Logistic Regression

Vibhav Gogate
The University of Texas at Dallas

Some Slides from Carlos Guestrin, Luke Zettlemoyer and Dan Weld.

Generative *vs.* Discriminative Classifiers

- **Want to Learn**: h:**X** $\mapsto$ Y
  - **X** – features
  - Y – target classes
- **Generative classifier**, e.g., Naïve Bayes:
  - Assume some **functional form for P(X|Y), P(Y)**
  - Estimate parameters of P(X|Y), P(Y) directly from training data
  - Use Bayes rule to calculate P(Y|X= x)
  - This is a **'*generative*' model**
    - **Indirect** computation of P(Y|X) through Bayes rule
    - As a result, **can also generate a sample of the data**, $P(X) = \sum_y P(y)\, P(X|y)$

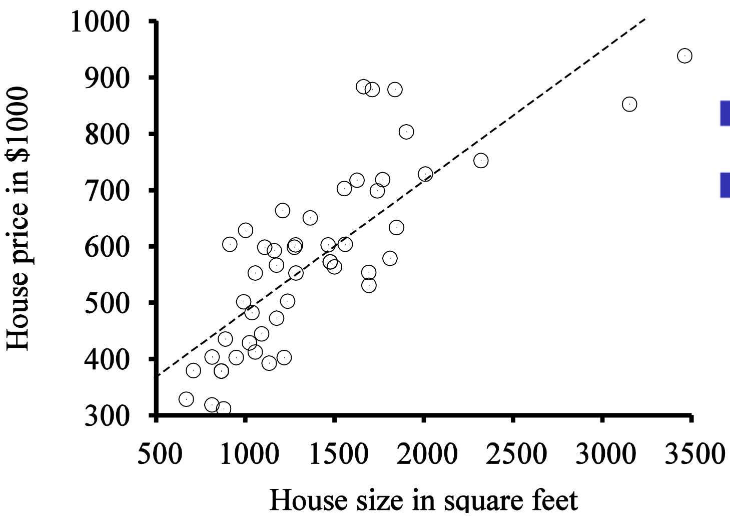
$$P(Y \mid \mathbf{X}) \propto P(\mathbf{X} \mid Y)\, P(Y)$$

- **Discriminative classifiers**, e.g., Logistic Regression:
  - Assume some **functional form for P(Y|X)**
  - Estimate parameters of P(Y|X) directly from training data
  - This is the **'*discriminative*' model**
    - Directly learn P(Y|X)
    - But **cannot obtain a sample of the data**, because P(X) is not available

Optimization

- Learning task: minimizing or maximizing an evaluation function $J(w_1, \ldots, w_n)$ given data $\mathcal{D}$
- $w_1, \ldots, w_n$ are the parameters that you need to tune.
- Simple example: Try to fit a line to the following data such that the error is minimized.
- Input: "x", desired output "y"

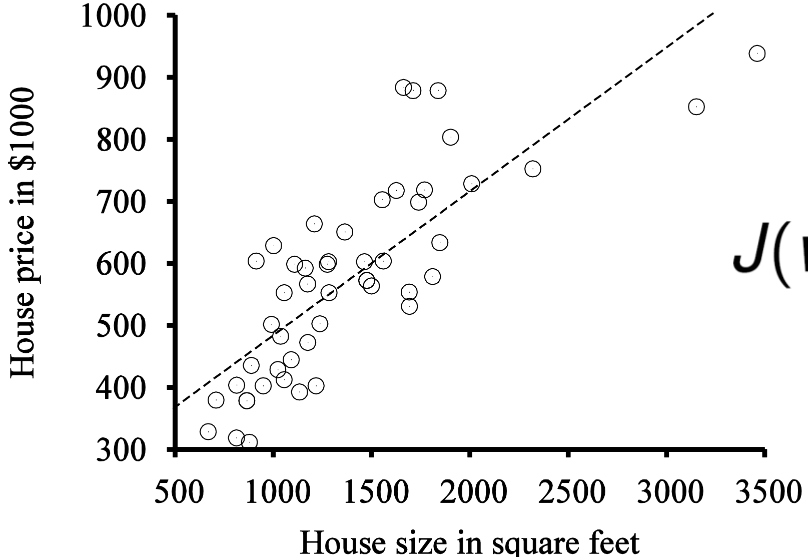
Linear Regression!

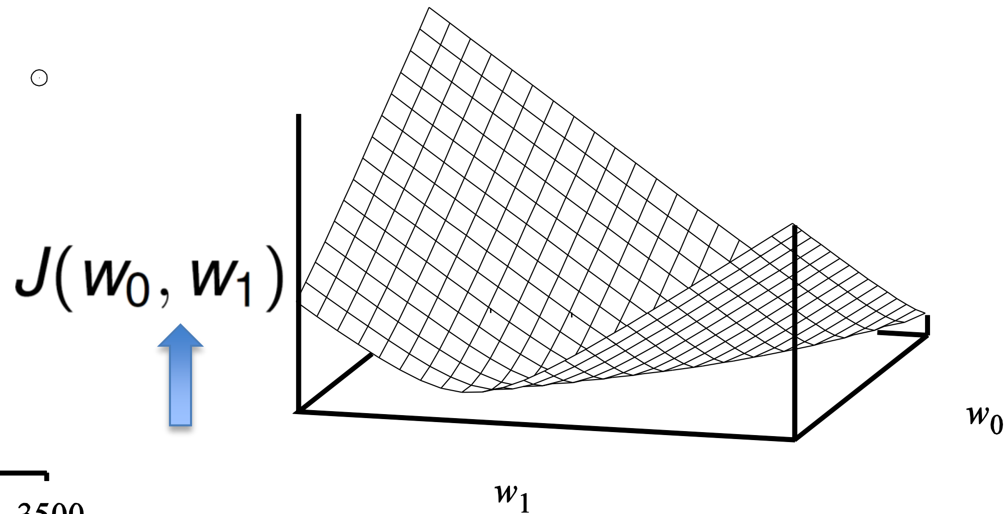


- Equation of line: $y = h(x) = w_0 + w_1 x$
- Error: $J(w_0, w_1) = \sum_{i=1}^{m} (y_i - (w_0 + w_1 x_i))^2$

(Point-wise) squared error
Problem: Minimize error

Question: How to solve the optimization problem



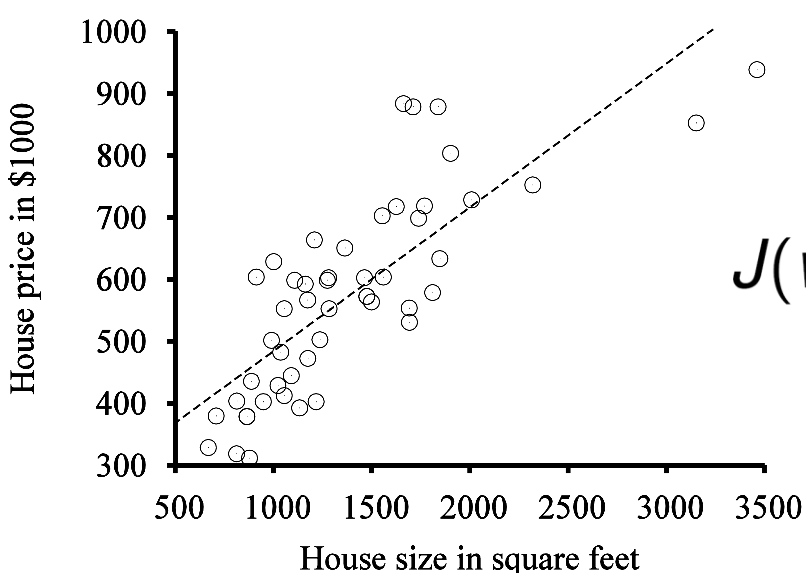


- Set the derivative of "J" to zero and solve

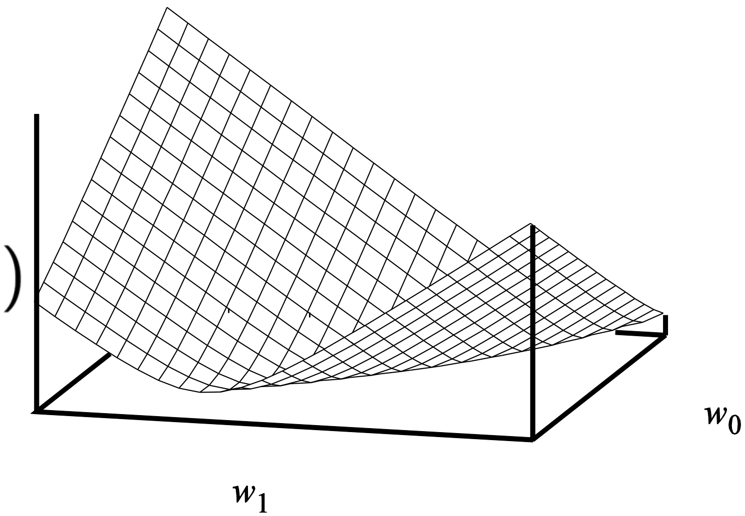
$$J(w_0, w_1) = \sum_{i=1}^{m}(y_i - (w_0 + w_1 x_i))^2$$

$$\frac{\partial}{\partial w_0} J(w_0, w_1) = 0 \qquad \frac{\partial}{\partial w_1} J(w_0, w_1) = 0$$

Question: How to solve the optimization problem



$J(w_0, w_1)$



$$w_1 = \frac{m \sum_{i=1}^{m} x_i y_i - \sum_{i=1}^{m} x_i \sum_{i=1}^{m} y_i}{m \sum_{i=1}^{m} x_i^2 - \left(\sum_{i=1}^{m} x_i\right)^2}$$

$$w_0 = \frac{\sum_{i=1}^{m} y_i - w_1 \sum_{i=1}^{m} x_i}{m}$$

Homework: Prove this! (Messy; algebraic manipulation)

Multivariate Linear Regression

- Input: $\mathbf{x}$ is a vector; desired output y.

Assuming a dummy attribute $x_0=1$ **for all examples**

$$y = h(\mathbf{x}) = w_0 + \sum_{j=1}^{n} w_j x_j = \sum_{j=0}^{n} w_j x_j = \mathbf{w}^T \mathbf{x}$$

Inner product or dot product (yields a number)

$$J(\mathbf{w}) = \sum_{i=1}^{m} (y_i - \mathbf{w}^T x_i)^2$$

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \cdots \\ \cdots \\ \mathbf{x}_m^T \end{bmatrix}$$

X is a m-by-n matrix

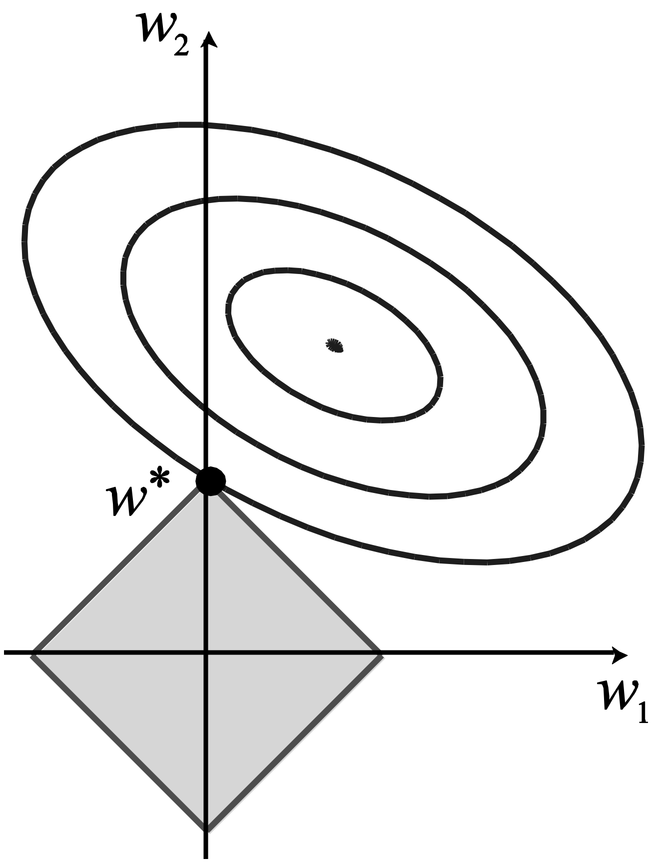
Overfitting

- MLE estimate: Some weights are large because of chance (coincidental regularities)
- Regularize!!
  - Penalize high weights (complex hypothesis)
  - Minimize cost: Loss + Complexity

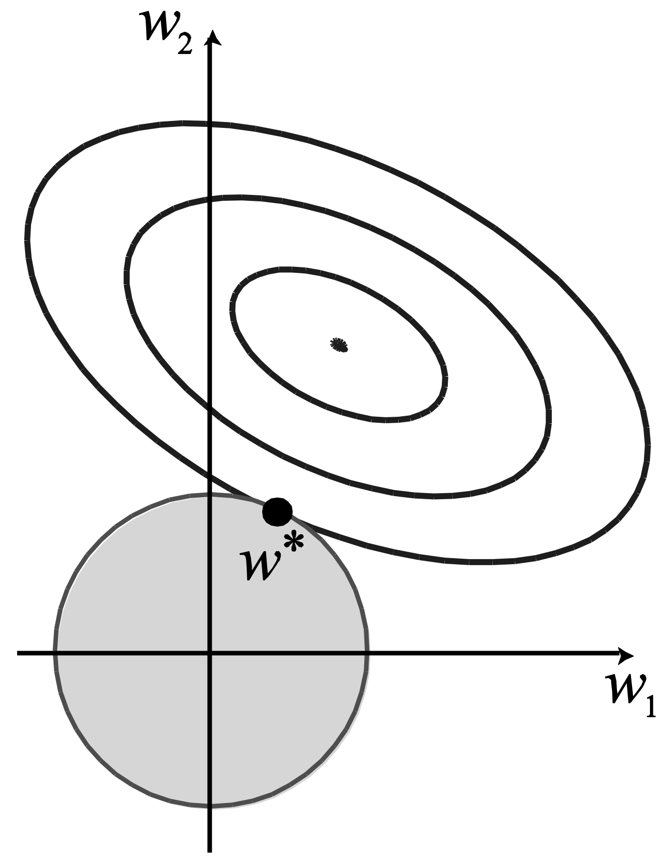
$$JR(\mathbf{w}) = \sum_{i=1}^{m} \left(y_i - \sum_{j=1}^{n} w_j x_{i,j} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{m} |w_j|^q$$

p=1: L1 regularization (Lasso)
p=2: L2 regularization

Regularization



L1



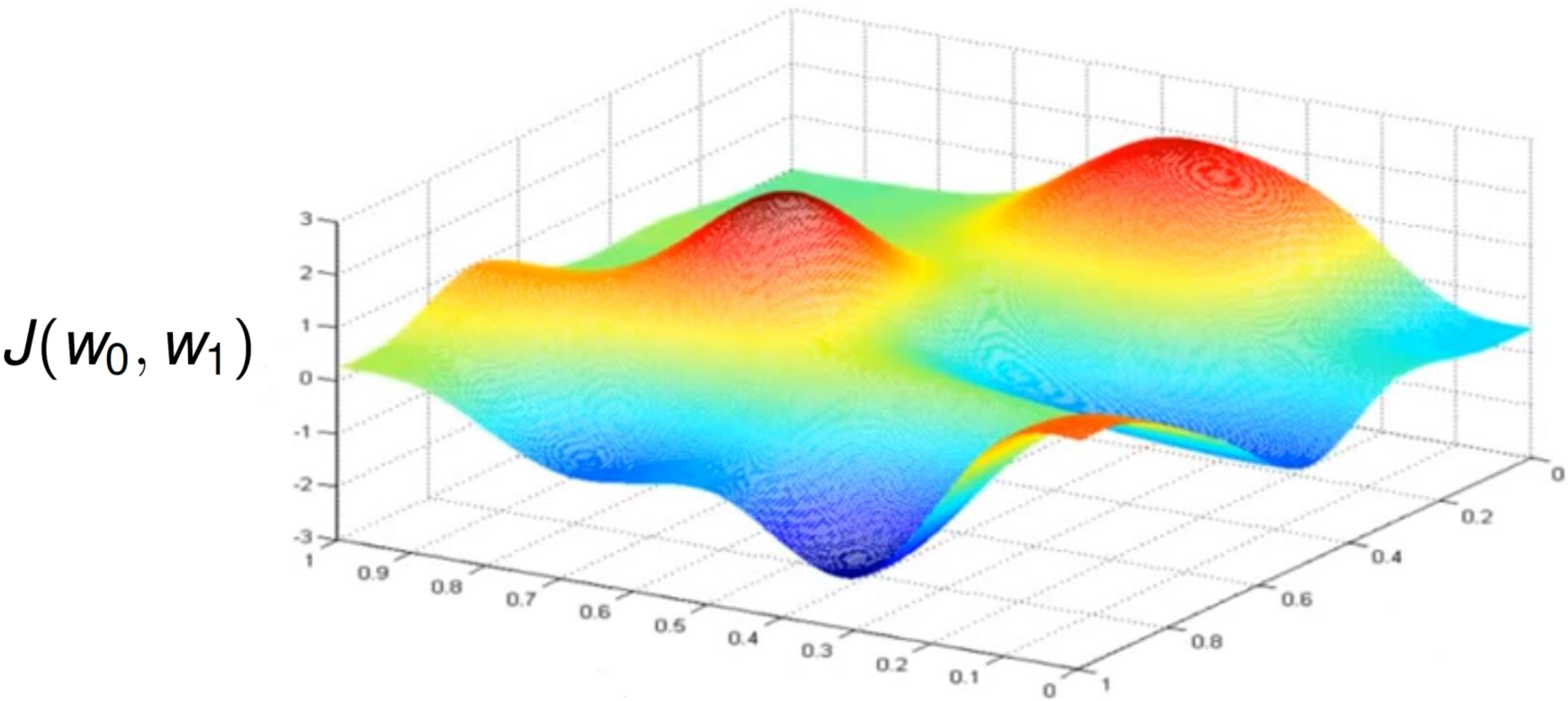
L2

Gradient Descent

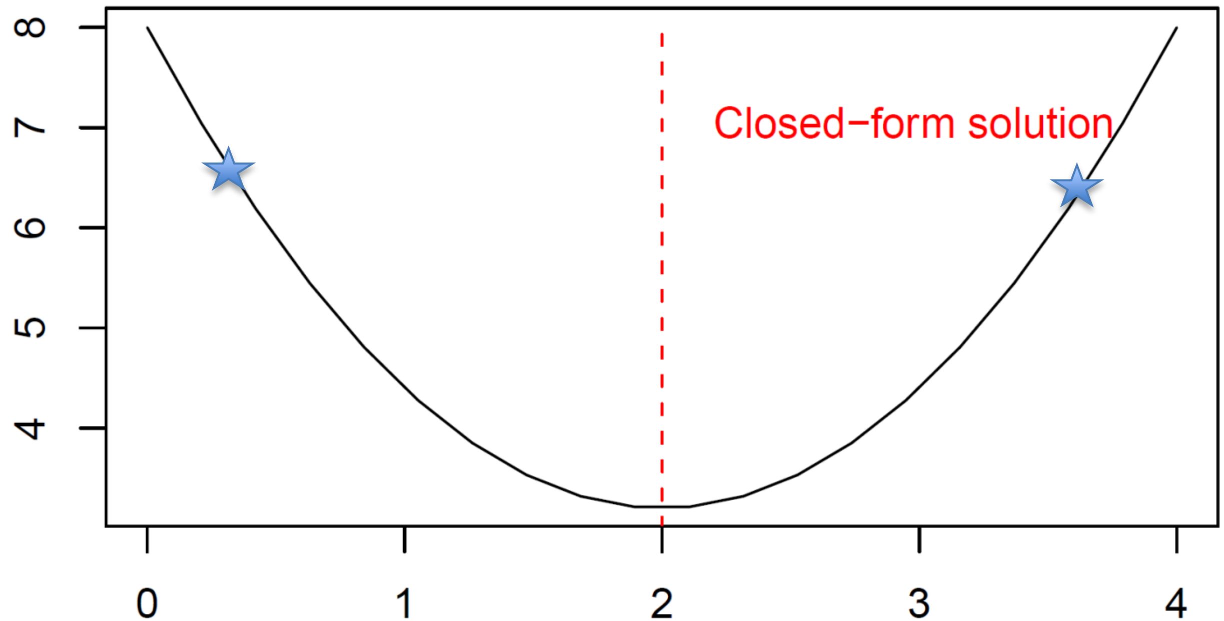
- Closed form solution is not always possible.
- In that case, we can use the following iterative approach.
- **Algorithm Gradient Descent**
  - $\mathbf{w}$ = Any point in the weight space
  - Loop Until Convergence
    - Simultaneously update each w_j in $\mathbf{w}$ as follows:
      - $w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\mathbf{w})$

Learning rate (arrow pointing to α)

Gradient Descent: Example



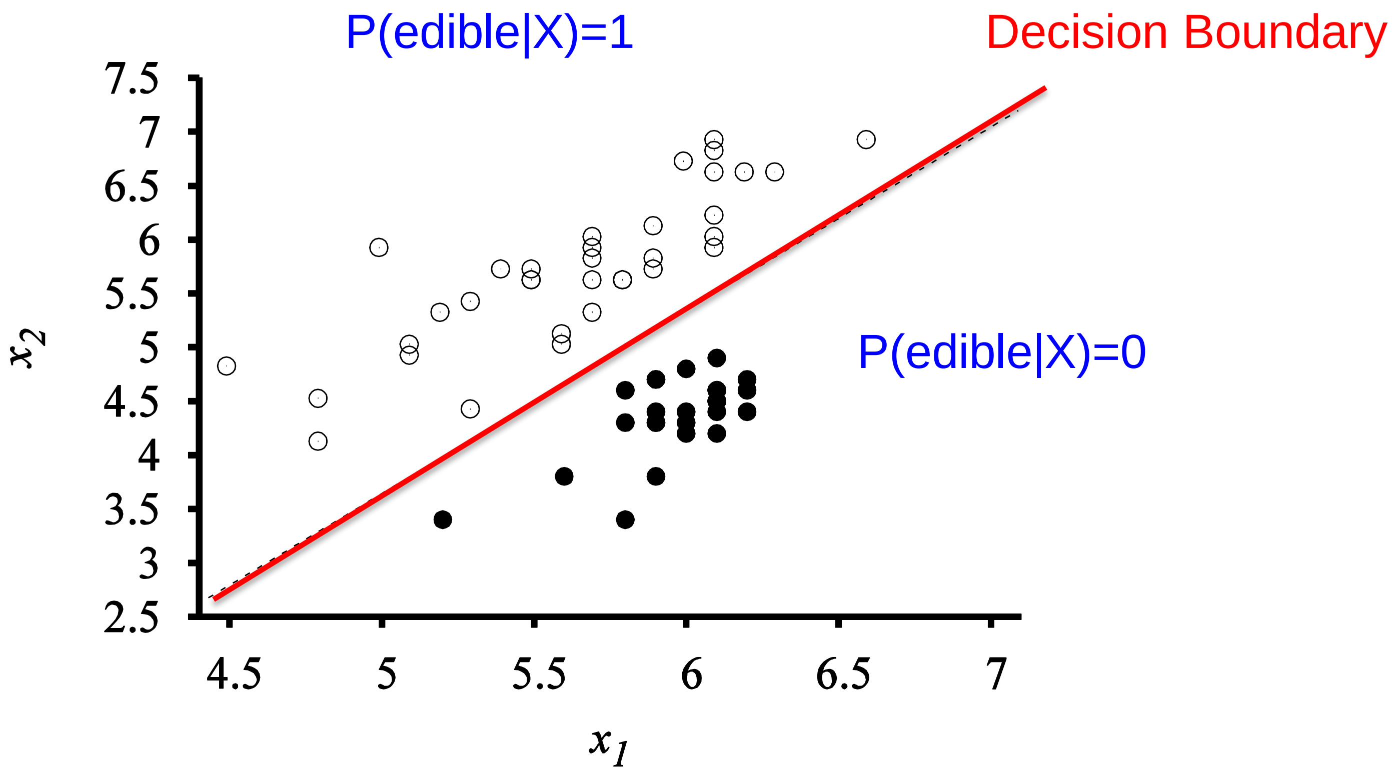
Gradient Descent: 1-D



- **Remember:** Derivative is the slope of the line that is tangent to the function
- **Question:** What if the learning rate is small? (Slow convergence)
- **Question:** What if the learning rate is large? (Fail to converge; even diverge)

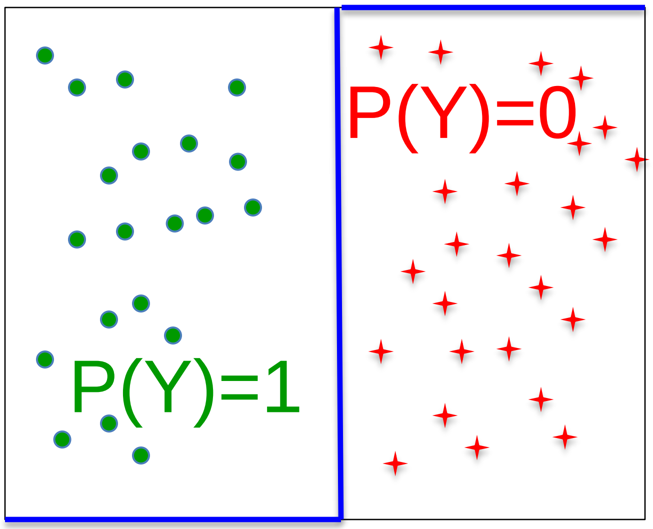
Rule: $$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\mathbf{w})$$

Back to Classification



Logistic Regression

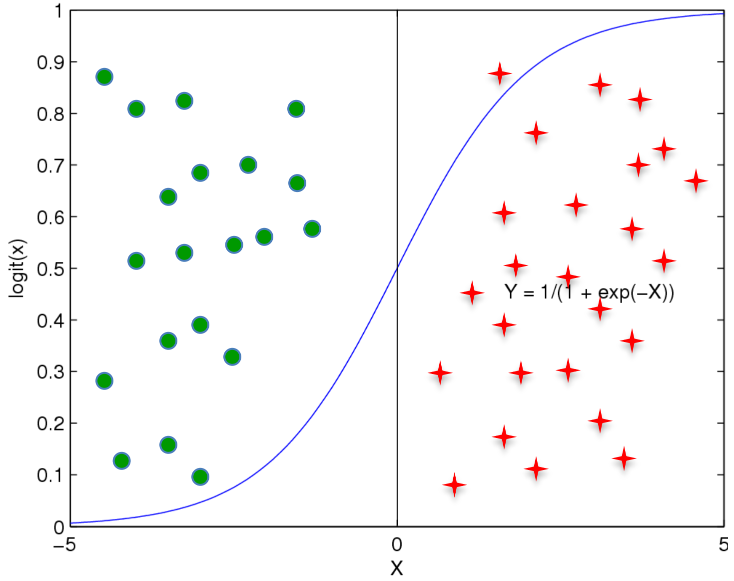
- Learn P(Y|**X**) directly!
  - Assume a particular functional form
  - ☹ ***Not differentiable…***



Logistic Regression

- Learn P(Y|**X**) directly!
  - Assume a particular functional form
  - Logistic Function
    - Aka Sigmoid

$$\frac{1}{1+exp(-z)}$$



Functional Form: Two classes

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

implies

$$P(Y=0|X) = \frac{\exp(w_0+\sum_{i=1}^n w_i X_i)}{1+\exp(w_0+\sum_{i=1}^n w_i X_i)}$$

Classification Rule: Assign the label Y=0 if

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

Take logs and simplify:

$$0 < w_0 + \sum_{i=1}^n w_i X_i$$

linear classification rule!

Y=0 if the RHS>0

How to learn the weights?

- Evaluation function: Maximize the conditional log-likelihood

l indexes the examples

$$W \leftarrow \arg\max_{W} \prod_{l} P(Y^l | X^l, W)$$

$$W = \langle w_0, w_1 \ldots w_n \rangle$$ Weight vector

- **Note that actually we are just computing P(Y|X)**
- **W is included in P(Y|X) just to show that the probability is computed using W**

How to Learn the weights?

$$W \leftarrow \arg\max_{W} \sum_{l} \ln P(Y^l | X^l, W)$$

How to Learn the weights?

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

$$l(W) = \sum_l Y^l \ln P(Y^l = 1 | X^l, W) + (1 - Y^l) \ln P(Y^l = 0 | X^l, W)$$

Why?

If the domain of a variable Y is {0,1}
Then any function f(Y) can be written as:
f(Y) = Yf(Y=1)+(1-Y)f(Y=0)

How to learn the weights?

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

Remember

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

$$P(Y = 0|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^n w_i X_i)}$$

Log of this = -ln(denominator)

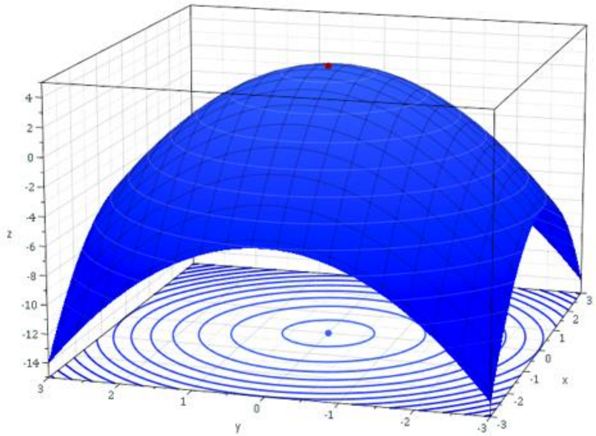
How to Learn the weights?

Bad news: no closed-form solution to maximize $l(W)$

Good news: $l(W)$ is concave function of W!

No local minima

Concave functions easy to optimize using Gradient Ascent



Update w_i as follows:
$w_i = w_i + (\text{learning rate}) * (\text{partial derivate of } l(W) \text{ w.r.t. } w_i)$

Notice that unlike gradient descent, in gradient ascent we are interested in the maximim value and therefore we have a "+" sign on the RHS of the update rule instead of "-" sign.

How to learn the weights?

$$l(W) = \sum_l Y^l (w_0 + \sum_i^n w_i X_i^l) - \ln(1 + \exp(w_0 + \sum_i^n w_i X_i^l))$$

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

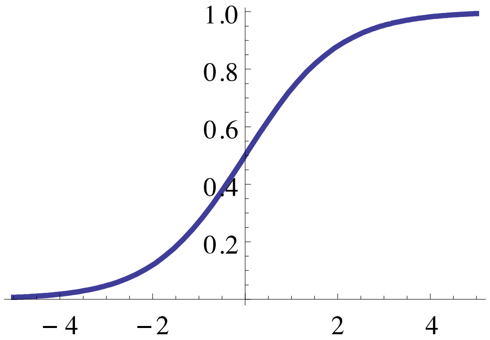
The term inside the parenthesis is the prediction error (difference between the observed value and the predicted probability)

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

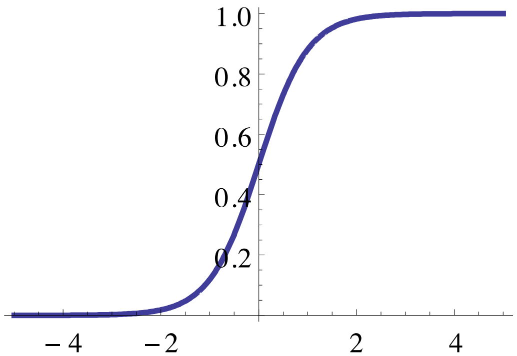
Learning rate

Large parameters…

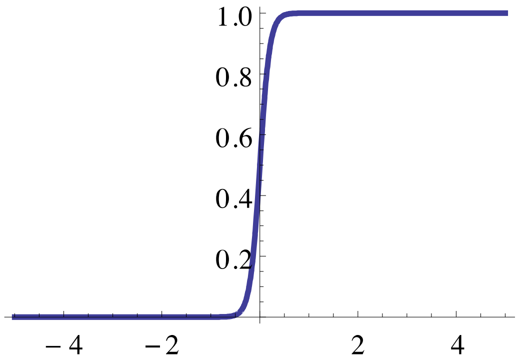
$$\frac{1}{1+e^{-ax}}$$



a=1



a=5



a=10

- Maximum likelihood solution: prefers higher weights
  - higher likelihood of (properly classified) examples close to decision boundary
  - larger influence of corresponding features on decision
  - *can cause overfitting!!!*
- Regularization: penalize high weights

That's all MCLE. How about MCAP?

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

- One common approach is to define priors on W
  - Normal distribution, zero mean, identity covariance
  - "Pushes" parameters towards zero
- ***Regularization***
  - Helps avoid very large weights and overfitting
- MAP estimate:

$$W \leftarrow \arg\max_W \sum_l \ln P(Y^l | X^l, W) - \frac{\lambda}{2} ||W||^2$$

MCAP as Regularization

$$W \leftarrow \arg\max_{W} \sum_{l} \ln P(Y^l|X^l, W) - \frac{\lambda}{2}||W||^2$$

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

- Weight update rule:

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

  - Quadratic penalty: drives weights towards zero
  - Adds a negative linear term to the gradients

Penalizes high weights, like we did in linear regression

Naïve Bayes *vs.* Logistic Regression

Learning: h:**X** $\mapsto$ Y

X – features

Y – target classes

Generative

- Assume functional form for
  - P(X|Y) **assume cond indep**
  - P(Y)
  - Est params from train data
- Gaussian NB for cont features
- Bayes rule to calc. P(Y|X= x)
  - P(Y | **X**) $\propto$ P(**X** | Y) P(Y)
- **Indirect** computation
  - Can also generate a sample of the data

Discriminative

- Assume functional form for
  - P(Y|X) **no assumptions**
  - Est params from training data
- Handles discrete & cont features
- Directly calculate P(Y|X=x)
  - Can't generate data sample

Remember Gaussian Naïve Bayes?

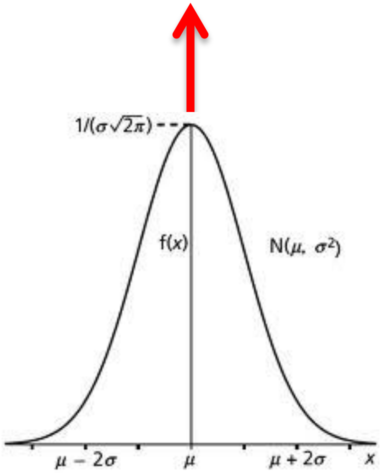
Sometimes Assume Variance

- is independent of Y (i.e., σ_i),
- or independent of X_i (i.e., σ_k)
- or both (i.e., σ)

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

$$P(X_i = x \mid Y = y_k) = N(\mu_{ik}, \sigma_{ik})$$

$$N(\mu_{ik}, \sigma_{ik}) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} \; e^{\frac{-(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$



Gaussian Naïve Bayes *vs.* Logistic Regression

Learning: h:**X** $\mapsto$ Y

X – ***Real-valued*** features

Y – target classes

Generative

- Assume functional form for
  - P(X|Y) assume X_i cond indep given Y
  - P(Y)
  - Est params from train data
- Gaussian NB for continuous features
  - model $P(X_i \mid Y = y_k)$ as **Gaussian** $N(\mu_{ik},\sigma_i)$
  - model P(Y) as **Bernoulli** $(\pi,1-\pi)$
- Bayes rule to calc. P(Y|X= x)
  - $P(Y \mid \mathbf{X}) \propto P(\mathbf{X} \mid Y)\, P(Y)$

What can we say about the form of $P(Y=1 \mid \ldots X_i \ldots)$?

$$\frac{1}{1 + exp(w_0 + \sum_i w_i X_i)}$$

Cool!!!!

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

$$\frac{1}{1 + exp(w_0 + \sum_i w_i X_i)}$$

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

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

up to now, all arithmetic

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

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

only for Naïve Bayes models

Looks like a setting for w_0?

Can we solve for w_i ?

- Yes, but only in Gaussian case

Ratio of class-conditional probabilities

$$\frac{1}{1 + exp(w_0 + \sum_i w_i X_i)}$$

$$\begin{aligned}
\sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)} &= \sum_i \ln \frac{\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(\frac{-(X_i-\mu_{i0})^2}{2\sigma_i^2}\right)}{\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(\frac{-(X_i-\mu_{i1})^2}{2\sigma_i^2}\right)} \\
&= \sum_i \ln \exp\left(\frac{(X_i-\mu_{i1})^2-(X_i-\mu_{i0})^2}{2\sigma_i^2}\right) \\
&= \sum_i \left(\frac{(X_i-\mu_{i1})^2-(X_i-\mu_{i0})^2}{2\sigma_i^2}\right) \\
&= \sum_i \left(\frac{(X_i^2-2X_i\mu_{i1}+\mu_{i1}^2)-(X_i^2-2X_i\mu_{i0}+\mu_{i0}^2)}{2\sigma_i^2}\right) \\
&= \sum_i \left(\frac{2X_i(\mu_{i0}-\mu_{i1})+\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}\right) \\
&= \sum_i \left(\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}X_i + \frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}\right)
\end{aligned}$$

Linear function! Coefficients expressed with original Gaussian parameters!

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

$$P(Y=1|X) = \frac{1}{1+\exp(\ln\frac{1-\pi}{\pi} + \sum_i \left(\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}X_i + \frac{\mu_{i1}^2-\mu_{i0}^2)}{2\sigma_i^2}\right))}$$

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

Just like Logistic Regression!!!

$$w_i = \frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2} \qquad w_0 = \ln\frac{1-\pi}{\pi} + \sum_i \frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2}$$

Gaussian Naïve Bayes vs. Logistic Regression

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

Can go both ways, we only did one way

Set of Logistic Regression parameters

- **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**!!!

---- Optimize different functions ! Obtain different solutions

Naïve Bayes vs. Logistic Regression

[Ng & Jordan, 2002]

- Generative vs. Discriminative classifiers
- Asymptotic comparison

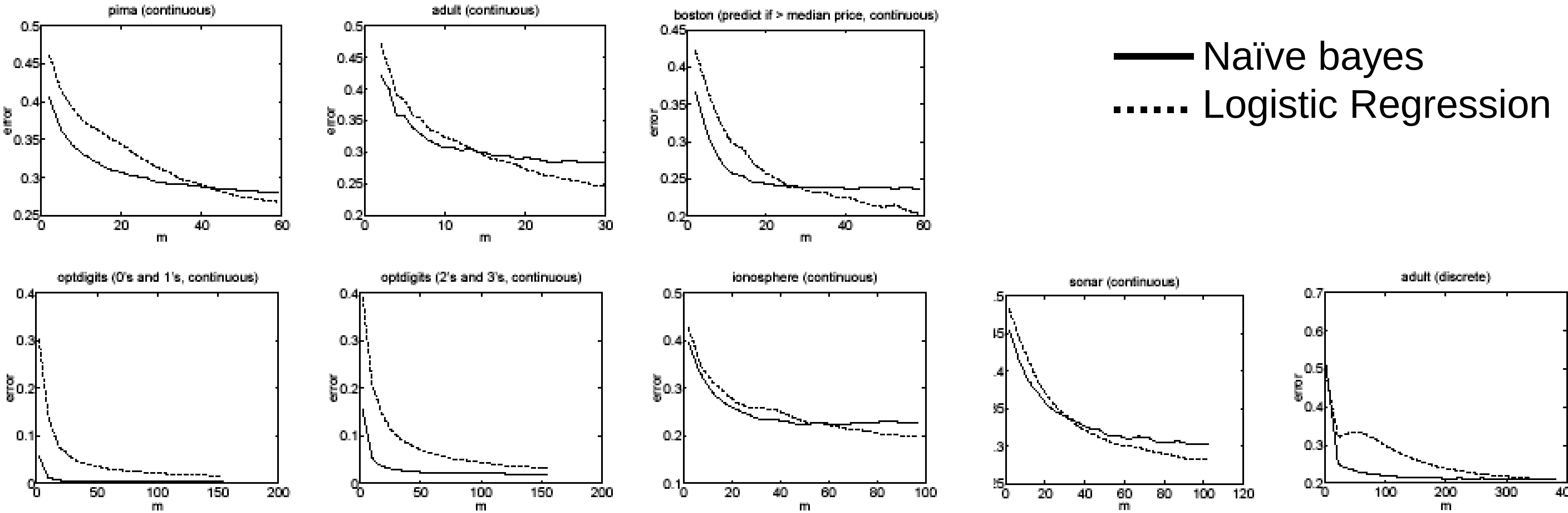
  (# training examples → infinity)

  - 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**

Naïve Bayes vs. Logistic Regression

[Ng & Jordan, 2002]

- Generative vs. Discriminative classifiers
- Non-asymptotic analysis
  - convergence rate of parameter estimates,
    (n = # of attributes in X)
    - Size of training data to get close to infinite data solution
    - Naïve Bayes needs *O*(log n) samples
    - Logistic Regression needs *O*(n) samples
  - GNB converges more quickly to its ***(perhaps less helpful)*** asymptotic estimates



Some experiments from UCI data sets

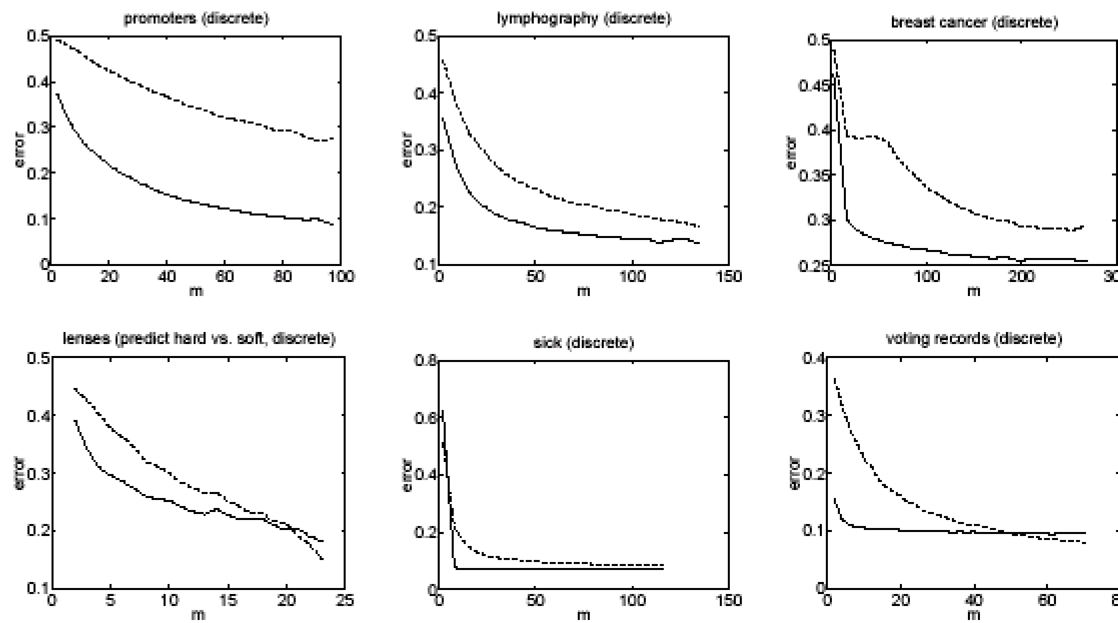


Figure 1: Results of 15 experiments on datasets from the UCI Machine Learnin
repository. Plots are of generalization error vs. m (averaged over 1000 rando
train/test splits). Dashed line is logistic regression; solid line is naive Bayes.

What you should know about Logistic Regression (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