Introduction to Machine Learning: CS 436/580L

Logistic Regression

Instructor: Arti Ramesh

Binghamton University

Administrivia

- Big quiz
- HW 2 is out! Due Oct 13th, 11:59 pm.
  - Naïve bayes
    - SPAM/HAM dataset
  - Point estimation
  - Submission instructions on myCourses

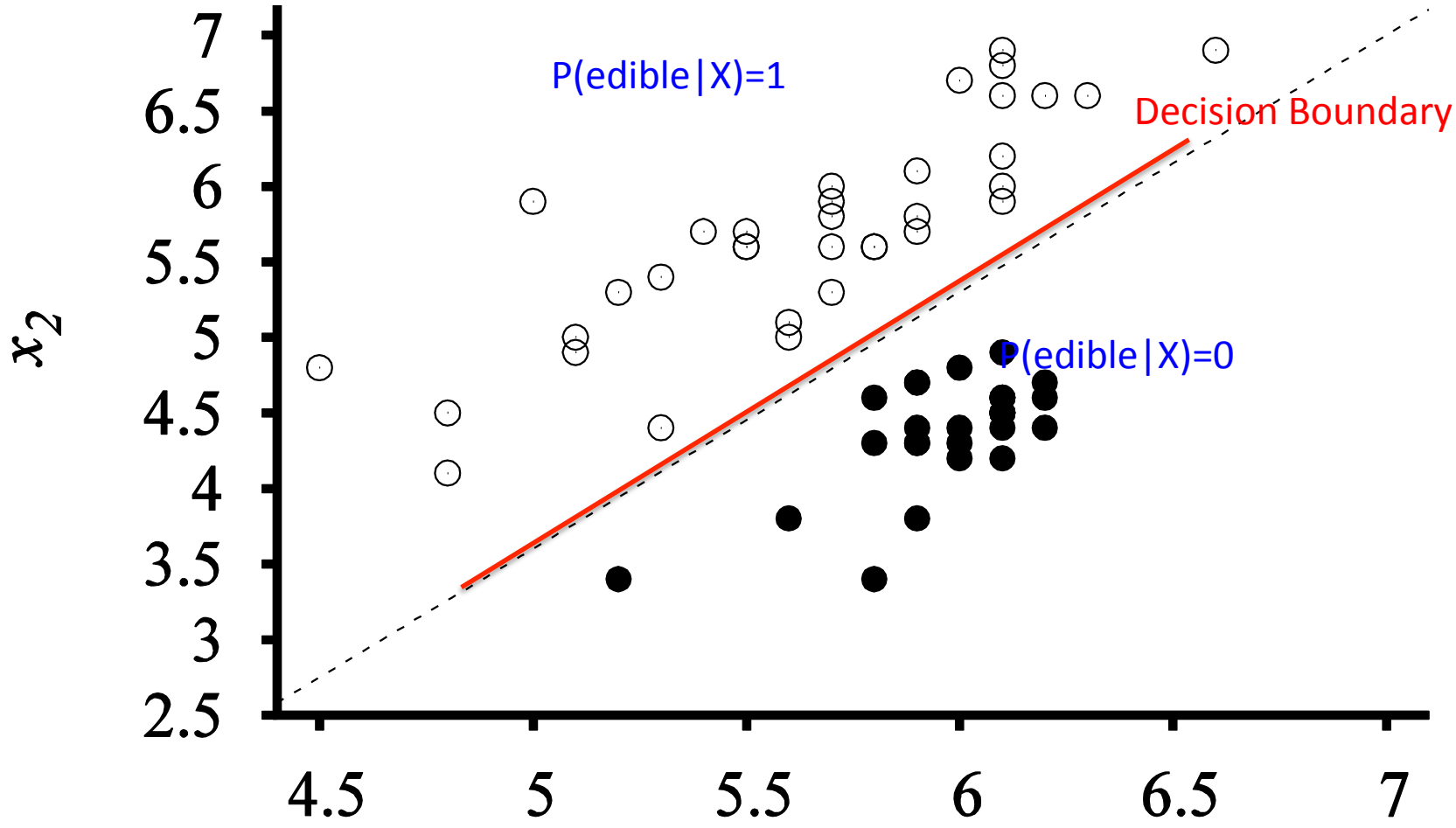
Generative *vs.* Discriminative Classifiers

- **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**, $P(X) = \sum_y P(y)\, P(X|y)$
- **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

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

3

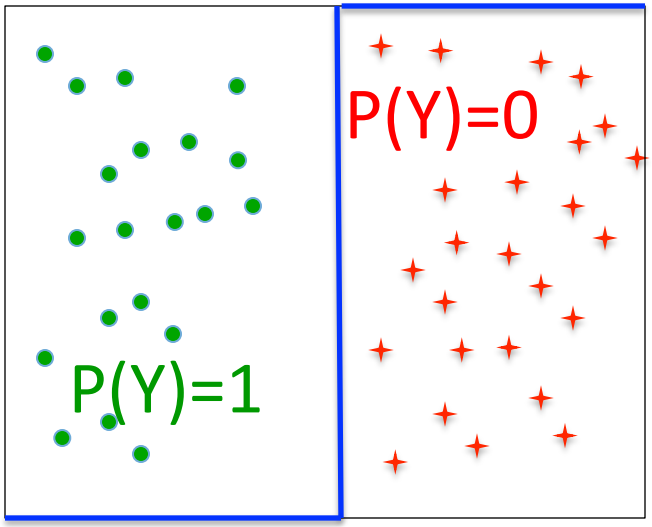
Classification



4

Logistic Regression

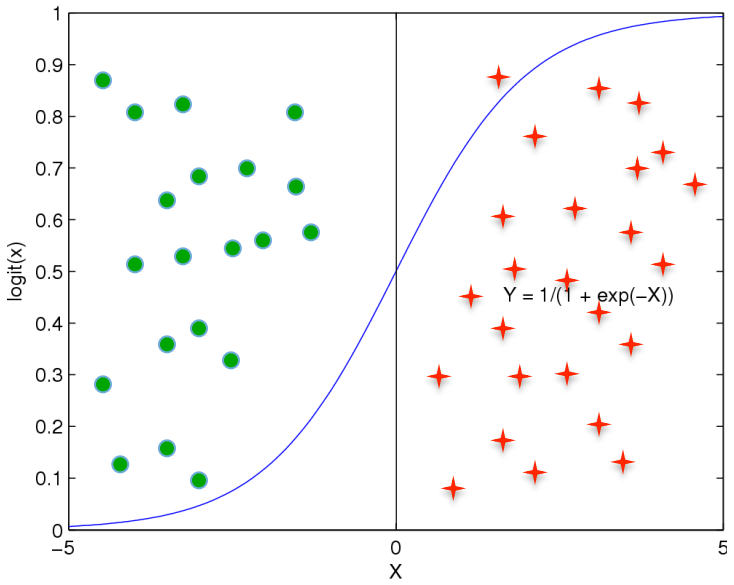
- Learn P(Y|**X**) directly!
  - Assume a particular functional form
  - ***Not differentiable…***



Logistic Regression

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

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

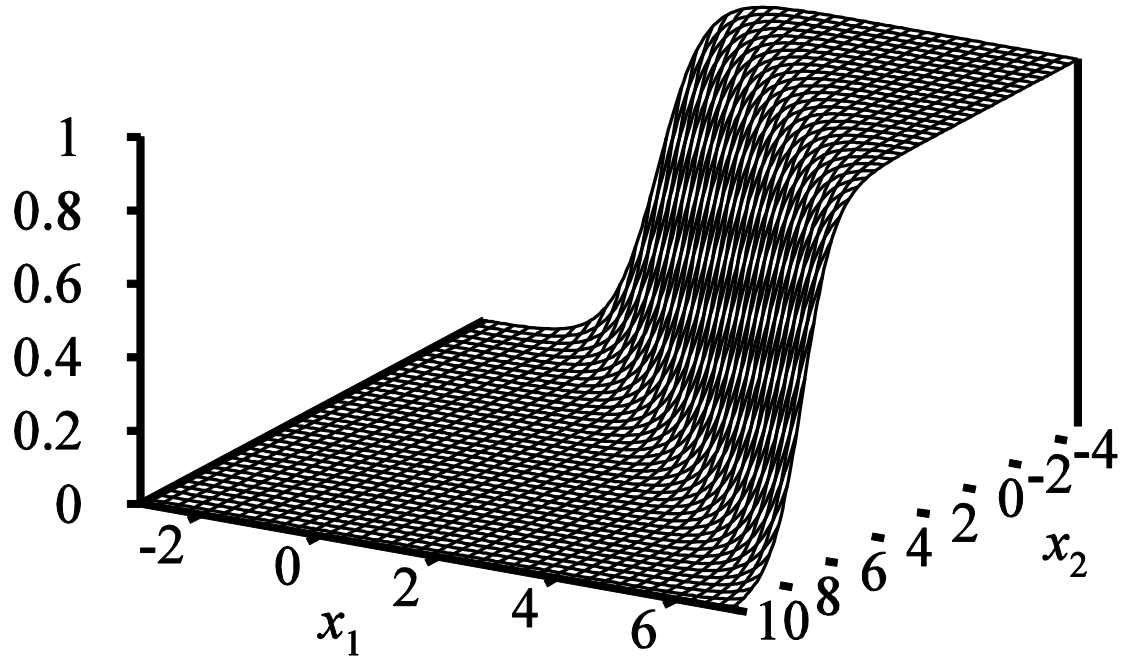


6

Logistic Function in n Dimensions

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

Sigmoid applied to a linear function of the data:



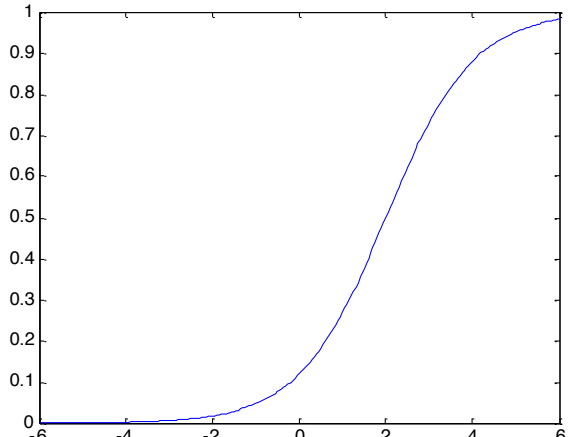
Features can be discrete or continuous!

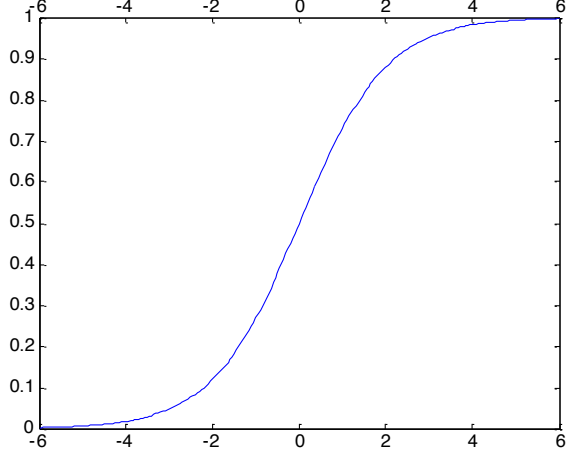
7

Understanding Sigmoids

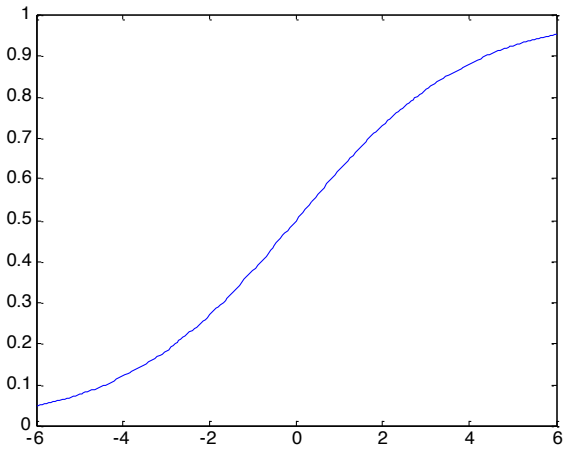
$$g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + e^{w_0 + \sum_i w_i x_i}}$$

w_0=2, w_1=-1





w_0=0, w_1=-1



w_0=0, w_1= -0.5

Functional Form: Two classes

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

implies

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

Classification Rule: Assign the label Y=1 if

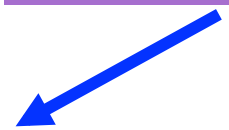
$$P(Y=1|X)/P(Y=0|X) > 1$$

linear classification rule!

Classify as Y=1 if

Take logs and simplify:

$$w_0 + \sum_{i=1}^{n} w_i x_i > 0$$



9

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**

10

How to Learn the weights?

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

11

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)

12

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)

13

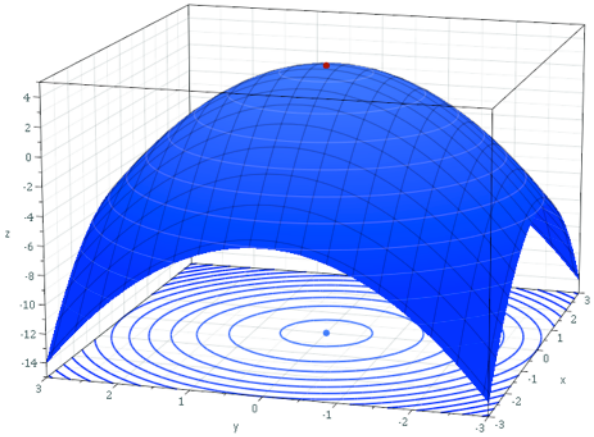
How to Learn the weights?

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:

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

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

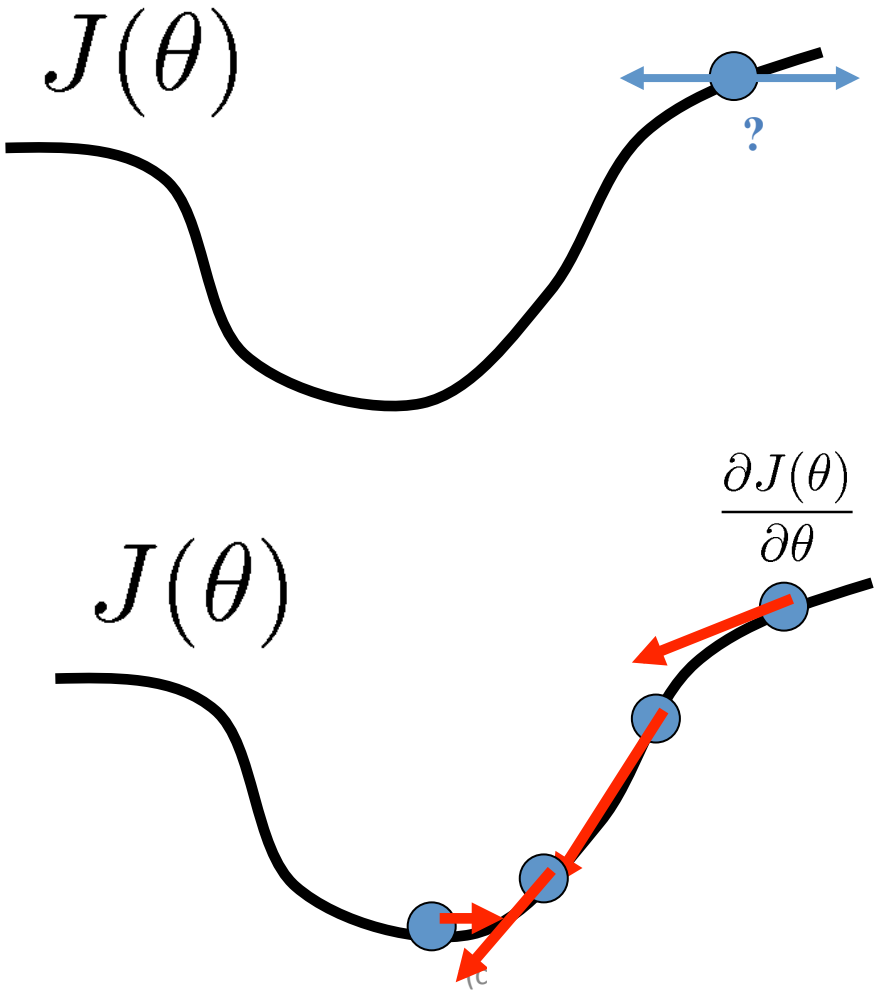
15

Gradient Descent

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

Learning rate

Gradient Descent

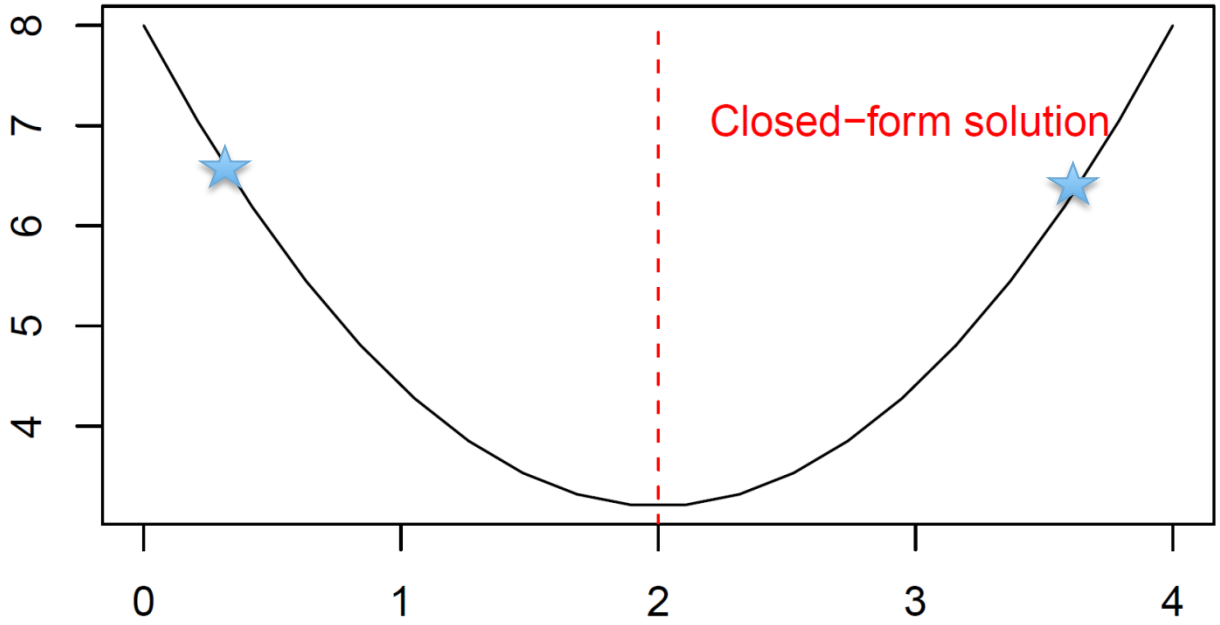


How to change θ to improve J(θ)?

Choose a direction in which J(θ) is decreasing

Derivative $\frac{\partial J(\theta)}{\partial \theta}$

Gradient Descent: 1-D



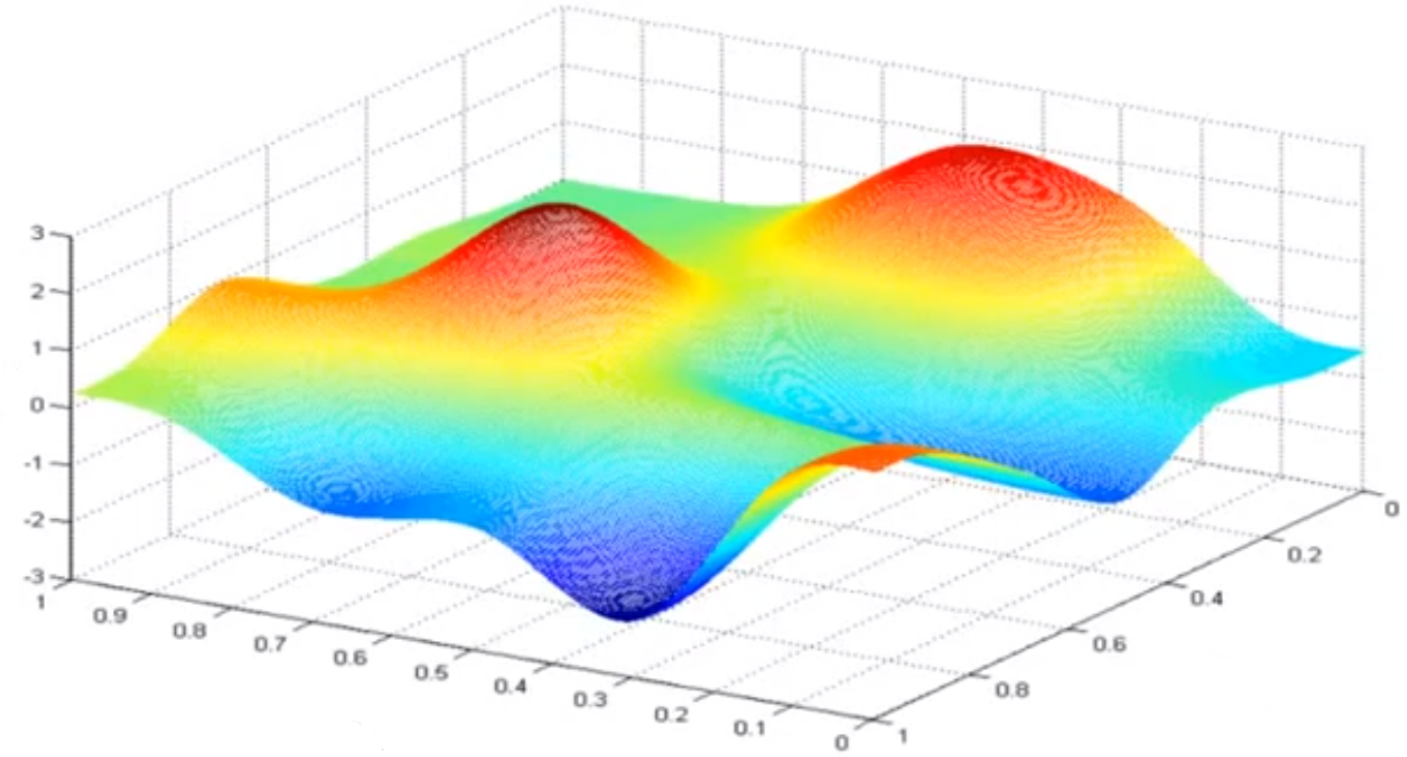
- **Remember:** Derivative is the slope of the line that is tangent to the function
- **Question:** What if the learning rate is small? (Slow convergence)
- **Question:** What if the learning rate is large? (Fail to converge; even diverge)

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

18

Gradient Descent: Example

$J(w_0, w_1)$

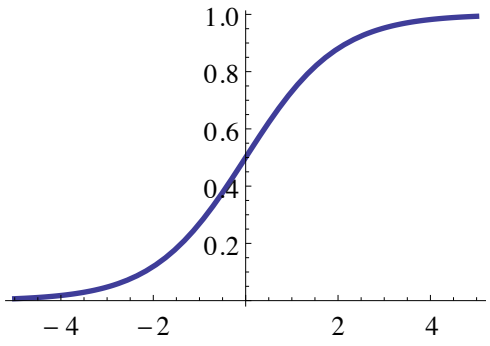


19

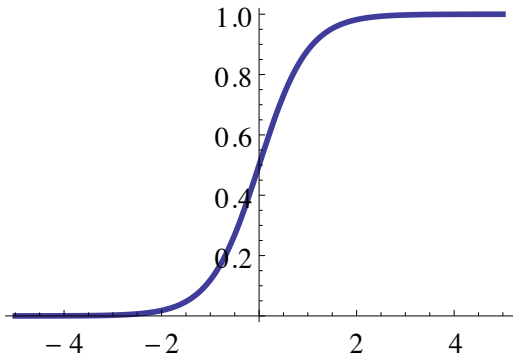
Large parameters…

- 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

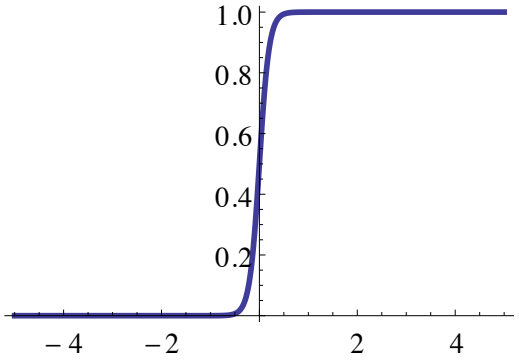
$$\frac{1}{1+e^{-ax}}$$



a=1



a=5

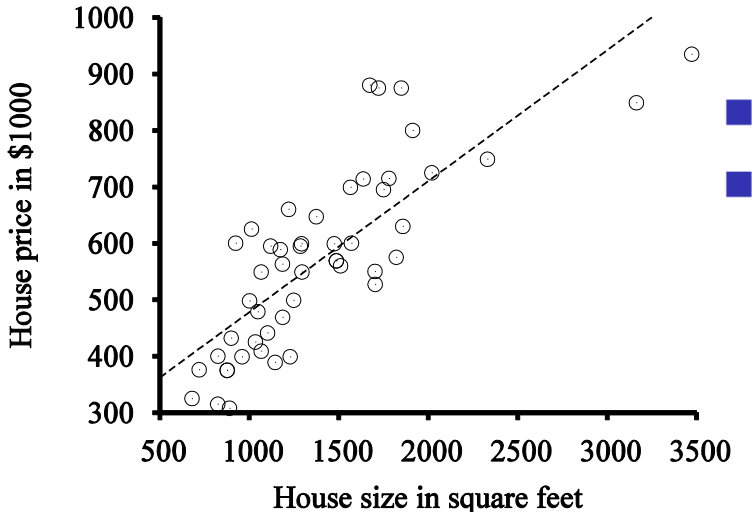


a=10

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!

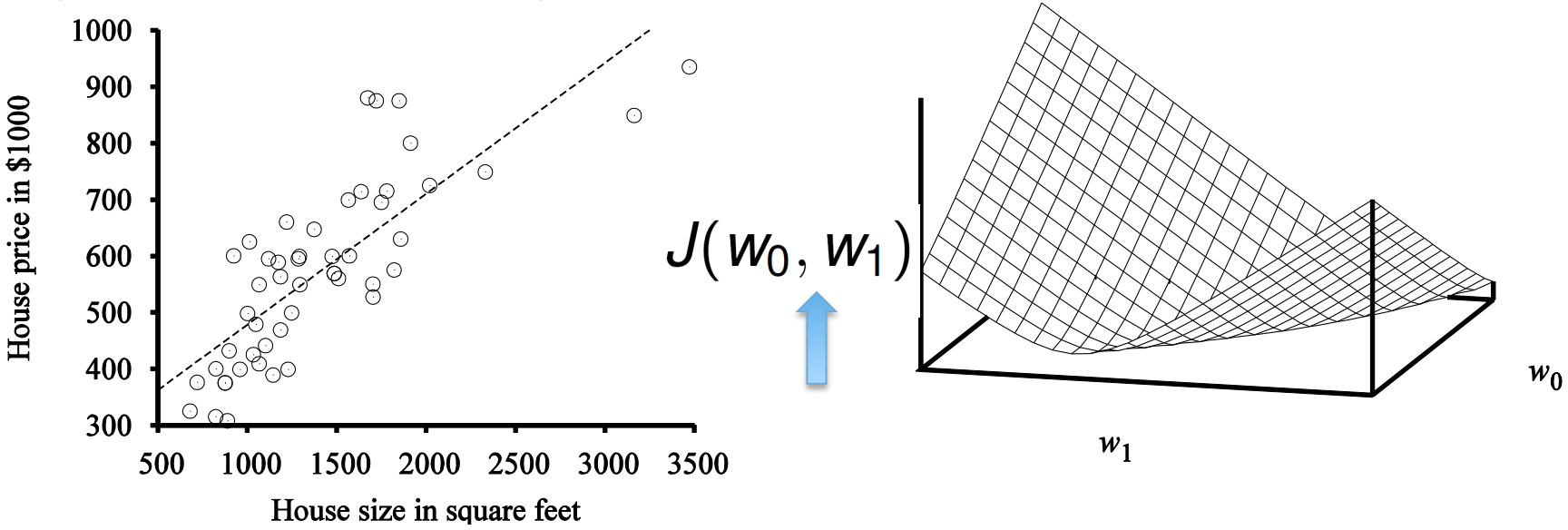


- Equation of line: $y = h(x) = w_0 + w_1 x$
- Error: $J(w_0, w_1) = \sum_{i=1}^{m} (y_i - (w_0 + w_1 x_i))^2$

(Point-wise) squared error
Problem: Minimize error

21

Question: How to solve the optimization problem



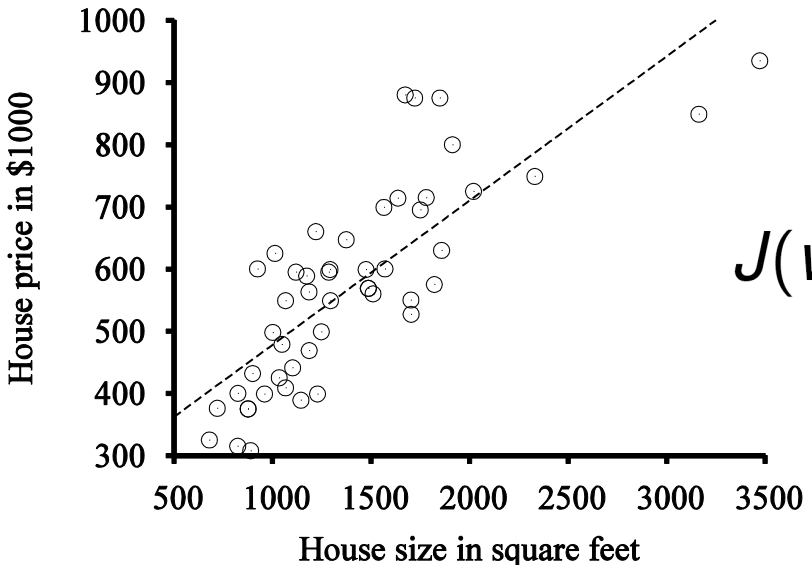
- Set the derivative of "J" to zero and solve

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

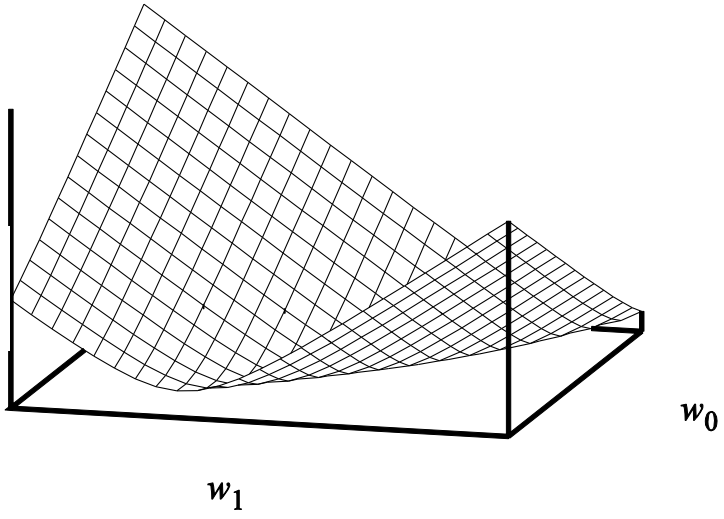
$$\frac{\partial}{\partial w_0} J(w_0, w_1) = 0 \qquad \frac{\partial}{\partial w_1} J(w_0, w_1) = 0$$

22

Question: How to solve the optimization problem



$J(w_0, w_1)$



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

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

Homework:
Prove this!
(Messy; algebraic manipulation)

23

Multivariate Linear Regression

- Input: **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

24

Overfitting

- MLE estimate: Some weights are large because of chance (coincidental regularities)
- Regularization
  - 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 (Ridge)

25

L2 Regularization

- Reduces complexity by adding a complexity penalty to the loss function
- L_2 Regularization: Complexity = sum of squares of weights

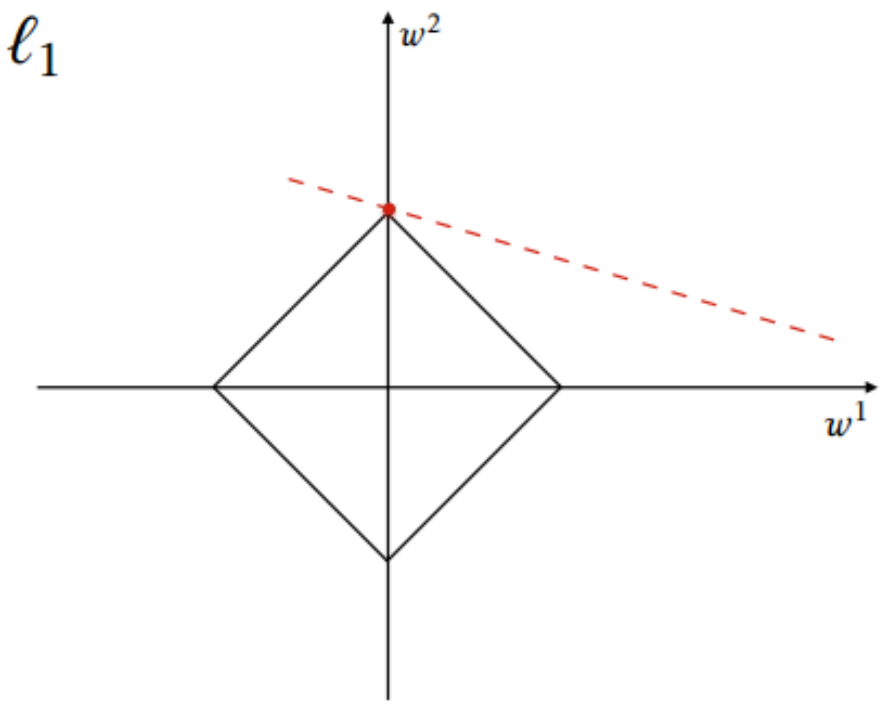
```
double sumSquaredVals = 0.0; // L2 penalty
for (int i = 0; i < weights.Length; ++i)
    sumSquaredVals += (weights[i] * weights[i]);
```

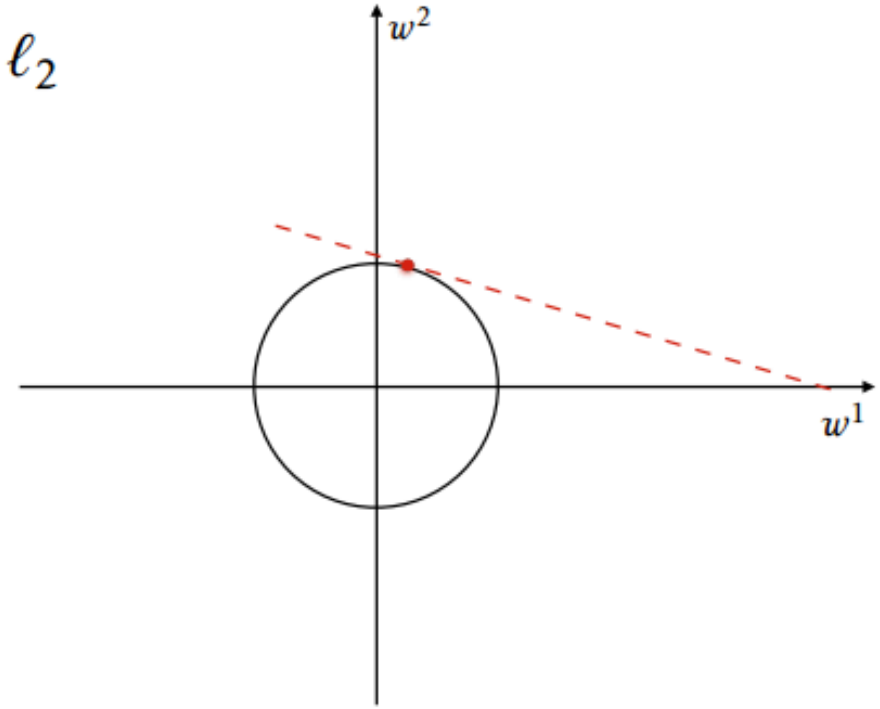
L1 Regularization

- Reduces complexity by adding a complexity penalty to the loss function
- L_1 Regularization: Complexity = absolute value of weights

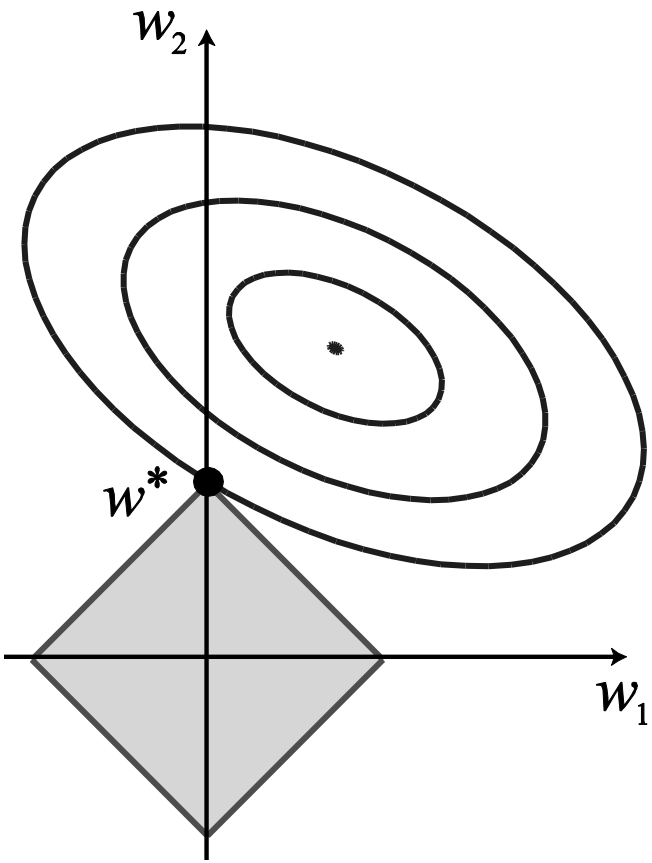
```
double sumAbsVals = 0.0; // L1 penalty
for (int i = 0; i < weights.Length; ++i)
        sumAbsVals += Math.Abs(weights[i]);
```

Regularization: L1 vs L2

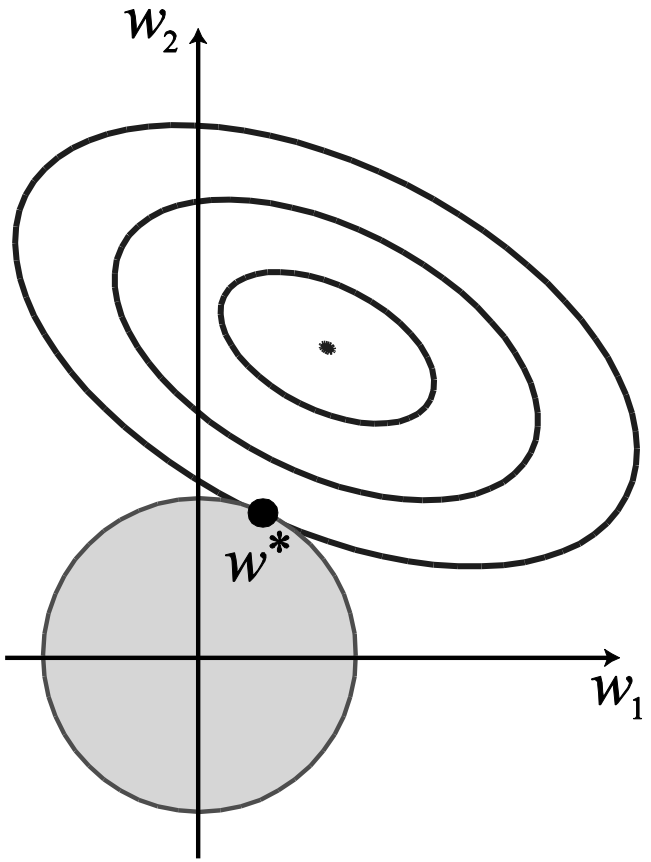




Regularization



L1

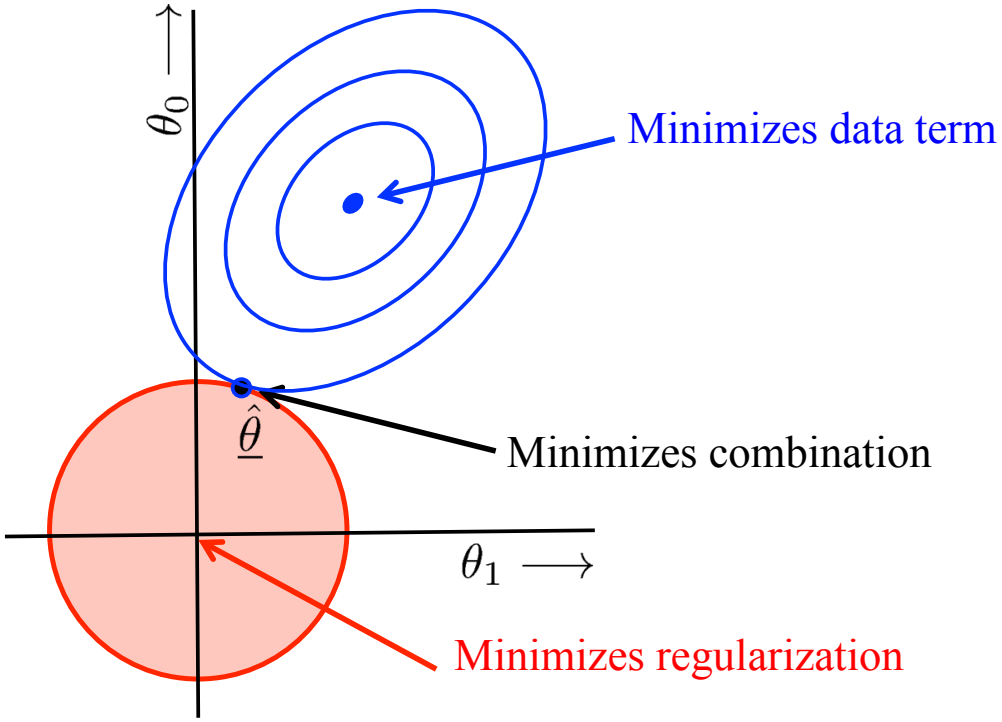


L2

29

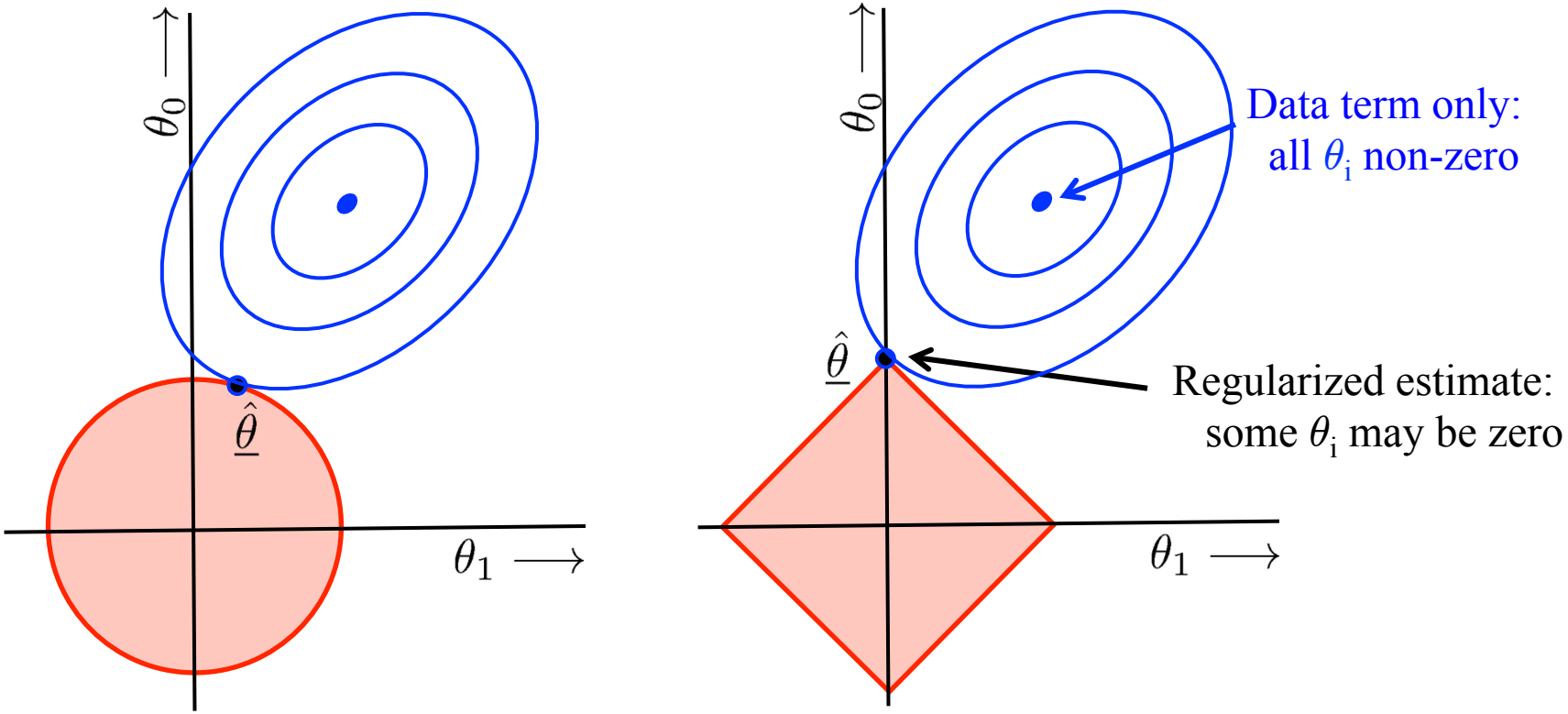
Regularization: L1 vs L2

- Estimate balances data term and regularization term



Regularization: L1 vs L2

- L1 tends to generate sparser solutions than a quadratic regularizer



Which Regularization is better?

- L1 regularization can sometimes have a beneficial side effect of driving one or more weight values to 0.0, which effectively means the associated feature isn't needed.
- Downside of using L1 regularization is that the technique can't be easily used with some ML training algorithms, in particular those algorithms that use calculus to compute a gradient

That's all MCLE. How about MCAP?

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

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!

MCLE vs. MCAP

- 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

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

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

Administrivia

- HW1 grades available tomorrow
  - DO NOT FORGET PLAGIARISM PLEDGE!!!
  - ASSIGNMENT WILL NOT BE GRADED
  - 1 week to dispute grades
  - Grade disputes: Email TA.
- HW2 Questions?
- Quiz in class on Tuesday, Feb 21st
  - Will account towards a percentage of HW3
  - Quiz will be online, please bring your laptops

Bayesian Categorization

$P(y_1 \mid X) \sim P(y_i)P(X \mid y_i)$

- Need to know:
  - Priors: $P(y_i)$
  - Conditionals: $P(\mathbf{X} \mid y_i)$
- $P(y_i)$ are easily estimated from data.
  - If n_i of the examples in D are in y_i, then $P(y_i) = n_i / |D|$
- Conditionals:
  - $\mathbf{X} = X_1 \wedge \ldots \wedge X_n$
  - Estimate $P(X_1 \wedge \ldots \wedge X_n \mid y_i)$
- Too many possible instances to estimate!
  - *(exponential in n)*
  - Even ***with*** bag of words assumption!

Problem!

37

Naïve Bayes

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

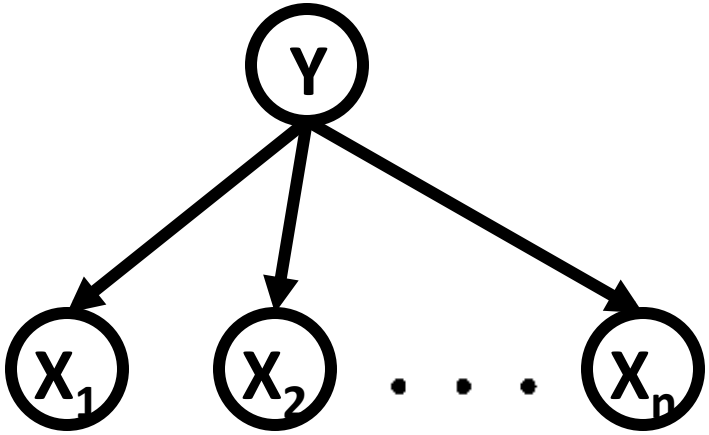
$$P(X_1, X_2|Y) = P(X_1|X_2, Y)P(X_2|Y)$$
$$= P(X_1|Y)P(X_2|Y)$$

  - More generally:

$$P(X_1 \ldots X_n|Y) = \prod_i P(X_i|Y)$$

The Naïve Bayes Classifier

- **Given:**
  - Prior
  - *n* conditionally independent features given the class
  - For each, we have likelihood



Decision rule:

$$y^* = h_{NB}(\mathbf{x}) = \arg\max_y P(y)P(x_1, \ldots, x_n \mid y)$$

$$= \arg\max_y P(y) \prod_i P(x_i|y)$$

Estimating the parameters of NB

- **Given dataset, count occurrences for all pairs**
  - **Count(X|i=x, Y=y)**
  - **How many pairs?**
- Prior:

$$P(Y = y) = \frac{Count(Y = y)}{\sum_{y'} Count(Y = y')}$$

- Likelihood

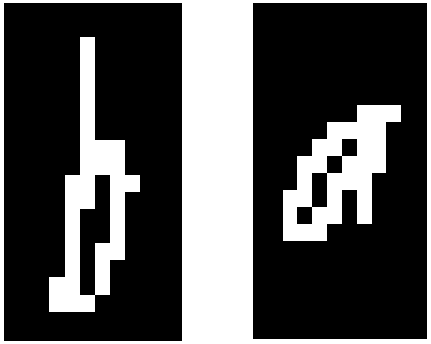
$$P(X_i = x|Y = y) = \frac{Count(X_i = x, Y = y)}{\sum_{x'} Count(X_i = x', Y = y)}$$

Bayesian Learning

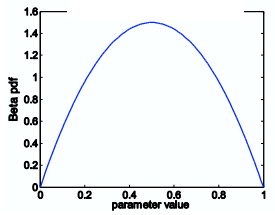
What if Features are Continuous?

Eg., Character Recognition:

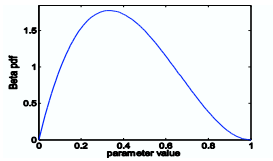
X_i is ith pixel



Prior



Posterior



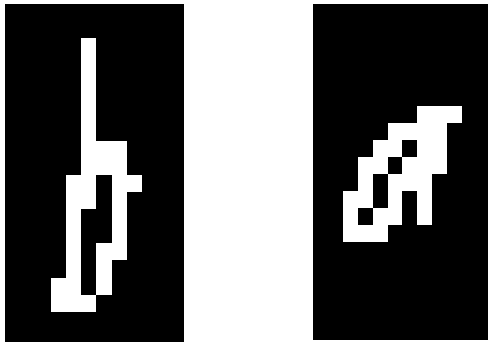
$$P(Y \mid \mathbf{X}) \propto P(\mathbf{X} \mid Y)\, P(Y)$$

Data Likelihood

Bayesian Learning: What if Features are Continuous?

Eg., Character Recognition:

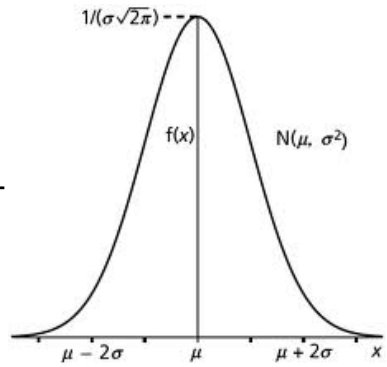
X_i is i^{th} pixel



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

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

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



Learning a Gaussian

- Collect a bunch of data
  - Hopefully, i.i.d. samples
  - e.g., exam scores
- Learn parameters
  - Mean: μ
  - Variance: σ

$X_i = i$	Exam Score
0	85
1	95
2	100
3	12
…	…
99	89

$$P(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

MLE for Gaussian:

$$P(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

- Prob. of i.i.d. samples $D=\{x_1,\ldots,x_N\}$:

$$P(\mathcal{D} \mid \mu, \sigma) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^N \prod_{i=1}^{N} e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}$$

$$\mu_{MLE}, \sigma_{MLE} = \arg\max_{\mu,\sigma} P(\mathcal{D} \mid \mu, \sigma)$$

- Log-likelihood of data:

$$\begin{aligned} \ln P(\mathcal{D} \mid \mu, \sigma) &= \ln\left[\left(\frac{1}{\sigma\sqrt{2\pi}}\right)^N \prod_{i=1}^{N} e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}\right] \\ &= -N \ln \sigma\sqrt{2\pi} - \sum_{i=1}^{N} \frac{(x_i-\mu)^2}{2\sigma^2} \end{aligned}$$

MLE for mean of a Gaussian

- What's MLE for mean?

$$\frac{d}{d\mu} \ln P(\mathcal{D} \mid \mu, \sigma) = \frac{d}{d\mu}\left[-N \ln \sigma\sqrt{2\pi} - \sum_{i=1}^{N} \frac{(x_i - \mu)^2}{2\sigma^2}\right]$$

$$= \frac{d}{d\mu}\left[-N \ln \sigma\sqrt{2\pi}\right] - \sum_{i=1}^{N} \frac{d}{d\mu}\left[\frac{(x_i - \mu)^2}{2\sigma^2}\right]$$

$$= -\sum_{i=1}^{N} \frac{(x_i - \mu)}{\sigma^2} = 0$$

$$= -\sum_{i=1}^{N} x_i + N\mu = 0$$

$$\hat{\mu}_{MLE} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

MLE for variance

- Again, set derivative to zero:

$$\begin{aligned}\frac{d}{d\sigma}\ln P(\mathcal{D}\mid\mu,\sigma) &= \frac{d}{d\sigma}\left[-N\ln\sigma\sqrt{2\pi}-\sum_{i=1}^{N}\frac{(x_i-\mu)^2}{2\sigma^2}\right]\\ &= \frac{d}{d\sigma}\left[-N\ln\sigma\sqrt{2\pi}\right]-\sum_{i=1}^{N}\frac{d}{d\sigma}\left[\frac{(x_i-\mu)^2}{2\sigma^2}\right]\\ &= -\frac{N}{\sigma}+\sum_{i=1}^{N}\frac{(x_i-\mu)^2}{\sigma^3} = 0\end{aligned}$$

$$\hat{\sigma}^2_{MLE} = \frac{1}{N}\sum_{i=1}^{N}(x_i-\hat{\mu})^2$$

Learning Gaussian parameters

- MLE:

$$\hat{\mu}_{MLE} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\hat{\sigma}^2_{MLE} = \frac{1}{N}\sum_{i=1}^{N} (x_i - \hat{\mu})^2$$

Gaussian Naïve Bayes

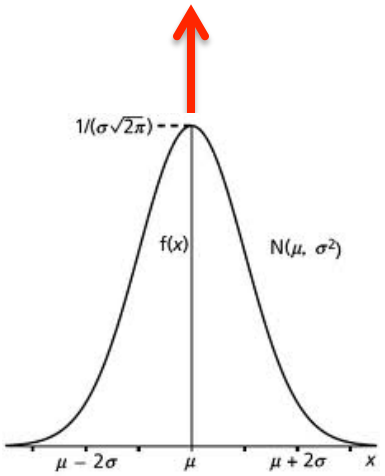
Sometimes Assume Variance

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

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

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

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



Learning Gaussian Parameters

Maximum Likelihood Estimates:

- Mean:

$$\widehat{\mu}_{MLE} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

- Variance:

$$\widehat{\sigma}^2_{MLE} = \frac{1}{N}\sum_{i=1}^{N} (x_i - \widehat{\mu})^2$$

Learning Gaussian Parameters

Maximum Likelihood Estimates:

- Mean:

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

j^{th} training example

$\delta(x)$=1 if x true, else 0

- Variance:

$$\hat{\sigma}^2_{MLE} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{\mu})^2$$

Learning Gaussian Parameters

Maximum Likelihood Estimates:

- Mean:

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

- Variance:

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

Naïve Bayes *vs.* Logistic Regression

Learning: h:**X** ↦ 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**) ∝ 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

52

Gaussian Naïve Bayes

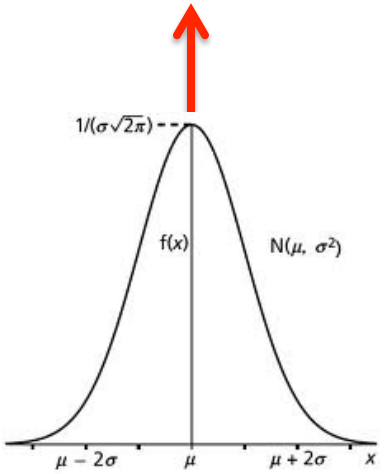
Sometimes Assume Variance

- 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 | Y = y_k) as **Gaussian** N(μ_{ik},σ_i)
  - model P(Y) as **Bernoulli** (π,1-π)
- Bayes rule to calc. P(Y|X= x)
  - P(Y | **X**) $\propto$ P(**X** | Y) P(Y)

What can we say about the form of P(Y=1 | …X_i…)?

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

Cool!!!!

54

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

- 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**

[Ng & Jordan, 2002]

Naïve Bayes vs. Logistic Regression

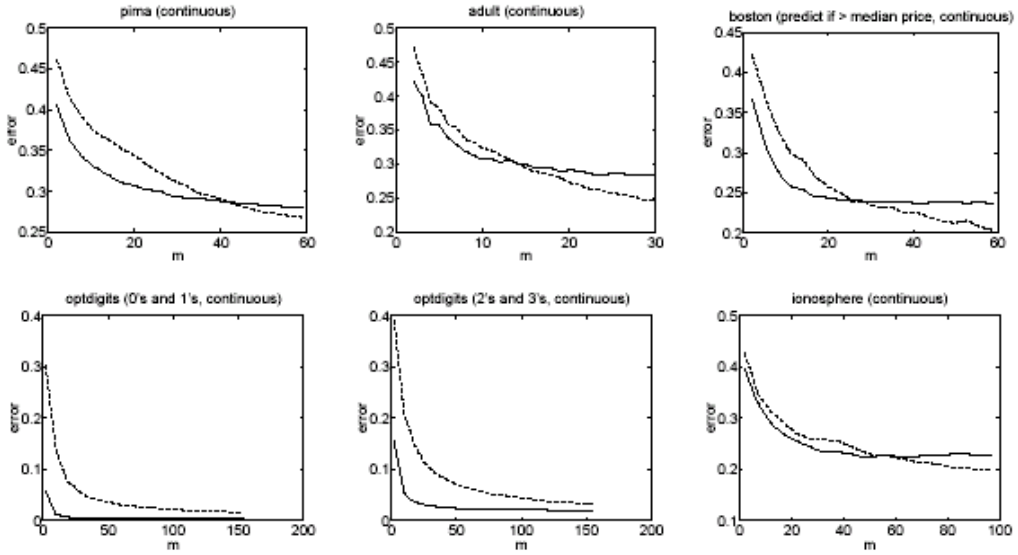
- Generative vs. Discriminative classifiers
- Non-asymptotic analysis
  - convergence rate of parameter estimates,
    (n = # of attributes in X)
    - Size of training data to get close to infinite data solution
    - Naïve Bayes needs *O*(log n) samples
    - Logistic Regression needs *O*(n) samples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

[Ng & Jordan, 2002]

Naïve bayes

Logistic Regression

Some experiments from UCI data sets



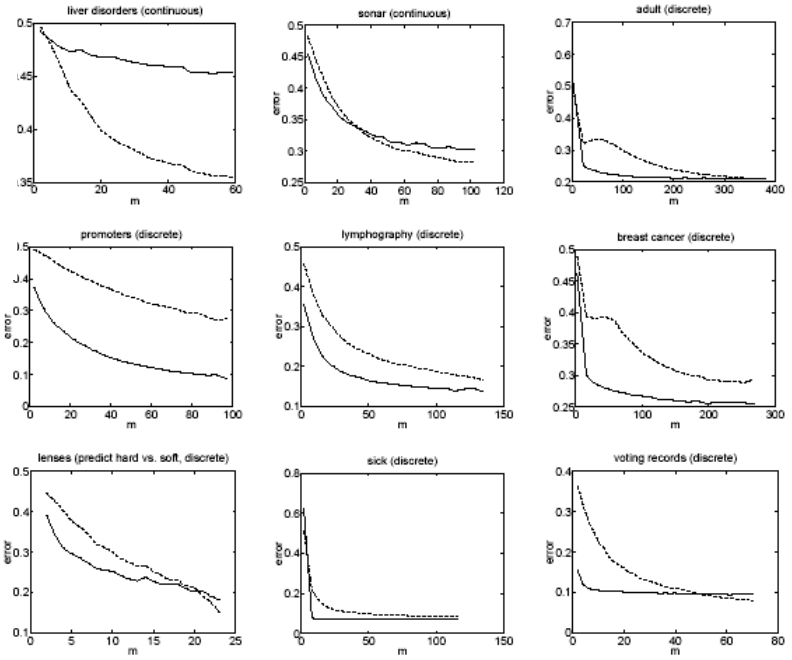


Figure 1: Results of 15 experiments on datasets from the UCI Machine Learning repository. Plots are of generalization error vs. m (averaged over 1000 random train/test splits). Dashed line is logistic regression; solid line is naive Bayes.

What you should know about 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

LR: MCAP Algorithm

- Given Data matrix of size m x (n+2) (n attributes and m examples)
  - Data[i][n+1] gives the class for example i
  - Data[i][0] is the dummy threshold attribute always set to 1.
- Arrays Pr[0..m-1] and w[0..n] initialized to random values
- Until convergence do
  - For each example i
    - Compute Pr[i]=Pr(class=1|Data[i],w)
  - Array dw[0..n] initialized to zero
  - For i=0 to n // Go over all the weights
    - For j=0 to m-1 // Go over all the training examples
      - dw[i]=dw[i]+Data[j][i]*(Data[j][n+1]-Pr[j])
  - For i=0 to n
    - $w[i]=w[i]+\eta(dw[i]-\lambda w[i])$

63