ECE 5984: Introduction to Machine Learning

Topics:

- Classification: Logistic Regression
- NB & LR connections

Readings: Barber 17.4

Dhruv Batra

Virginia Tech

Administrativia

- HW2
  - Due: Friday ~~03/06~~, 03/15, 11:55pm
  - Implement linear regression, Naïve Bayes, Logistic Regression

- Need a couple of catch-up lectures
  - How about 4-6pm?

Recap of last time

Naïve Bayes
(your first probabilistic classifier)

Classification

- **Learn**: $h:\mathbf{X} \mapsto Y$
  - **X** – features
  - Y – target classes

- Suppose you know $P(Y|\mathbf{X})$ exactly, how should you classify?
  - Bayes classifier:

- **Why?**

Slide Credit: Carlos Guestrin

Error Decomposition

- Approximation/Modeling Error
  - You approximated reality with model
- Estimation Error
  - You tried to learn model with finite data
- Optimization Error
  - You were lazy and couldn't/didn't optimize to completion
- Bayes Error
  - Reality just sucks
  - http://psych.hanover.edu/JavaTest/SDT/ROC.html

Generative vs. Discriminative

- Using Bayes rule, optimal classifier

$$h^*(\mathbf{x}) = \underset{c}{\operatorname{argmax}} \{\log p(\mathbf{x}|y=c) + \log p(y=c)\}$$

- Generative Approach (Naïve Bayes)
  - Estimate p(x|y) and p(y)
  - Use Bayes Rule to predict y

- Discriminative Approach
  - Estimate p(y|x) directly (Logistic Regression)
  - Learn “discriminant” function h(x) (Support Vector Machine)

The Naïve Bayes assumption

- Naïve Bayes assumption:
  - Features are independent given class:

$$P(X_1, X_2|Y) = P(X_1|X_2, Y)P(X_2|Y)$$
$$= P(X_1|Y)P(X_2|Y)$$

  - More generally:

$$P(X_1 ... X_d|Y) = \prod_i P(X_i|Y)$$

- How many parameters now?
  - Suppose **X** is composed of *d* binary features

Generative vs. Discriminative

- Using Bayes rule, optimal classifier

$$h^*(\mathbf{x}) = \underset{c}{\operatorname{argmax}} \{\log p(\mathbf{x}|y=c) + \log p(y=c)\}$$

- Generative Approach (Naïve Bayes)
  - Estimate p(x|y) and p(y)
  - Use Bayes Rule to predict y

- Discriminative Approach
  - Estimate p(y|x) directly (Logistic Regression)
  - Learn “discriminant” function h(x) (Support Vector Machine)

Today: Logistic Regression

- Main idea
  - Think about a 2 class problem {0,1}
  - Can we regress to P(Y=1 | X=x)?

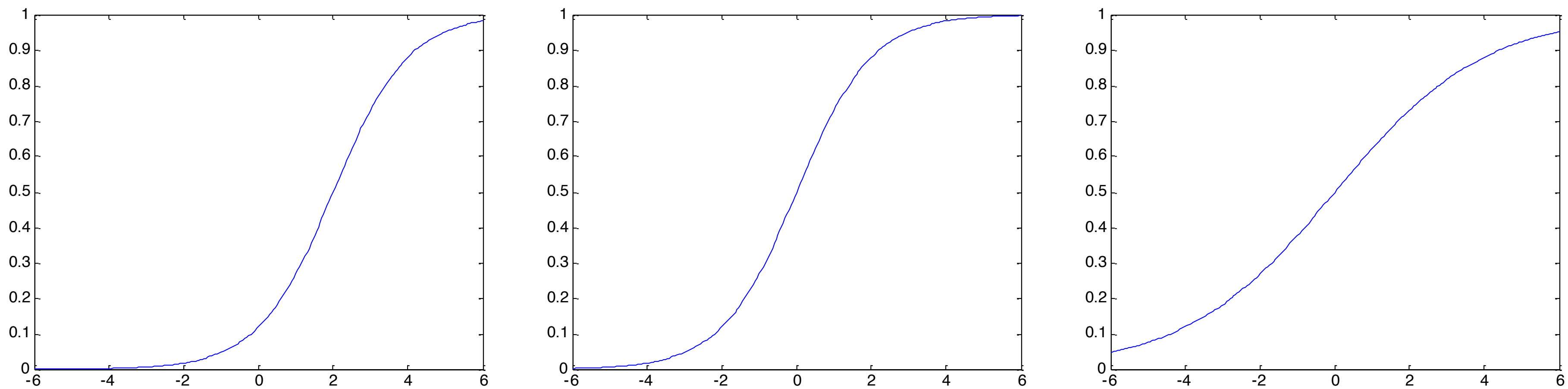
- Meet the Logistic or Sigmoid function
  - Crunches real numbers down to 0-1

- Model
  - In regression: $y \sim N(w'x, \lambda^2)$
  - Logistic Regression: $y \sim \text{Bernoulli}(\sigma(w'x))$

Understanding the sigmoid

$$\sigma(w_0 + \sum_i w_i x_i) = \frac{1}{1 + e^{-w_0 - \sum_i w_i x_i}}$$

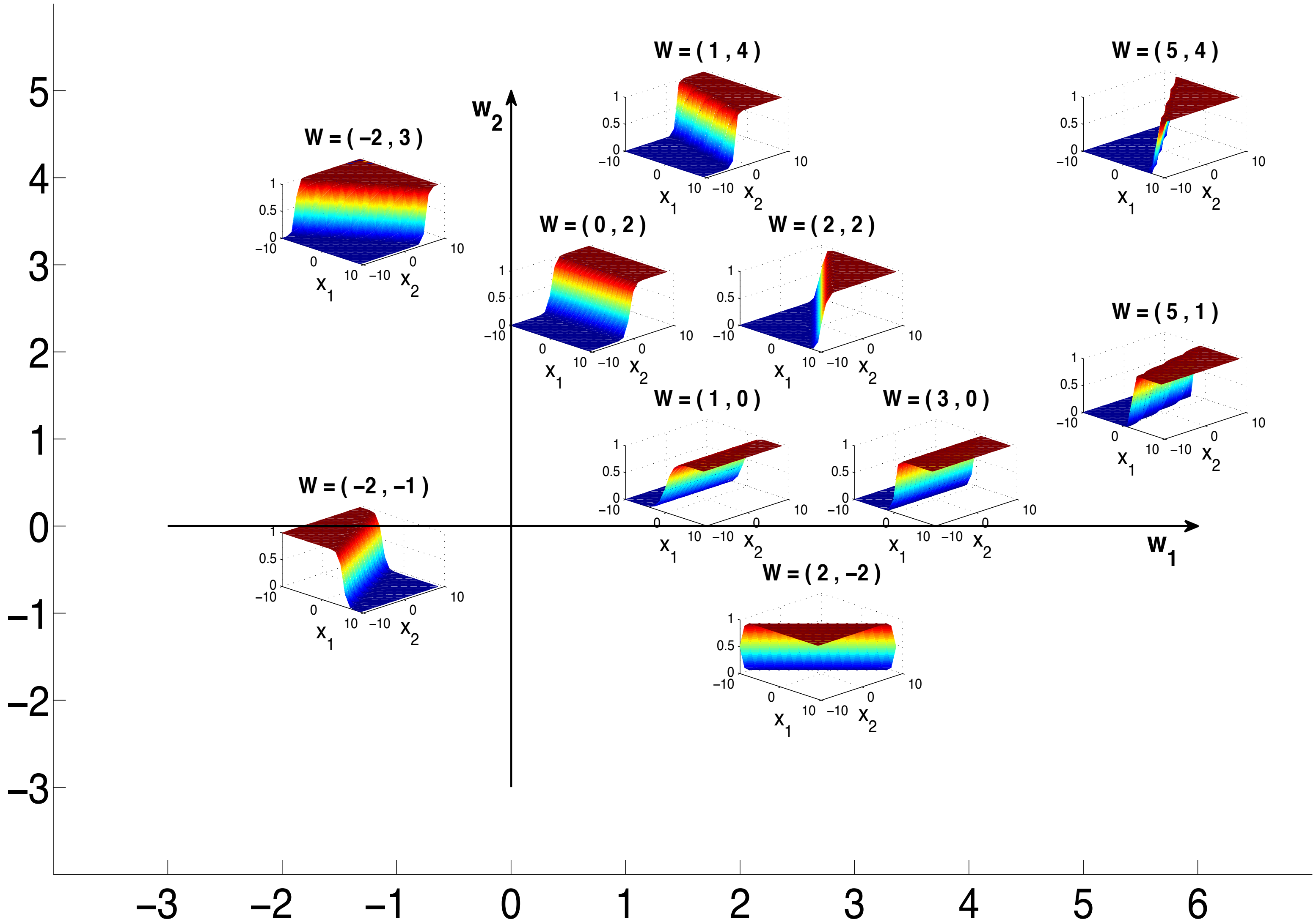
w_0=2, w_1=1 w_0=0, w_1=1 w_0=0, w_1=0.5



Logistic Regression – a Linear classifier

- Demo
  - http://www.cs.technion.ac.il/~rani/LocBoost/

Visualization



Expressing Conditional Log Likelihood

$$l(\mathbf{w}) \equiv \sum_j \ln P(y^j|\mathbf{x}^j, \mathbf{w})$$

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

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

$$l(\mathbf{w}) \;=\; \sum_j y^j \ln P(y^j=1|\mathbf{x}^j,\mathbf{w}) + (1-y^j)\ln P(y^j=0|\mathbf{x}^j,\mathbf{w})$$

Maximizing Conditional Log Likelihood

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

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

$$l(\mathbf{w}) \equiv \ln \prod_j P(y^j|\mathbf{x}^j,\mathbf{w})$$

$$= \sum_j y^j(w_0+\sum_i^d w_i x_i^j) - \ln(1+exp(w_0+\sum_i^d w_i x_i^j))$$

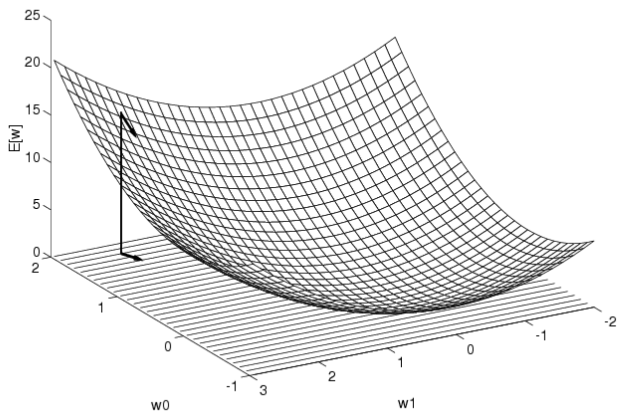
Bad news: no closed-form solution to maximize *l*(**w**)

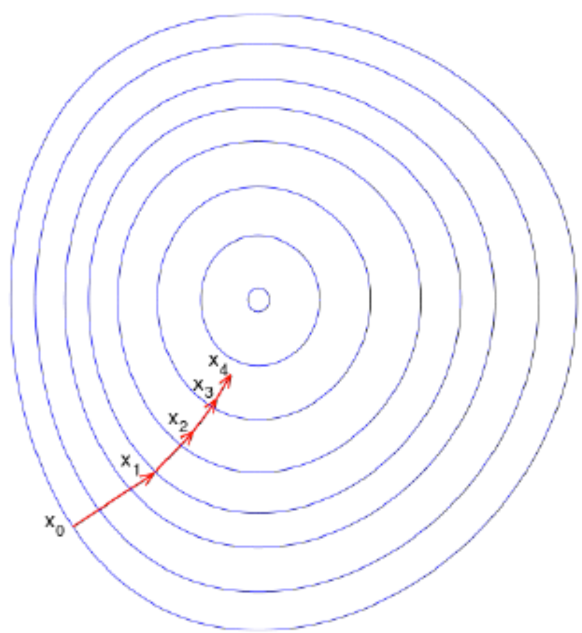
Good news: *l*(**w**) is concave function of **w!**

Gradient Descent

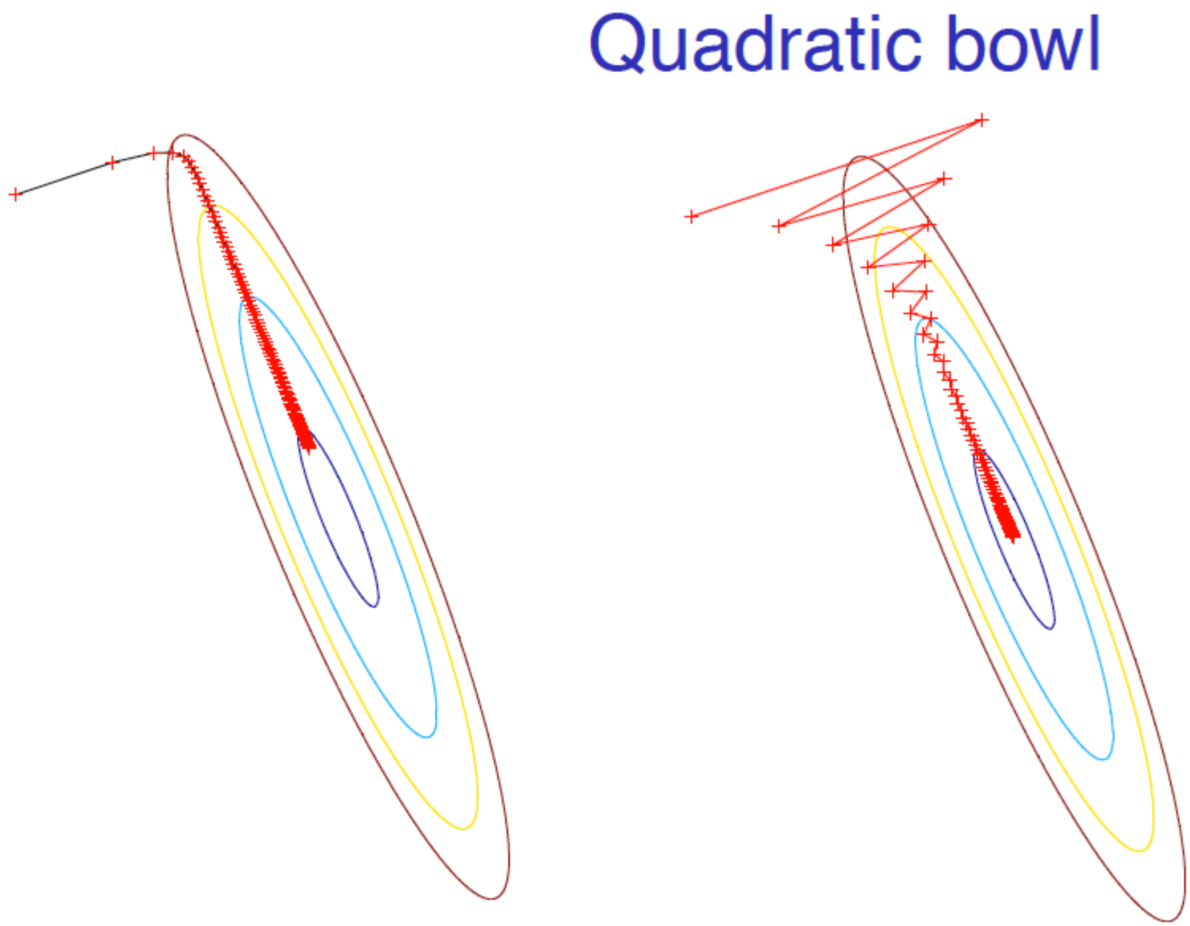
- Choose a starting point w_0 when $t = 0$ and the desired tolerance ϵ.
- Repeat until $\|\nabla f(w_t)\| \leq \epsilon$ is satisfied

$$w_{t+1} = w_t - \eta_t \nabla f(w_t)$$





Careful about step-size



$\eta = .1$

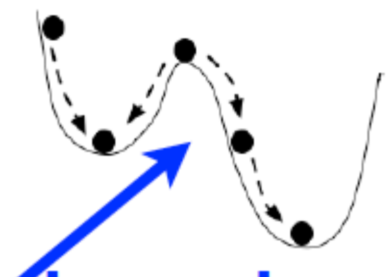
$\eta = .3$

Local vs. global optimal

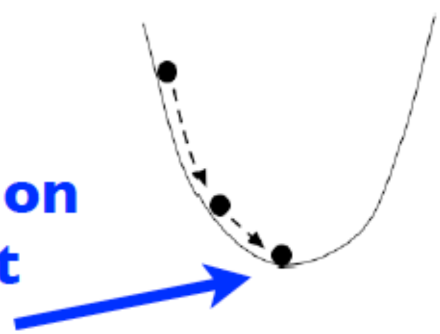
For general objective functions f(x)

We get local optimum

Consider rolling a ball on a hill



depends on where you start



does not depend on where you start

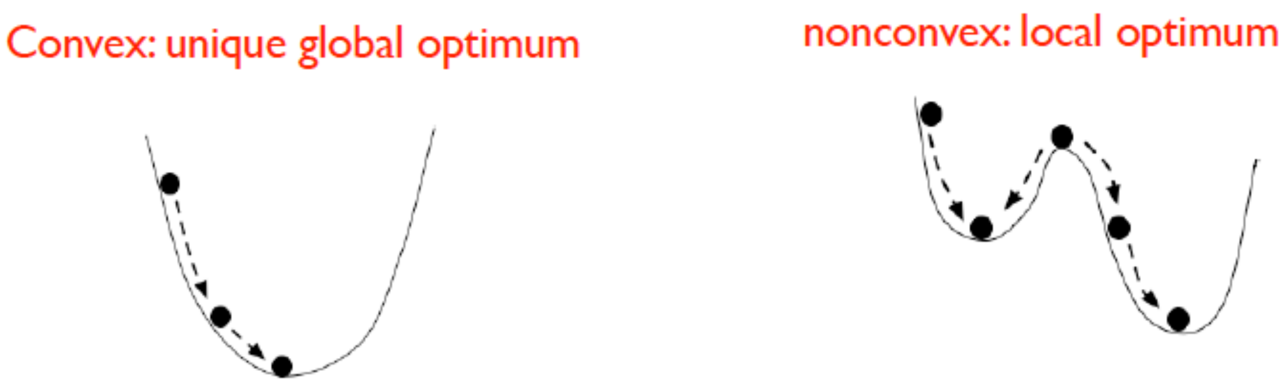
When does it work?

Local vs. global optimal

In practice, convexity can be a very nice thing

In general, convex problems -- minimizing a convex function over a convex set -- can be solved numerically very efficiently

This is advantageous especially if stationary points cannot be found analytically in closed-form



Convex Functions

- $f : \Re^d \rightarrow \Re$ is a convex function if domain of f is a convex set and for all $\lambda \in [0, 1]$

$$f(\lambda w_1 + (1 - \lambda) w_2) \leq \lambda f(w_1) + (1 - \lambda) f(w_2)$$

Multivariate functions

Definition

$f(\boldsymbol{x})$ is convex if

$$f(\lambda \boldsymbol{a} + (1-\lambda)\boldsymbol{b}) \leq \lambda f(\boldsymbol{a}) + (1-\lambda) f(\boldsymbol{b})$$

How to determine convexity in this case?

Second-order derivative becomes Hessian matrix

$$\boldsymbol{H} = \begin{bmatrix} \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1^2} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_D} \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_2} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2^2} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_D} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_D} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_2 \partial x_D} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_D^2} \end{bmatrix}$$

Convexity for multivariate function

If the Hessian is positive semidefinite, then the function is convex

Ex: $$f(\boldsymbol{x}) = \frac{x_1^2}{x_2}$$

$$\boldsymbol{H} = \begin{bmatrix} \frac{2}{x_2} & -\frac{2x_1}{x_2^2} \\ -\frac{2x_1}{x_2^2} & \frac{2x_1^2}{x_2^3} \end{bmatrix} = \frac{2}{x_2^3} \begin{bmatrix} x_2^2 & -x_1 x_2 \\ -x_1 x_2 & x_1^2 \end{bmatrix}$$

Verify that the Hessian is positive definite

Assume x2 is positive, then

For any vector

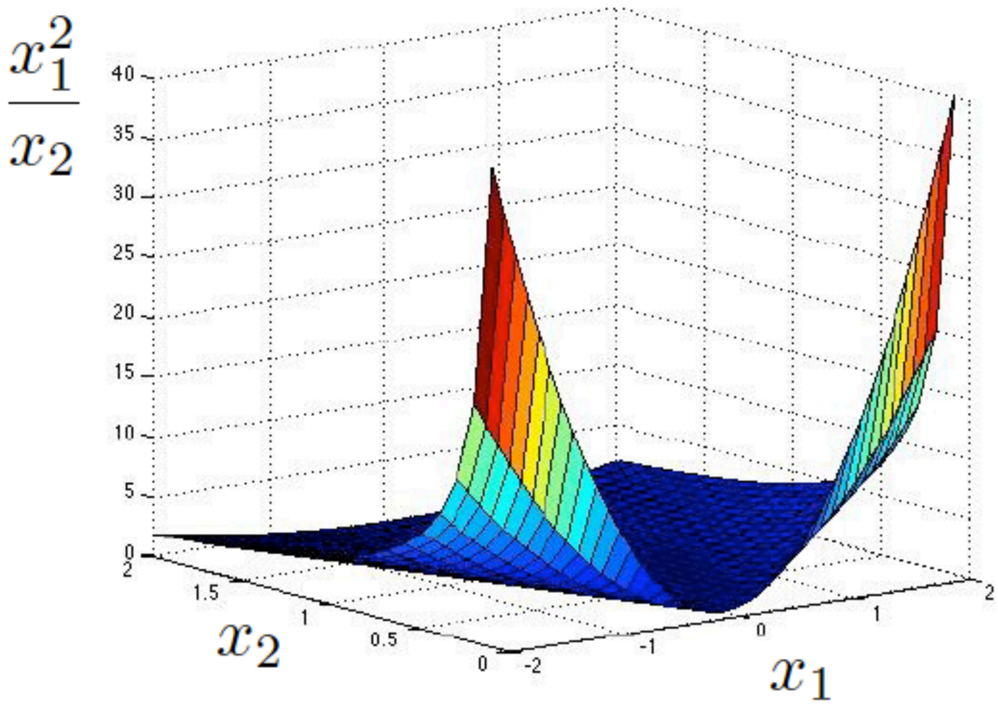
$$\boldsymbol{v} = \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\boldsymbol{v}^{\mathrm{T}} \boldsymbol{H} \boldsymbol{v} = \boldsymbol{v}^{\mathrm{T}} \frac{2}{x_2^3} \begin{bmatrix} x_2^2 & -x_1 x_2 \\ -x_1 x_2 & x_1^2 \end{bmatrix} \boldsymbol{v}$$

$$= \frac{2}{x_2^3} (a^2 x_2^2 - 2ab x_1 x_2 + b^2 x_1^2)$$

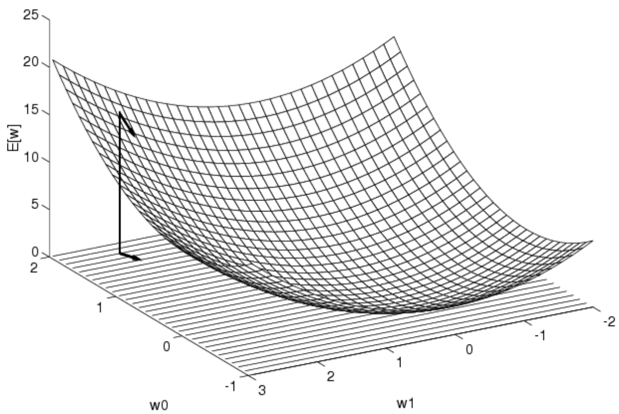
$$= \frac{2}{x_2^3} (a x_2 - b x_1)^2 \geq 0$$

What does this function look like?



Optimizing concave function – Gradient ascent

- Conditional likelihood for Logistic Regression is concave → Find optimum with gradient ascent



Gradient:

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

Learning rate, η>0

Update rule:

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

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \frac{\partial l(\mathbf{w})}{\partial w_i}$$

Maximize Conditional Log Likelihood: Gradient ascent

$$l(\mathbf{w}) \;=\; \sum_j y^j(w_0 + \sum_i^d w_i x_i^j) - \ln(1 + exp(w_0 + \sum_i^d w_i x_i^j))$$

Gradient Ascent for LR

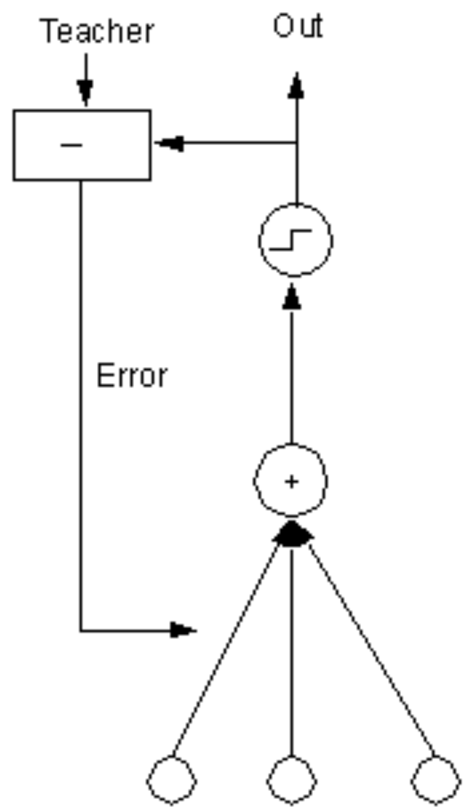
Gradient ascent algorithm: iterate until change < ε

$$w_0^{(t+1)} \leftarrow w_0^{(t)} + \eta \sum_j [y^j - \widehat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w})]$$

For i=1,…,n,

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \sum_j x_i^j [y^j - \widehat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w})]$$

repeat



Perceptron Learning

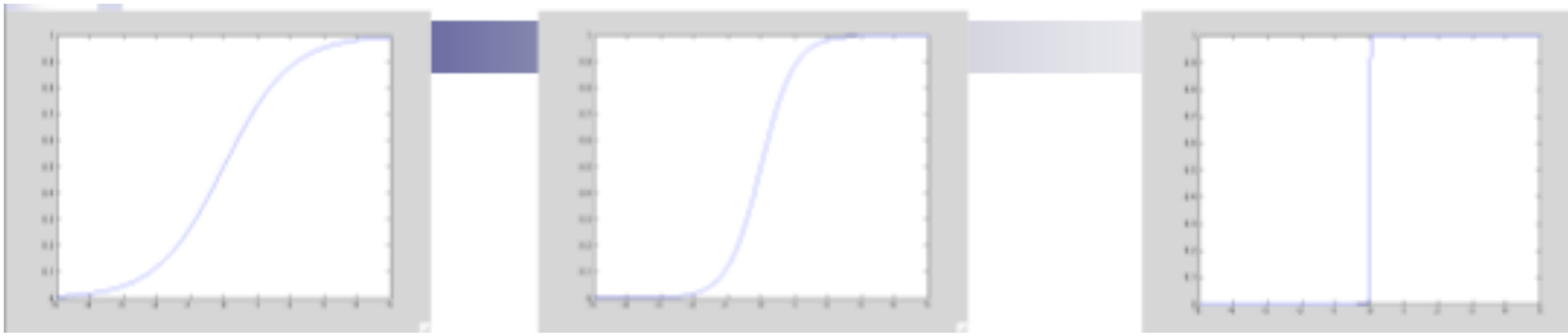
That's all M(C)LE. How about M(C)AP?

$$p(\mathbf{w} \mid Y, \mathbf{X}) \;\propto\; P(Y \mid \mathbf{X}, \mathbf{w})p(\mathbf{w})$$

- One common approach is to define priors on **w**
  - Normal distribution, zero mean, identity covariance
  - "Pushes" parameters towards zero
- Corresponds to ***Regularization***
  - Helps avoid very large weights and overfitting
  - More on this later in the semester
- MAP estimate

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \ln \left[p(\mathbf{w}) \prod_{j=1}^{N} P(y^j \mid \mathbf{x}^j, \mathbf{w}) \right]$$

Large parameters → Overfitting



$$\frac{1}{1+e^{-x}} \qquad \frac{1}{1+e^{-2x}} \qquad \frac{1}{1+e^{-100x}}$$

- If data is linearly separable, weights go to infinity
- Leads to overfitting

- Penalizing high weights can prevent overfitting

Gradient of M(C)AP

$$\frac{\partial}{\partial w_i} \ln \left[p(\mathbf{w}) \prod_{j=1}^{N} P(y^j \mid \mathbf{x}^j, \mathbf{w}) \right]$$

$$p(\mathbf{w}) = \prod_i \frac{1}{\kappa\sqrt{2\pi}} \; e^{\frac{-w_i^2}{2\kappa^2}}$$

MLE vs MAP

- Maximum conditional likelihood estimate

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \ln \left[\prod_{j=1}^{N} P(y^j \mid \mathbf{x}^j, \mathbf{w}) \right]$$

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w})]$$

- Maximum conditional a posteriori estimate

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \ln \left[p(\mathbf{w}) \prod_{j=1}^{N} P(y^j \mid \mathbf{x}^j, \mathbf{w}) \right]$$

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w})] \right\}$$

HW2 Tips

- Naïve Bayes
  - Train_NB
    - Implement “factor_tables” -- $|X_i|$ x $|Y|$ matrices
    - Prior $|Y|$ x 1 vector
    - Fill entries by counting + smoothing
  - Test_NB
    - argmax_y $P(Y=y)\ P(X_i=x_i)\ldots$
  - TIP: work in log domain

- Logistic Regression
  - Use small step-size at first
  - Make sure you maximize log-likelihood not minimize it
  - Sanity check: plot objective

Finishing up:
Connections between NB & LR

Logistic regression vs Naïve Bayes

- Consider learning f: X → Y, where
  - X is a vector of real-valued features, <X1 … Xd>
  - Y is boolean
- Gaussian Naïve Bayes classifier
  - assume all X_i are conditionally independent given Y
  - model $P(X_i \mid Y = k)$ as Gaussian $N(\mu_{ik},\sigma_i)$
  - model P(Y) as Bernoulli(θ,1-θ)
- What does that imply about the form of P(Y|X)?

$$P(Y=1 \mid \boldsymbol{X}=\boldsymbol{x}) = \frac{1}{1+exp(-w_0 - \sum_i w_i x_i)}$$

Cool!!!!

Derive form for P(Y|X) for continuous X_i

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

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

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

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

Ratio of class-conditional probabilities

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

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

Derive form for P(Y|X) for continuous X_i

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

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

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

$$P(Y=1 \mid \boldsymbol{X} = \boldsymbol{x}) = \frac{1}{1 + exp(-w_0 - \sum_i w_i x_i)}$$

Gaussian Naïve Bayes vs Logistic Regression

Set of Gaussian Naïve Bayes parameters (feature variance independent of class label)

Set of Logistic Regression parameters

Not necessarily

- Representation equivalence
  - **But only in a special case!!!** (GNB with class-independent variances)
- But what's the difference???
- **LR makes no assumptions about** P(**X**|Y) **in learning**!!!
- **Loss function!!!**
  - Optimize different functions → Obtain different solutions

Naïve Bayes vs Logistic Regression

Consider Y boolean, Xi continuous, X=<X1 ... Xd>

- Number of parameters:
  - NB: 4d +1 (or 3d+1)
  - LR: d+1
- Estimation method:
  - NB parameter estimates are uncoupled
  - LR parameter estimates are coupled

G. Naïve Bayes vs. Logistic Regression 1

[Ng & Jordan, 2002]

- Generative and Discriminative classifiers

- Asymptotic comparison
  (# training examples → infinity)
  - when model correct
    - GNB (with class independent variances) and LR produce identical classifiers
  - when model incorrect
    - LR is less biased – does not assume conditional independence
      - **therefore LR expected to outperform GNB**

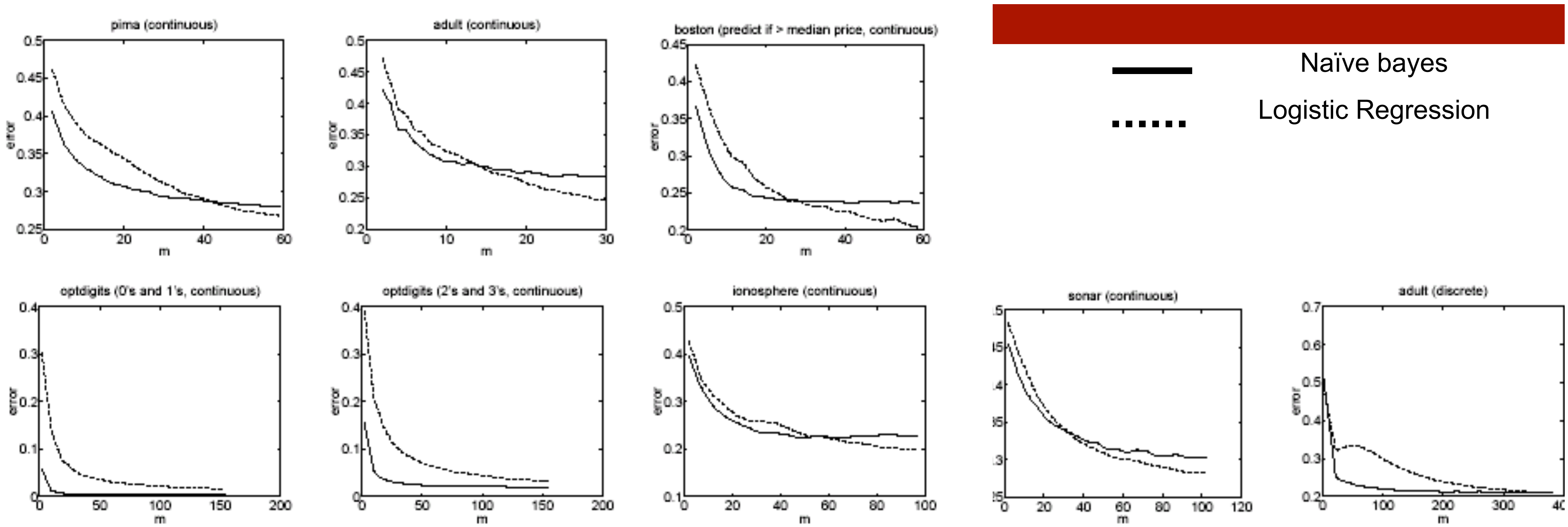
G. Naïve Bayes vs. Logistic Regression 2

[Ng & Jordan, 2002]

- Generative and Discriminative classifiers

- Non-asymptotic analysis
  - convergence rate of parameter estimates,
    d = # of attributes in X
    - Size of training data to get close to infinite data solution
    - GNB needs *O*(log d) samples
    - LR needs *O*(d) samples

  - **GNB converges more quickly to its (perhaps less helpful) asymptotic estimates**



Some experiments from UCI data sets

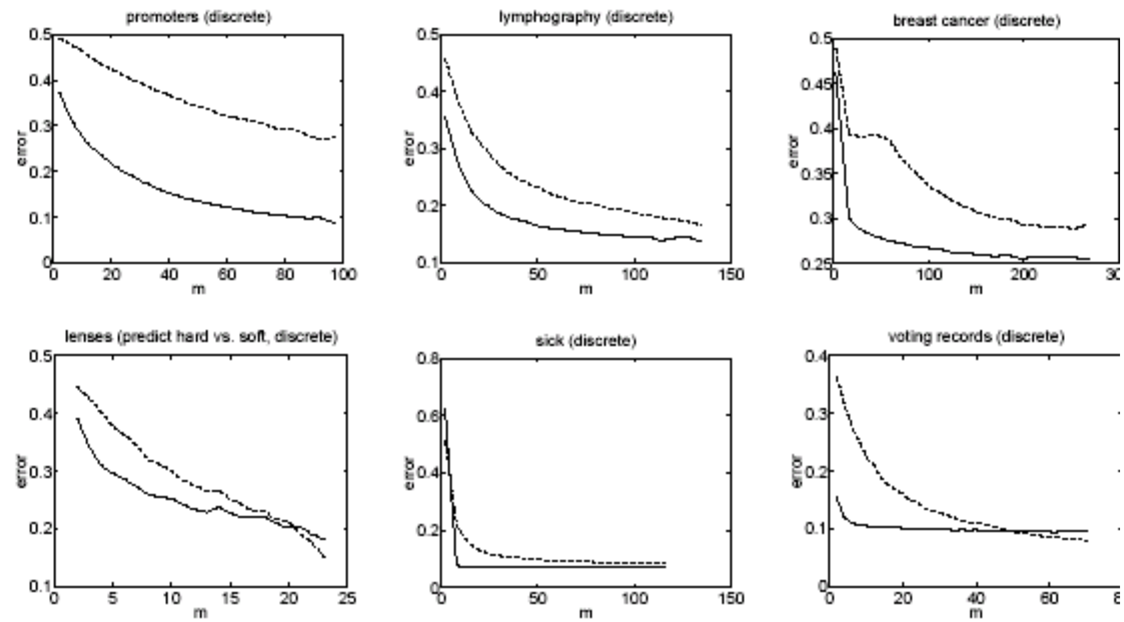


Figure 1: Results of 15 experiments on datasets from the UCI Machine Learning repository. Plots are of generalization error vs. m (averaged over 1000 random train/test splits). Dashed line is logistic regression; solid line is naive Bayes.

What you should know about LR

- 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