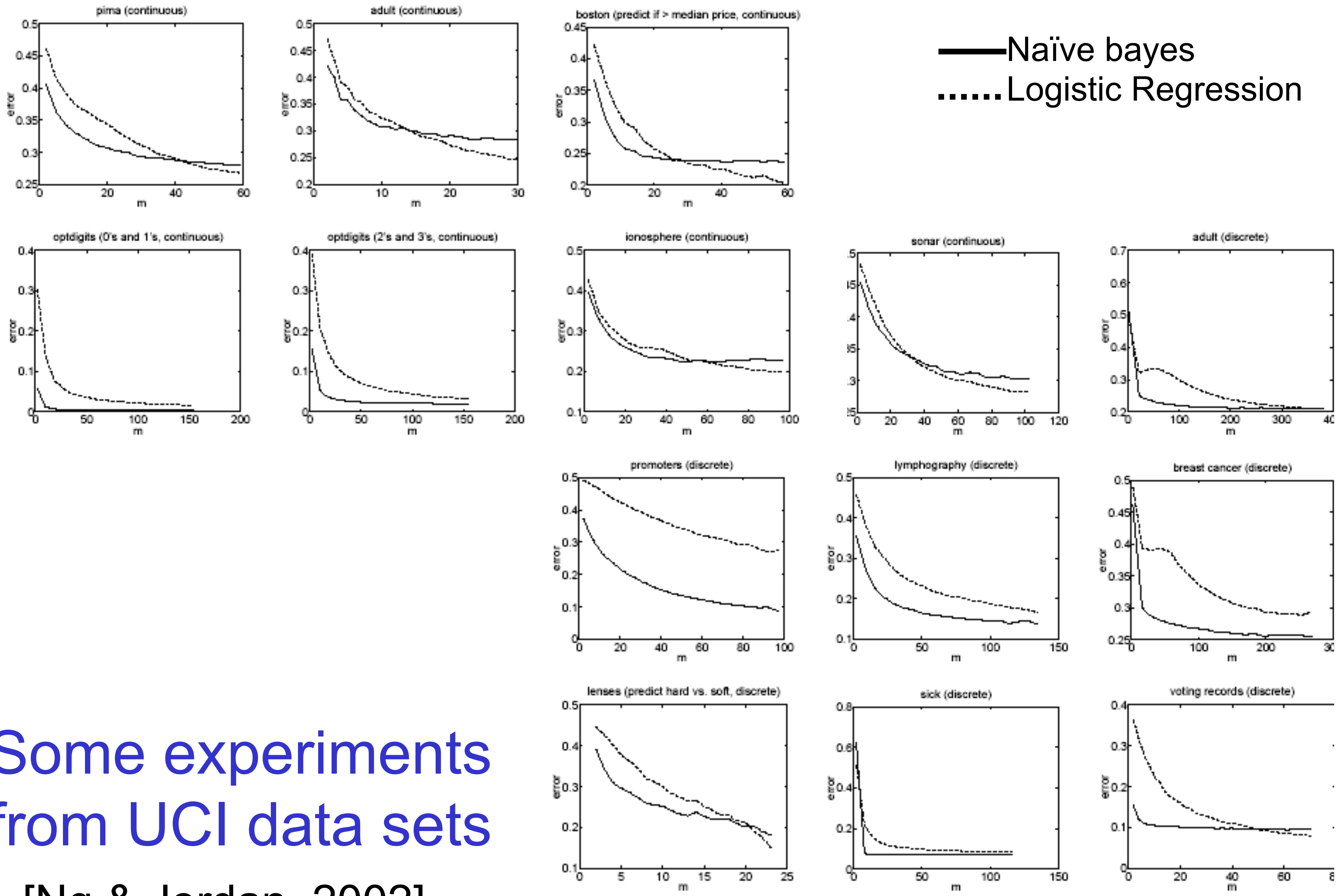
Let $\epsilon_{A,n}$ refer to expected error of learning algorithm A after *n* training examples

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

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

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

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 or equal to 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 outperform 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 making conditional independence assumption
  - MLE training: pick W to maximize P(Y | X, W)
  - MAP training: pick W to maximize P(W | X,Y)
    - 'regularization'
    - helps reduce overfitting

- Gradient ascent/descent
  - General approach when closed-form solutions unavailable

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

What you should know:

- Gaussian Naïve Bayes with class-independent variances representationally equivalent to LR
  - Solution differs because of objective (loss) function
- In general, NB and LR make different assumptions
  - NB: Features independent given class ! assumption on P(**X**|Y)
  - LR: Functional form of P(Y|**X**), no assumption on P(**X**|Y)
- LR is a linear classifier
  - decision rule is a hyperplane
- LR optimized by conditional likelihood
  - no closed-form solution
  - concave ! global optimum with gradient ascent
  - Maximum conditional a posteriori corresponds to regularization
- Convergence rates
  - GNB (usually) needs less data
  - LR (usually) gets to better solutions in the limit