

[70240413 Statistical Machine Learning, Spring, 2017]

Probabilistic Methods for Classification

Jun Zhu

`dcszj@mail.tsinghua.edu.cn`

`http://bigml.cs.tsinghua.edu.cn/~jun`

State Key Lab of Intelligent Technology & Systems

Tsinghua University

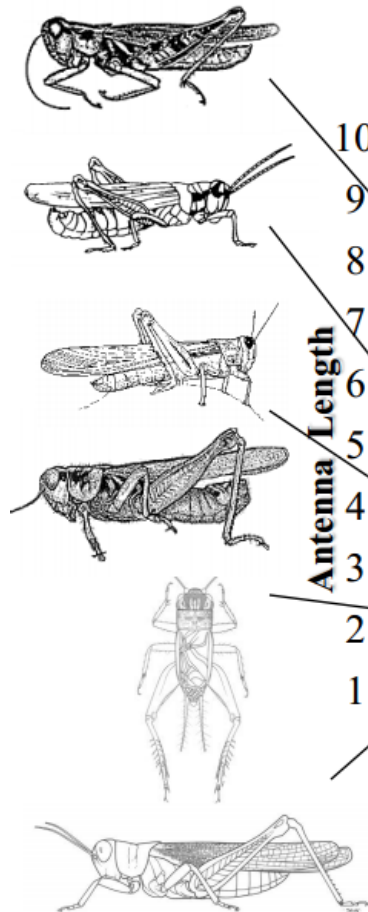
March 7, 2017

Outline

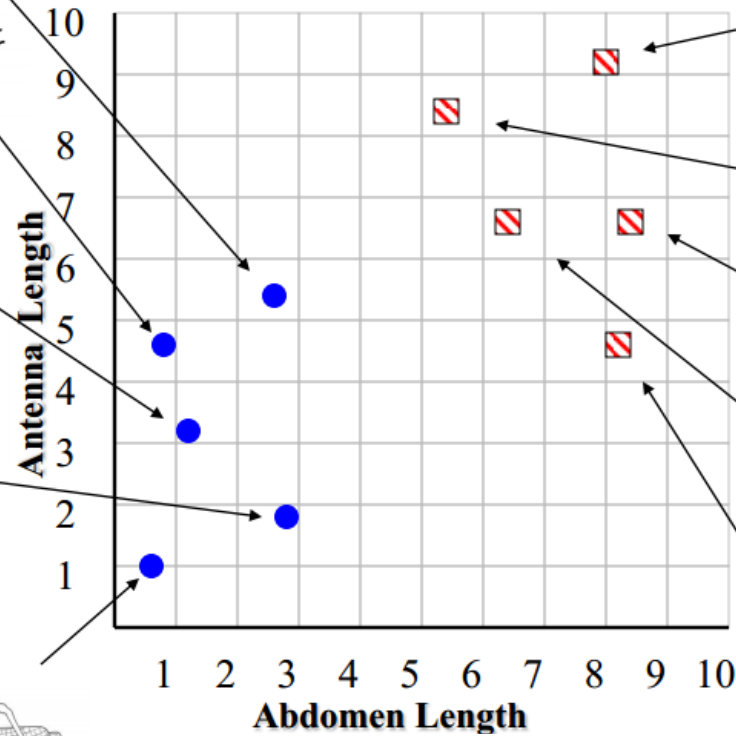
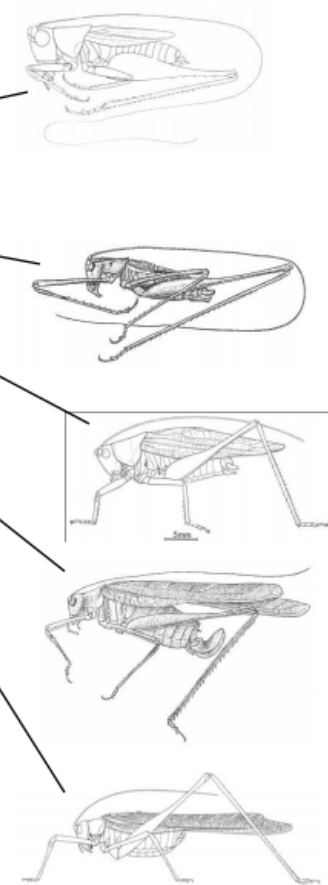
- ◆ Probabilistic methods for supervised learning
- ◆ Naive Bayes classifier
- ◆ Logistic regression
- ◆ Exponential family distributions
- ◆ Generalized linear models

An Intuitive Example

Grasshoppers



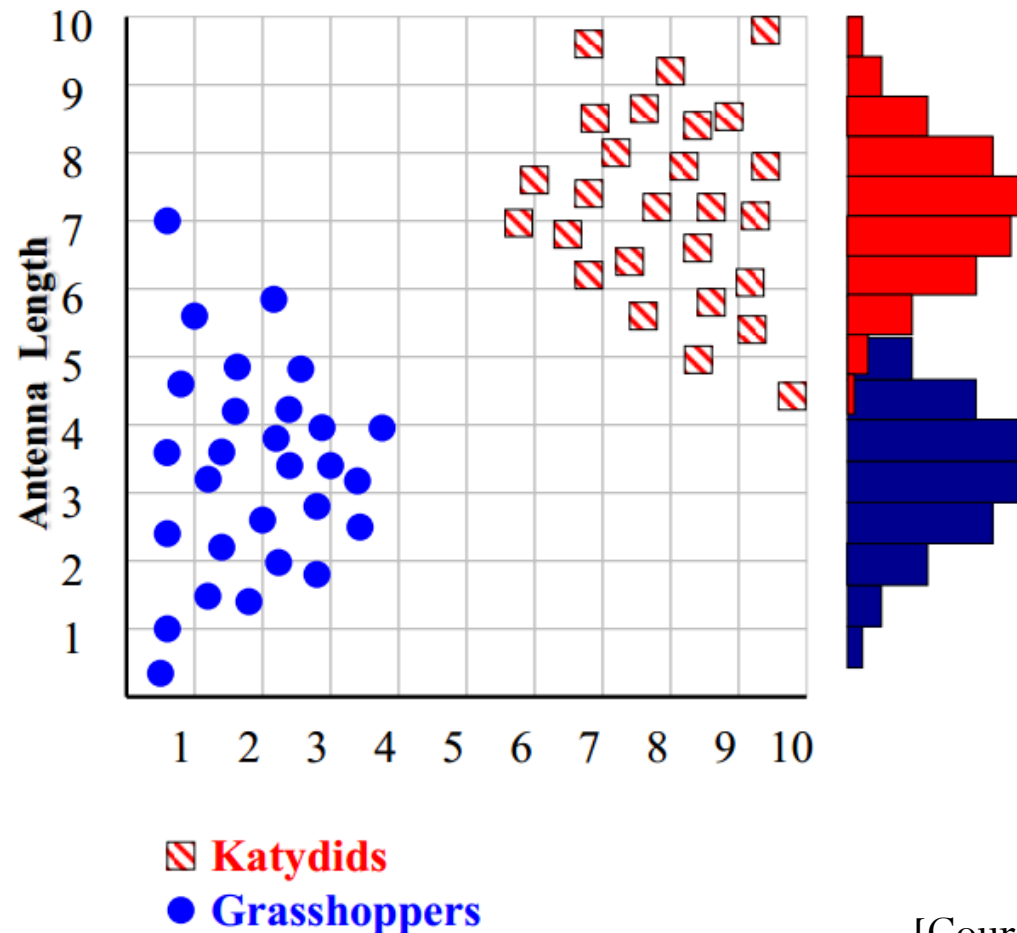
Katydid



[Courtesy of E. Keogh]

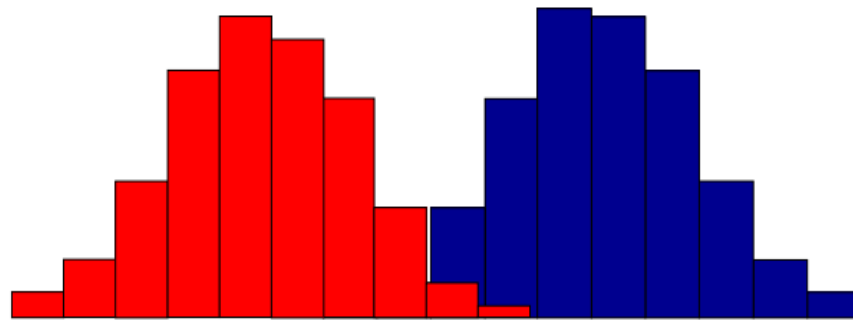
With more data ...

- ◆ Build a histogram, e.g., for “Antenna length”

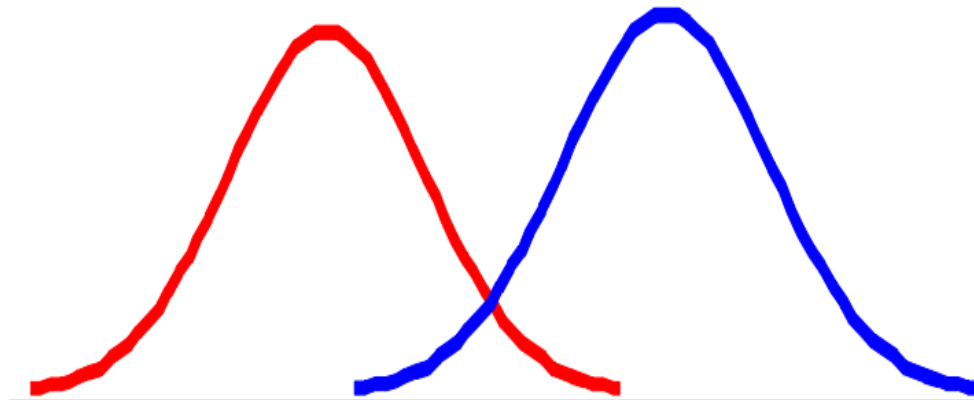


Empirical distribution

◆ Histogram (or empirical distribution)

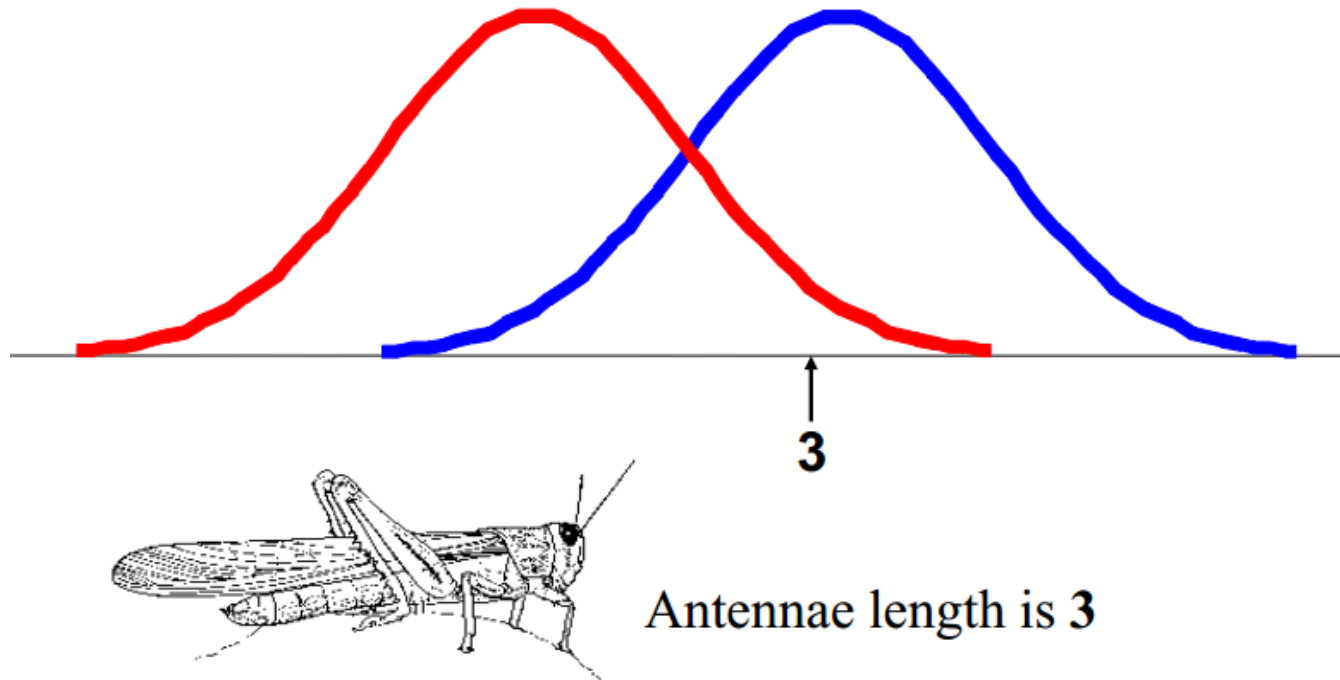


◆ Smooth with kernel density estimation (KDE):



Classification?

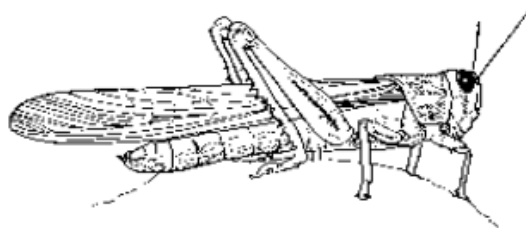
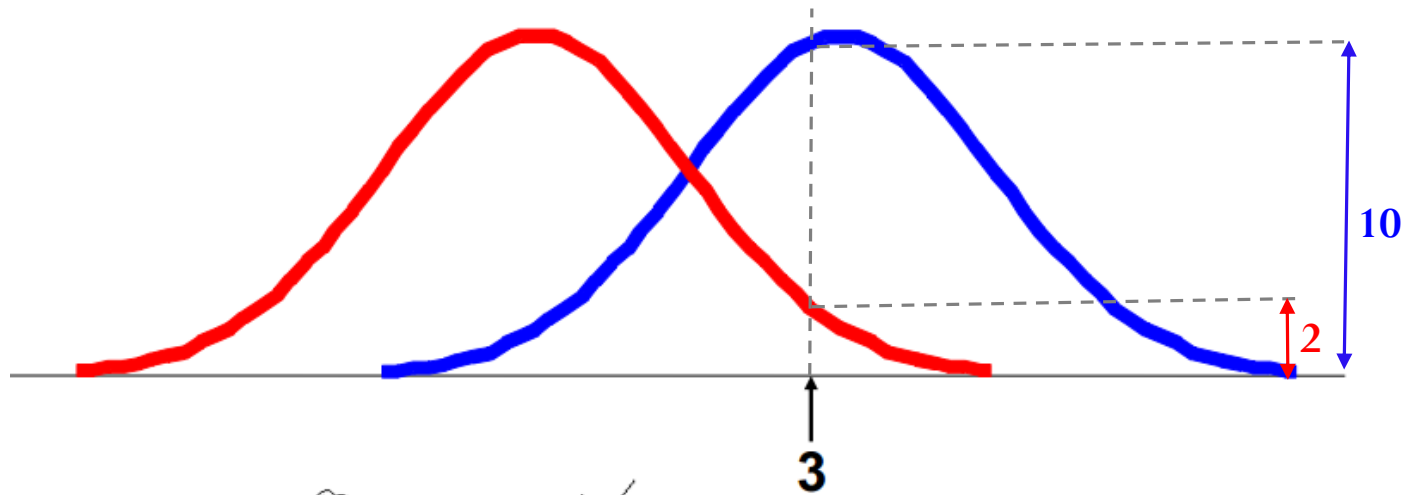
- ◆ Classify another insect we find. Its antennae are 3 units long
- ◆ Is it more probable that the insect is a **Grasshopper** or a **Katydid**?



Classification Probability

$$P(\text{Grasshopper} \mid 3) = 10 / (10 + 2) = 0.833$$

$$P(\text{Katydid} \mid 3) = 2 / (10 + 2) = 0.166$$

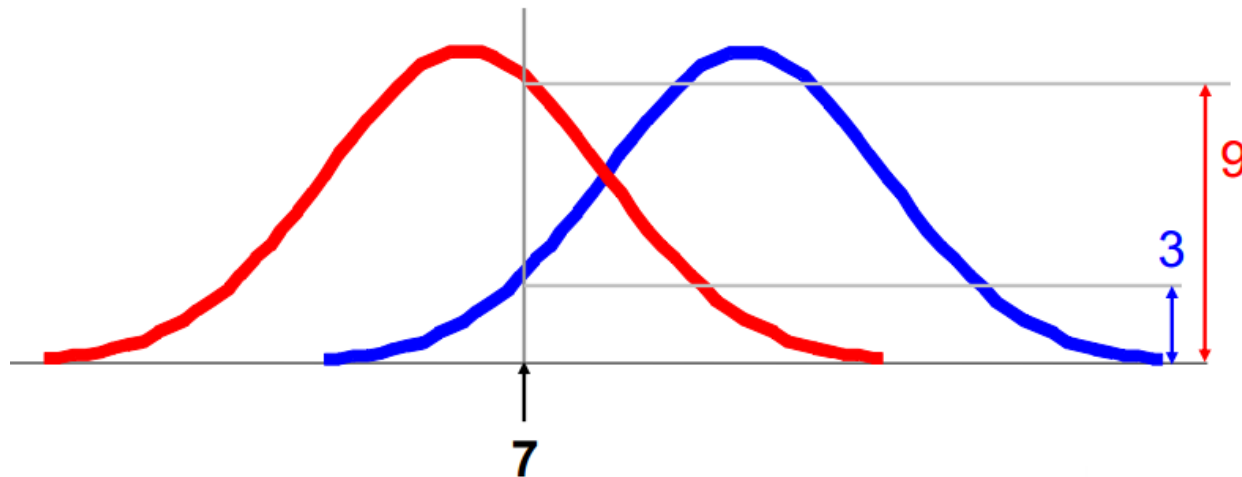


Antennae length is 3

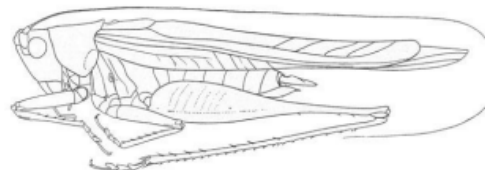
Classification Probability

$$P(\text{Grasshopper} \mid 7) = 3 / (3 + 9) = 0.250$$

$$P(\text{Katydid} \mid 7) = 9 / (3 + 9) = 0.750$$



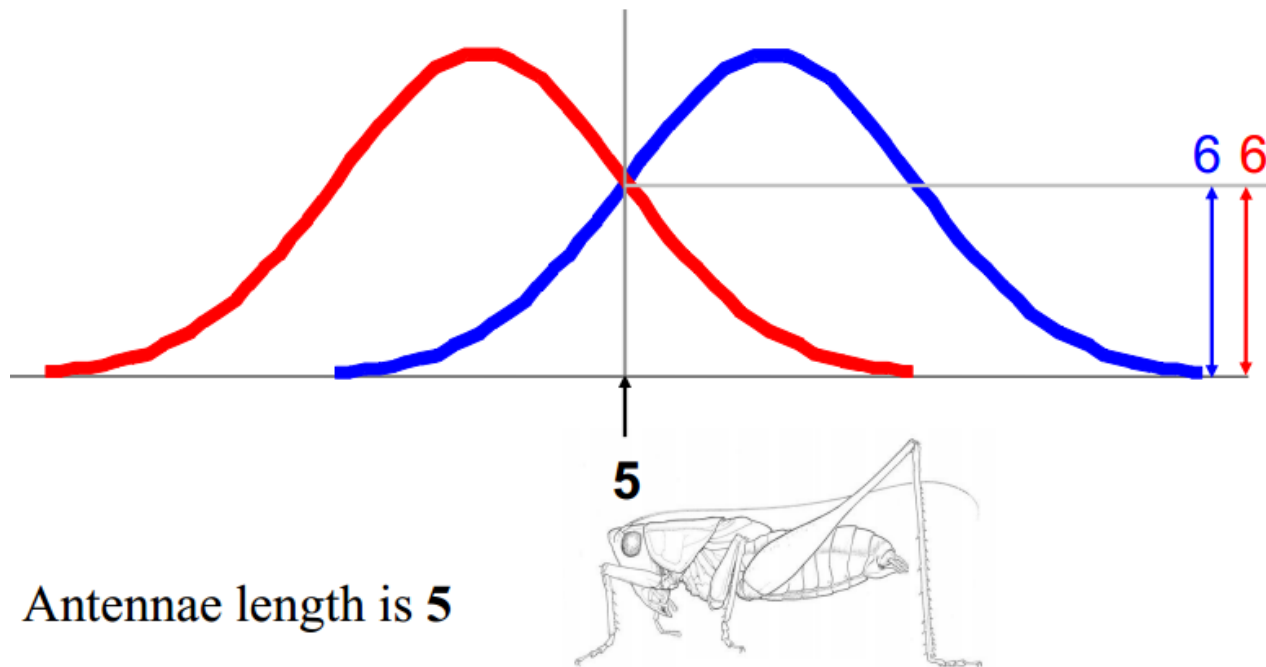
Antennae length is 7



Classification Probability

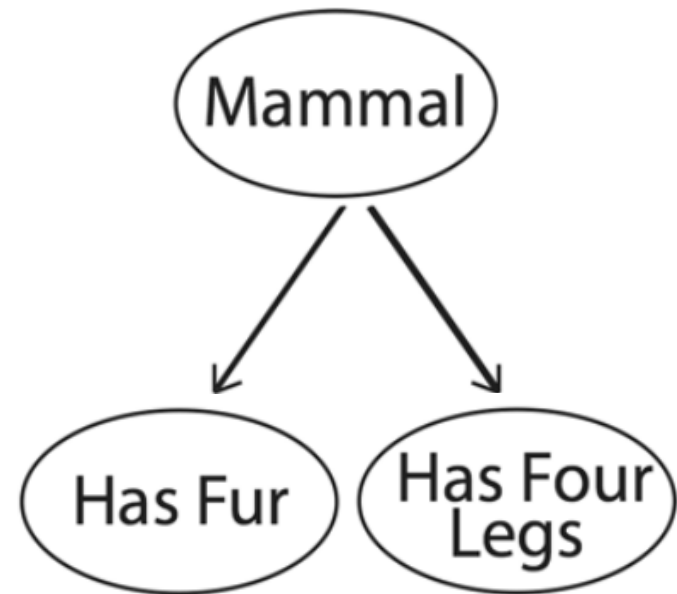
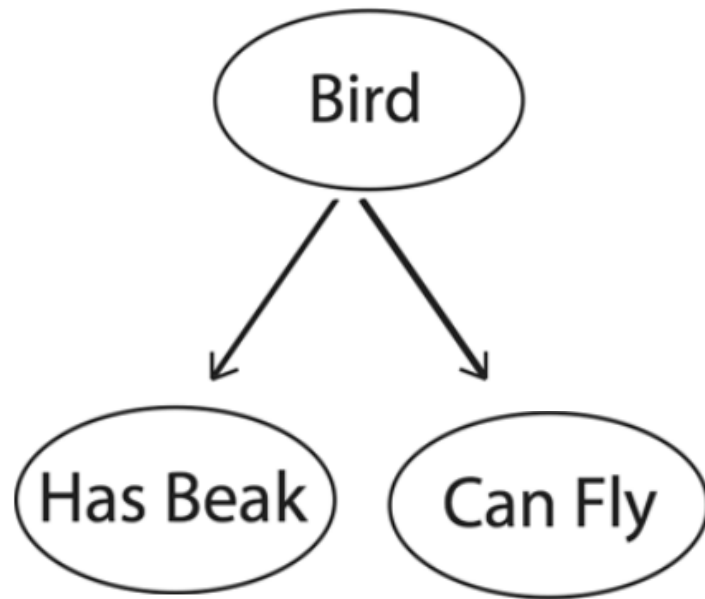
$$P(\text{Grasshopper} \mid 5) = 6 / (6 + 6) = 0.500$$

$$P(\text{Katydid} \mid 5) = 6 / (6 + 6) = 0.500$$



Naïve Bayes Classifier

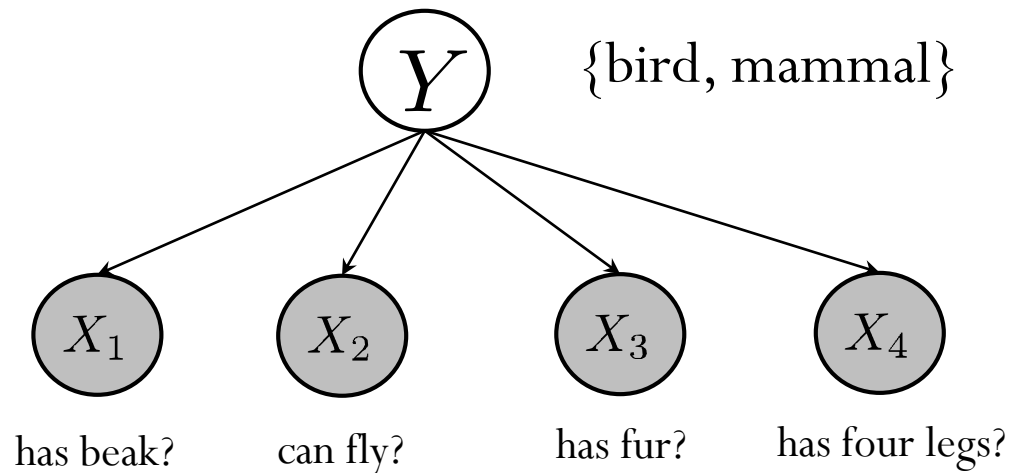
- ◆ The simplest “category-feature” generative model:
 - **Category:** “bird”, “Mammal”
 - **Features:** “has beak”, “can fly” ...



Naïve Bayes Classifier

◆ A mathematic model:

- **Naive Bayes assumption**: features X_1, \dots, X_d are conditionally independent given the class label Y



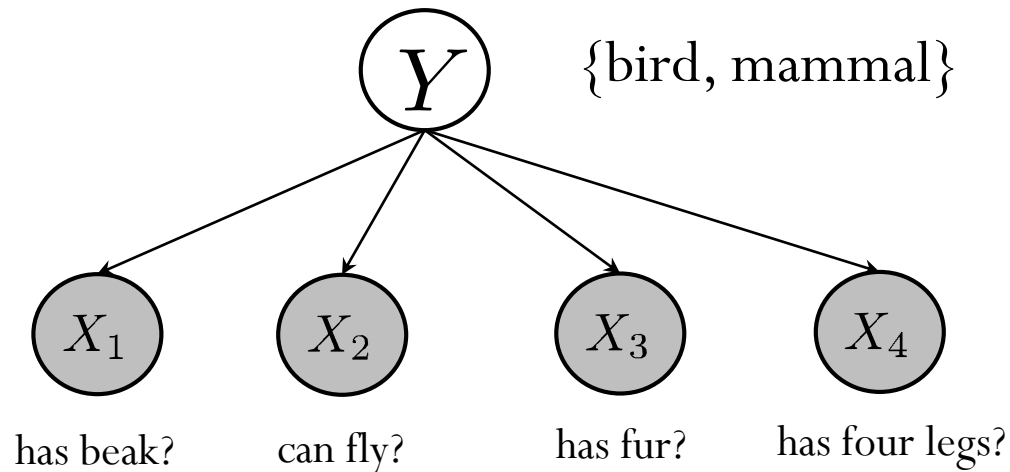
A joint distribution:

$$p(\mathbf{x}, y) = p(y)p(\mathbf{x}|y)$$

prior likelihood

Naïve Bayes Classifier

◆ A mathematic model:



Inference via Bayes rule:

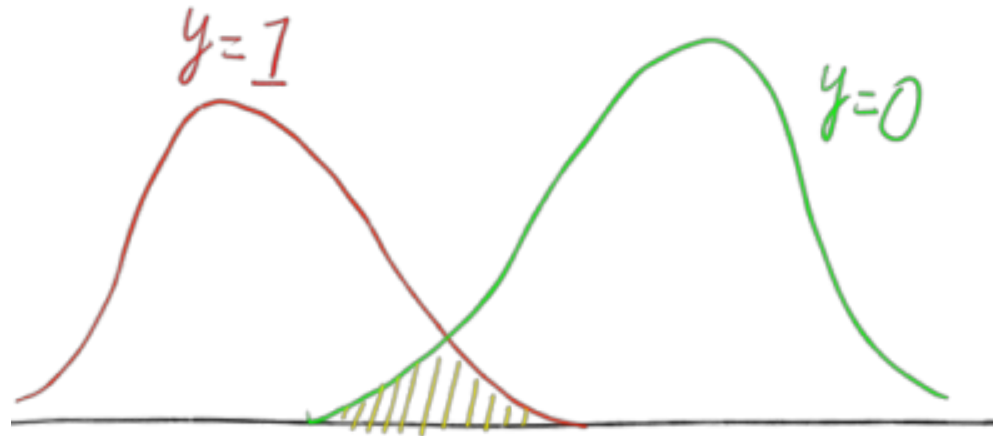
$$p(y|x) = \frac{p(x, y)}{p(x)} = \frac{p(y)p(x|y)}{p(x)}$$

Bayes' decision rule:

$$y^* = \arg \max_{y \in \mathcal{Y}} p(y|x)$$

Bayes Error

◆ **Theorem:** Bayes classifier is optimal!



$$p(error|\mathbf{x}) = \begin{cases} p(y = 1|\mathbf{x}) & \text{if we decide } y = 0 \\ p(y = 0|\mathbf{x}) & \text{if we decide } y = 1 \end{cases}$$

$$p(error) = \int_{-\infty}^{\infty} p(error|\mathbf{x})p(\mathbf{x})d\mathbf{x}$$

◆ *However, the true distribution is **unknown**.*

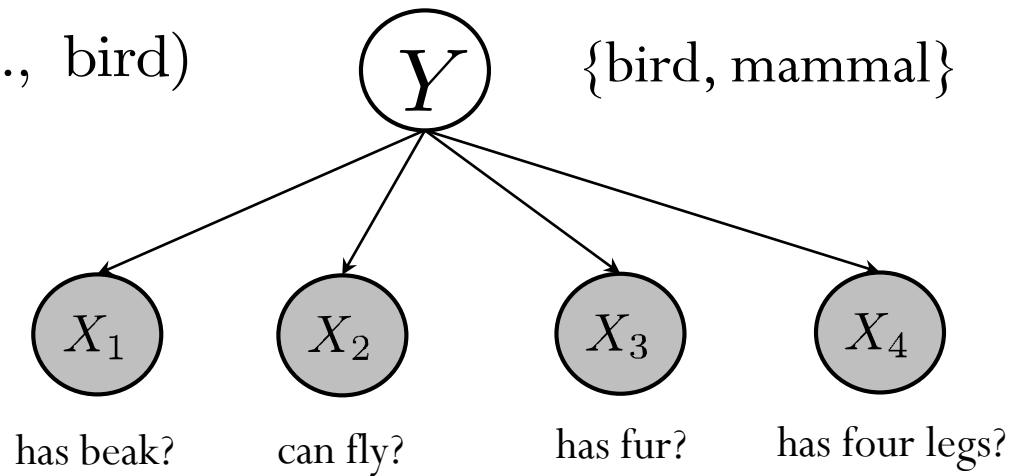
◆ *We need to estimate it!*

Naïve Bayes Classifier

◆ How to learn model parameters?

- Assume X are d binary features, Y has 2 possible labels

$$p(y|\pi) = \begin{cases} \pi & \text{if } y = 1 \text{ (i.e., bird)} \\ 1 - \pi & \text{otherwise} \end{cases}$$



$$p(x_j|y=0, q) = \begin{cases} q_{0j} & \text{if } x_j = 1 \\ 1 - q_{0j} & \text{otherwise} \end{cases} \quad p(x_j|y=1, q) = \begin{cases} q_{1j} & \text{if } x_j = 1 \\ 1 - q_{1j} & \text{otherwise} \end{cases}$$

- *How many parameters to estimate?*

Naïve Bayes Classifier

◆ How to learn model parameters?

◆ A set of training data:

- (1, 1, 0, 0; 1)
- (1, 0, 0, 0; 1)
- (0, 1, 1, 0; 0)
- (0, 0, 1, 1; 0)

◆ Maximum likelihood estimation (N : # of training data)

$$p(\{\mathbf{x}_i, y_i | \pi, q\}) = \prod_{i=1}^N p(\mathbf{x}_i, y_i | \pi, q)$$

Naïve Bayes Classifier

◆ **Maximum likelihood estimation** (N : # of training data)

$$(\hat{\pi}, \hat{q}) = \arg \max_{\pi, q} p(\{\mathbf{x}_i, y_i\} | \pi, q)$$

$$(\hat{\pi}, \hat{q}) = \arg \max_{\pi, q} \log p(\{\mathbf{x}_i, y_i\} | \pi, q)$$

◆ **Results** (count frequency! Exercise?):

$$\hat{\pi} = \frac{N_1}{N} \quad \hat{q}_{0j} = \frac{N_0^j}{N_0} \quad \hat{q}_{1j} = \frac{N_1^j}{N_1}$$

$$N_k = \sum_{i=1}^N \mathbf{I}(y_i = k) : \text{ \# of data in category } k$$

$$N_k^j = \sum_{i=1}^N \mathbf{I}(y_i = k, x_{ij} = 1) : \text{ \# of data in category } k \text{ that has feature } j$$

Naïve Bayes Classifier

◆ Data scarcity issue (zero-counts problem):

$$\hat{\pi} = \frac{N_1}{N} \quad \hat{q}_{0j} = \frac{N_0^j}{N_0} \quad \hat{q}_{1j} = \frac{N_1^j}{N_1}$$

□ *How about if some features do not appear?*

◆ Laplace smoothing (Additive smoothing):

$$\hat{q}_{0j} = \frac{N_0^j + \alpha}{N_0 + 2\alpha}$$

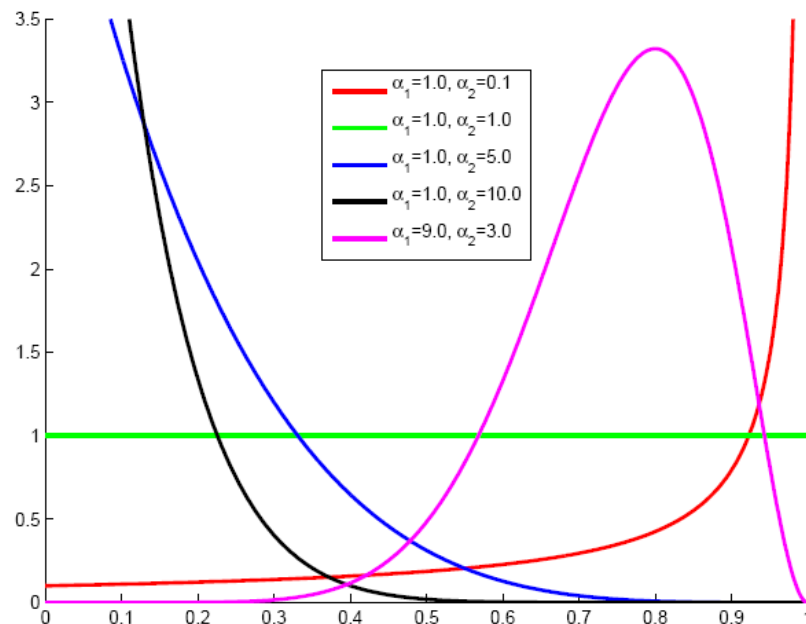
$$\alpha > 0$$

$$\hat{q}_{1j} = \frac{N_1^j + \alpha}{N_1 + 2\alpha}$$

A Bayesian Treatment

◆ Put a prior on the parameters

$$p_0(q_{0j}|\alpha_1, \alpha_2) = \text{Beta}(\alpha_1, \alpha_2) = \frac{\Gamma(\alpha_1 + \alpha_2)}{\Gamma(\alpha_1)\Gamma(\alpha_2)} q_{0j}^{\alpha_1-1} (1 - q_{0j})^{\alpha_2-1}$$



A Bayesian Treatment

◆ Maximum a Posterior Estimate (MAP):

$$\begin{aligned}\hat{q} &= \arg \max_q \log p(q | \{\mathbf{x}_i, y_i\}) \\ &= \arg \max_q \log p_0(q) + \log p(\{\mathbf{x}_i, y_i\} | q)\end{aligned}$$

◆ Results (**Exercise?**):

$$\hat{q}_{0j} = \frac{N_0^j + \alpha_1 - 1}{N_0 + \alpha_1 + \alpha_2 - 2}$$

$$\hat{q}_{1j} = \frac{N_1^j + \alpha_1 - 1}{N_1 + \alpha_1 + \alpha_2 - 2}$$

A Bayesian Treatment

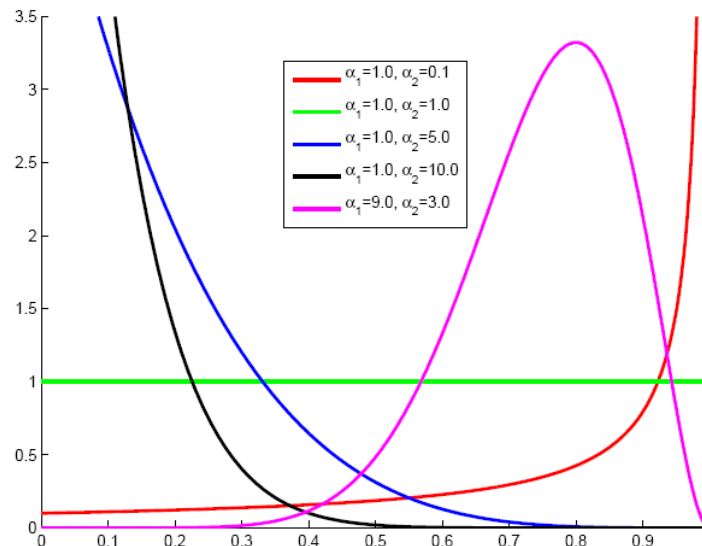
◆ Maximum a Posterior Estimate (MAP):

$$\hat{q}_{0j} = \frac{N_0^j + \alpha_1 - 1}{N_0 + \alpha_1 + \alpha_2 - 2}$$

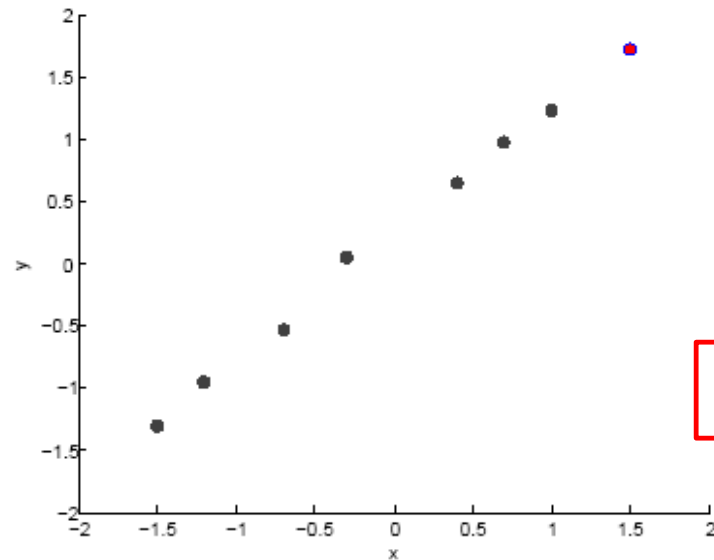
◆ If $\alpha_1 = \alpha_2 = 1$ (**non-informative prior**), no effect

□ MLE is a special case of Bayesian estimate

◆ Increase α_1, α_2 , lead to heavier influence from prior



Bayesian Regression



$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

- Goal: learn a function from noisy observed data

- Linear $\mathcal{F}_{linear} = \{f : f = wx + b, w, b \in \mathbb{R}\}$

- Polynomial $\mathcal{F}_{polynomial} = \{f : f = \sum_k w_k x^k, w_k \in \mathbb{R}\}$
- ...

□ . . .

Bayesian Regression

- ◆ Noisy observations

$$y = f(\mathbf{x}) + \epsilon, \text{ where } \epsilon \sim \mathcal{N}(0, \sigma_n^2)$$

- ◆ Gaussian likelihood function for linear regression $f(\mathbf{x}_i) = \mathbf{w}^\top \mathbf{x}_i$

$$p(\mathbf{y}|X, \mathbf{w}) = \prod_{i=1}^N p(y_i|\mathbf{x}_i, \mathbf{w}) = \mathcal{N}(X^\top \mathbf{w}, \sigma_n^2 I)$$

- ◆ Gaussian prior (Conjugate)

$$\mathbf{w} \sim \mathcal{N}(0, \Sigma_d)$$

- ◆ Inference with Bayes' rule

- Posterior $p(\mathbf{w}|X, \mathbf{y}) = \mathcal{N}(\frac{1}{\sigma_n^2} A^{-1} X \mathbf{y}, A^{-1}), \text{ where } A = \sigma_n^{-2} X X^\top + \Sigma_d^{-1}$

- Marginal likelihood

- Prediction

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|X, \mathbf{w}) p(\mathbf{w}) d\mathbf{w}$$

$$p(f_*|\mathbf{x}_*, X, \mathbf{y}) = \int p(f_*|\mathbf{x}_*, \mathbf{w}) p(\mathbf{w}|X, \mathbf{y}) d\mathbf{w} = \mathcal{N}\left(\frac{1}{\sigma_n^2} \mathbf{x}_*^\top A^{-1} X \mathbf{y}, \mathbf{x}_*^\top A^{-1} \mathbf{x}_*\right)$$

Extensions of NB

- ◆ We covered the case with binary features and binary class labels
- ◆ NB is applicable to the cases:
 - Discrete features + discrete class labels
 - Continuous features + discrete class labels
 - ...
- ◆ More dependency between features can be considered
 - Tree augmented NB
 - ...

Gaussian Naive Bayes (GNB)

◆ E.g.: character recognition: feature X_i is intensity at pixel i :



◆ The generative process is

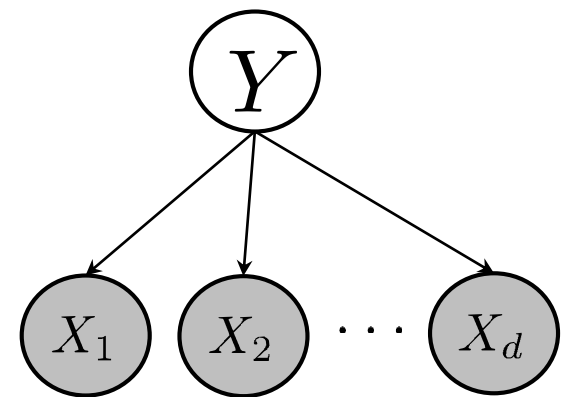
$$Y \sim \text{Bernoulli}(\pi)$$

$$P(X_i|Y = y) = \mathcal{N}(\mu_{iy}, \sigma_{iy}^2)$$

□ Different mean and variance for each class k and each feature i

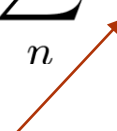
◆ Sometimes assume variance is:

- independent of Y (i.e., σ_i)
- or independent of X (i.e., σ_y)
- or both (i.e., σ)



Estimating Parameters & Prediction

◆ MLE estimates

$$\hat{\mu}_{ik} = \frac{1}{\sum_n \mathbb{I}(y_n = k)} \sum_n x_{ni} \mathbb{I}(y_n = k)$$


pixel i in
training image n

$$\hat{\sigma}_{ik}^2 = \frac{1}{\sum_n \mathbb{I}(y_n = k)} \sum_n (x_{ni} - \hat{\mu}_{ik})^2 \mathbb{I}(y_n = k)$$

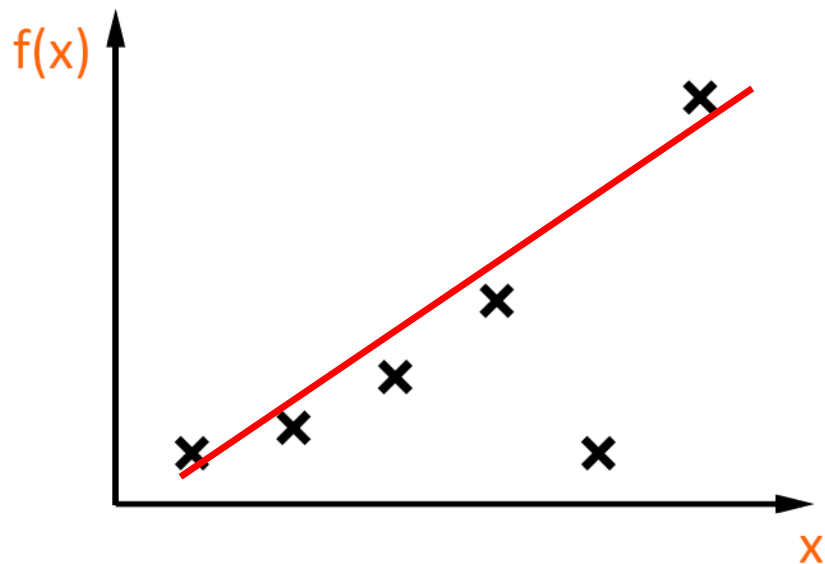
◆ Prediction:

$$h(\mathbf{x}) = \operatorname{argmax}_y P(y) \prod_i P(x_i|y)$$

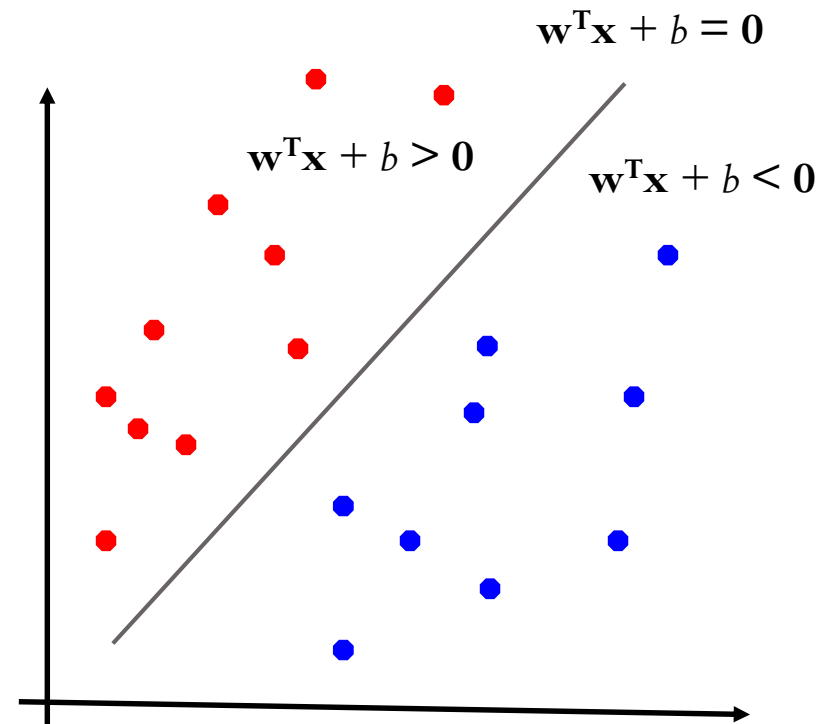
What you need to know about NB classifier

- ◆ What's the assumption
- ◆ Why we use it
- ◆ How do we learn it
- ◆ Why is Bayesian estimation (MAP) important

Linear regression and linear classification



Linear fit



Linear decision boundary

What's the decision boundary of NB?

◆ Is it linear or non-linear?

◆ There are several distributions that lead to a linear decision boundary, e.g., **GNB with equal variance**

$$P(X_i|Y = y) = \mathcal{N}(\mu_{iy}, \sigma_i^2)$$

□ Decision boundary (??):

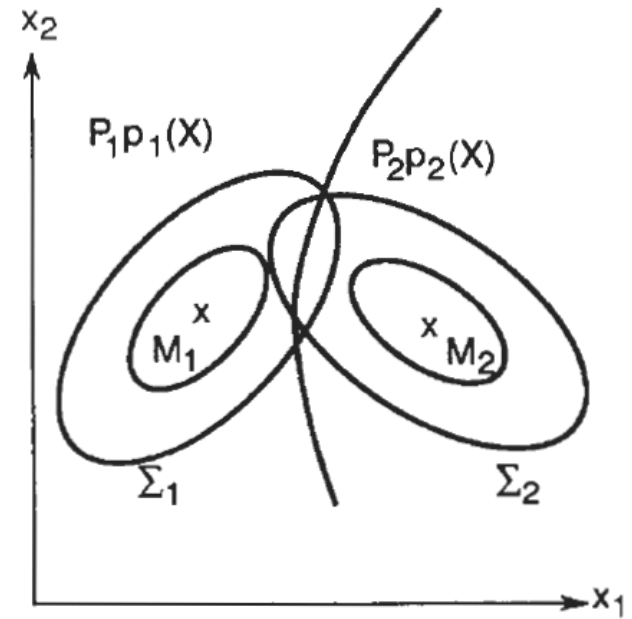
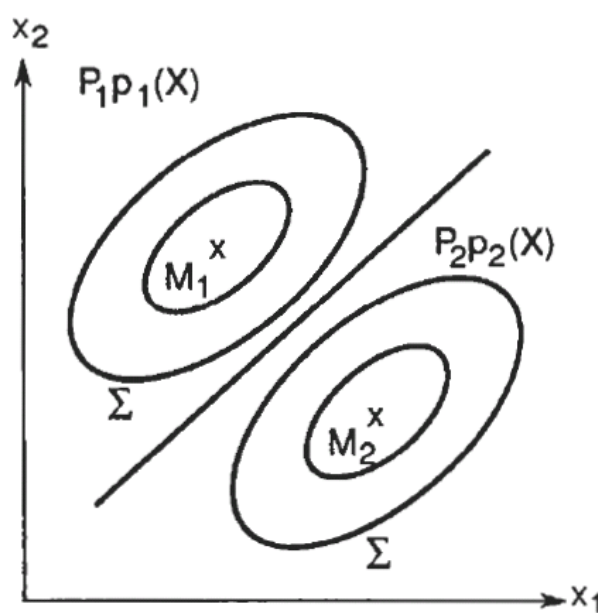
$$\log \frac{\prod_{i=1}^d P(X_i|Y = 0)P(Y = 0)}{\prod_{i=1}^d P(X_i|Y = 1)P(Y = 1)} = 0$$

$$\Rightarrow \log \frac{1 - \pi}{\pi} + \sum_i \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} + \sum_i \frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} x_i = 0$$

$$\Rightarrow w_0 + \sum_i w_i x_i = 0$$

Gaussian Naive Bayes (GNB)

◆ Decision boundary (the general multivariate Gaussian case):



$$P_1 = P(Y = 0), \quad P_2 = P(Y = 1)$$

$$p_1(X) = p(X|Y = 0) = \mathcal{N}(M_1, \Sigma_1)$$

$$p_2(X) = p(X|Y = 1) = \mathcal{N}(M_2, \Sigma_2)$$

The predictive distribution of GNB

- ◆ Understanding the predictive distribution

$$p(y = 1|\mathbf{x}, \mu, \Sigma, \pi) = \frac{p(y = 1, \mathbf{x}|\mu, \Sigma, \pi)}{p(\mathbf{x}|\mu, \Sigma, \pi)}$$

- ◆ Under naive Bayes assumption:

$$\begin{aligned} p(y = 1|\mathbf{x}, \mu, \Sigma, \pi) &= \frac{1}{1 + \frac{p(y=0, \mathbf{x}|\mu, \Sigma, \pi)}{p(y=1, \mathbf{x}|\mu, \Sigma, \pi)}} \\ &= \frac{1}{1 + \frac{(1-\pi) \prod_i \mathcal{N}(x_i|\mu_{i0}, \sigma_i^2)}{\pi \prod_i \mathcal{N}(x_i|\mu_{i1}, \sigma_i^2)}} \\ &= \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x} - w_0)} \end{aligned}$$

- ◆ **Note:** For multi-class, the predictive distribution is softmax!

Generative vs. Discriminative Classifiers

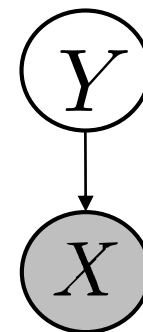
◆ Generative classifiers (e.g., Naive Bayes)

- Assume some functional form for $P(\mathbf{X}, Y)$ (or $P(Y)$ and $P(\mathbf{X} | Y)$)
- Estimate parameters of $P(\mathbf{X}, Y)$ directly from training data
- Make prediction

$$\hat{y} = \operatorname{argmax}_y P(\mathbf{x}, Y = y)$$

- But, we note that

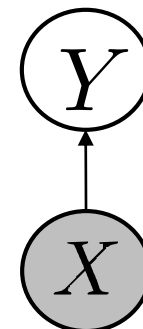
$$\hat{y} = \operatorname{argmax}_y P(Y = y | \mathbf{x})$$



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

◆ Discriminative classifiers (e.g., Logistic regression)

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



Logistic Regression

◆ Recall the predictive distribution of GNB!

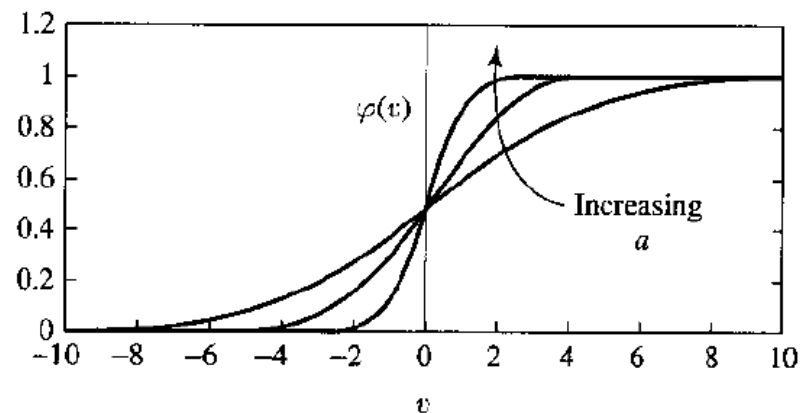
◆ Assume the following functional form for $P(Y | X)$

$$P(y = 1 | \mathbf{x}) = \frac{1}{1 + \exp(-(w_0 + \mathbf{w}^\top \mathbf{x}))}$$

□ Logistic function (or **Sigmoid**) applied to a linear function of the data (for $\alpha = 1$):

$$\psi_\alpha(v) = \frac{1}{1 + \exp(-\alpha v)}$$

$\alpha \rightarrow \infty$: step function



use a large α can be good for some neural networks

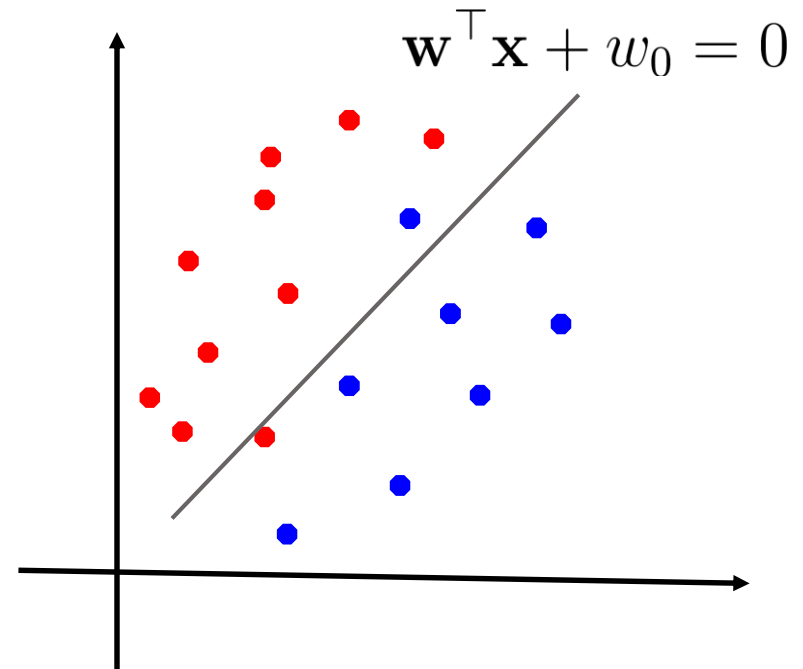
Logistic Regression

- ◆ What's the decision boundary of logistic regression? (linear or nonlinear?)

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-(w_0 + \mathbf{w}^\top \mathbf{x}))}$$

$$\log \frac{P(Y = 1|\mathbf{x})}{P(y = 0|\mathbf{x})} = 0$$

$$\mathbf{w}^\top \mathbf{x} + w_0 = 0$$



Logistic regression is a linear classifier!

Representation

◆ Logistic regression

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-(w_0 + \mathbf{w}^\top \mathbf{x}))}$$

◆ For notation simplicity, we use the augmented vector:

$$\text{input features : } \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \quad \text{model weights : } \begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix}$$

□ Then, we have

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x})}$$

Multiclass Logistic Regression

- ◆ For more than 2 classes, where $y \in \{1, \dots, K\}$, logistic regression classifier is defined as

$$\forall k < K : P(Y = k|\mathbf{x}) = \frac{\exp(\mathbf{w}_k^\top \mathbf{x})}{1 + \sum_{j=1}^{K-1} \exp(\mathbf{w}_j^\top \mathbf{x})}$$

$$P(Y = K|\mathbf{x}) = \frac{1}{1 + \sum_{j=1}^{K-1} \exp(\mathbf{w}_j^\top \mathbf{x})}$$

- Well normalized distribution! No weights for class K!

- ◆ Is the decision boundary still linear?

Training Logistic Regression

- ◆ We consider the binary classification

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x})}$$

- ◆ Training data $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$

- ◆ How to learn the parameters?

- ◆ Can we do MLE?

$$\hat{\mathbf{w}}_{MLE} = \underset{\mathbf{w}}{\operatorname{argmax}} \prod_{i=1}^N P(\mathbf{x}_i, y_i | \mathbf{w})$$

□ *No! Don't have a model for $P(X)$ or $P(X | Y)$*

- ◆ Can we do large-margin learning?

Maximum Conditional Likelihood Estimate

- ◆ We learn the parameters by solving

$$\hat{\mathbf{w}} = \operatorname{argmax}_{\mathbf{w}} \prod_{i=1}^N P(y_i | \mathbf{x}_i, \mathbf{w})$$

- ◆ **Discriminative philosophy** – don't waste effort on learning $P(X)$, focus on $P(Y | X)$ – that's all that matters for classification!

Maximum Conditional Likelihood Estimate

$$\hat{\mathbf{w}} = \operatorname{argmax}_{\mathbf{w}} \prod_{i=1}^N P(y_i | \mathbf{x}_i, \mathbf{w})$$

$$P(y = 1 | \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x})}$$

◆ We have:

$$\begin{aligned} \mathcal{L}(\mathbf{w}) &= \log \prod_{i=1}^N P(y_i | \mathbf{x}_i, \mathbf{w}) \\ &= \sum_i \left[y_i \mathbf{w}^\top \mathbf{x}_i - \log(1 + \exp(\mathbf{w}^\top \mathbf{x}_i)) \right] \end{aligned}$$

Maximum Conditional Likelihood Estimate

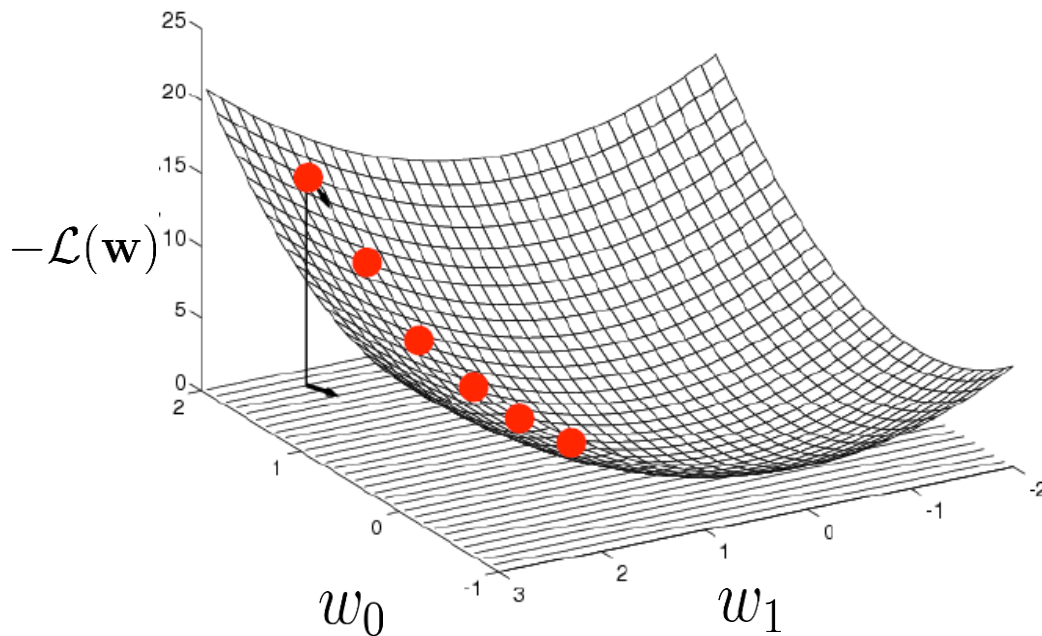
$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} \mathcal{L}(\mathbf{w})$$

$$\mathcal{L}(\mathbf{w}) = \sum_i [y_i \mathbf{w}^\top \mathbf{x}_i - \log(1 + \exp(\mathbf{w}^\top \mathbf{x}_i))]$$

- ◆ **Bad news:** no closed-form solution!
- ◆ **Good news:** $\mathcal{L}(\mathbf{w})$ is a concave function of \mathbf{w} !
 - Is the original logistic function concave?

Optimizing concave/convex function

- ◆ Conditional likelihood for logistic regression is concave
- ◆ Maximum of a concave function = minimum of a convex function
 - Gradient ascent (concave) / Gradient descent (convex)



Gradient:


$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \begin{pmatrix} \frac{\partial \mathcal{L}(\mathbf{w})}{\partial w_0} \\ \vdots \\ \frac{\partial \mathcal{L}(\mathbf{w})}{\partial w_d} \end{pmatrix}$$

Update rule:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w})|_{\mathbf{w}_t}$$

Gradient Ascent for Logistic Regression

- ◆ Property of sigmoid function


$$\Rightarrow \nabla_{\nu} \psi = \psi(1 - \psi)$$

- ◆ Gradient ascent algorithm iteratively does:

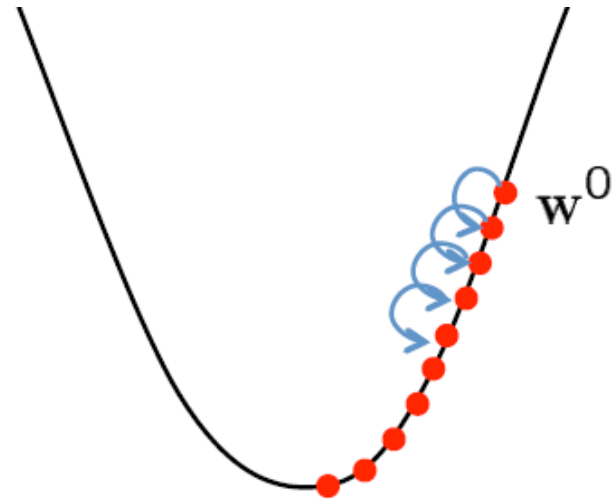
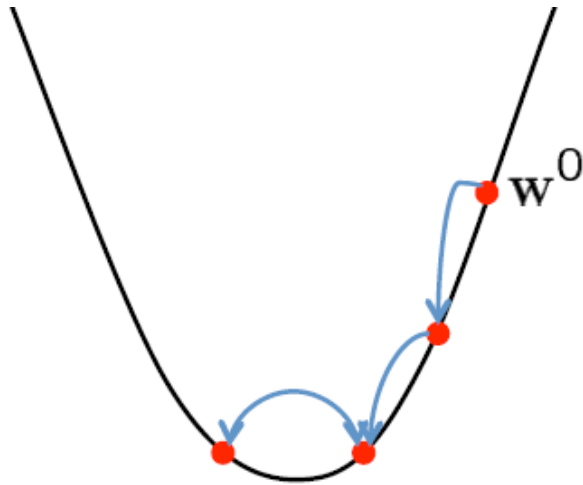
$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \sum_{i=1}^N \mathbf{x}_i (y_i - \mu_i^t)$$

- where $\mu_i^t = P(y = 1 | \mathbf{x}_i, \mathbf{w}_t)$ is the prediction made by the current model
- ◆ Until the change (of objective or gradient) falls below some threshold

Issues

- ◆ Gradient descent is the simplest optimization methods, faster convergence can be obtained by using
 - E.g., Newton method, conjugate gradient ascent, IRLS (iterative reweighted least squares)
- ◆ The vanilla logistic regression often over-fits; using a regularization can help a lot!

Effects of step-size



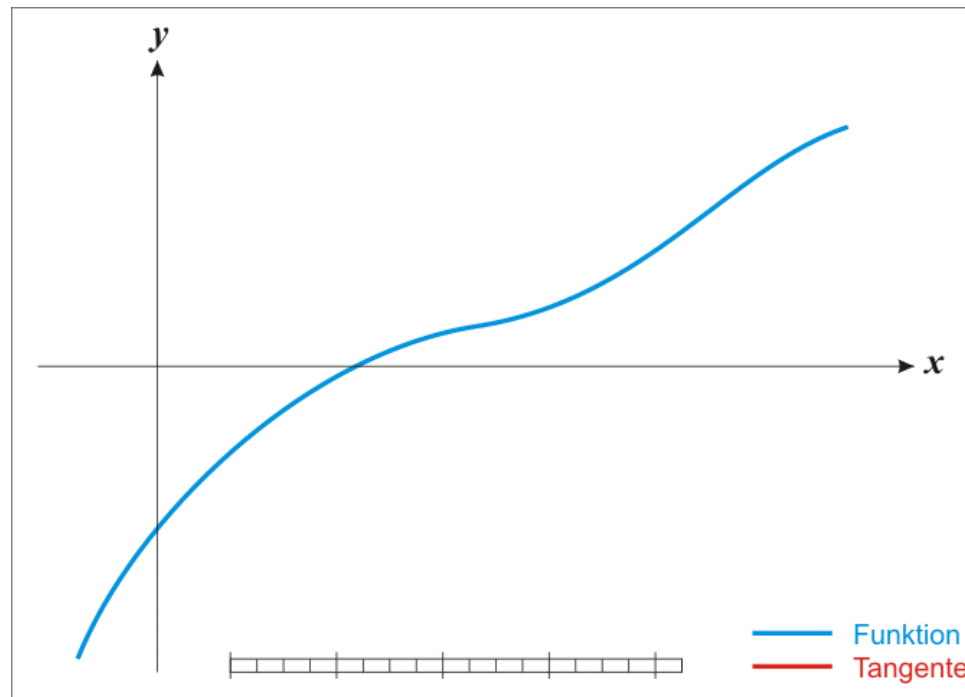
- ◆ Large $\eta \Rightarrow$ fast convergence but larger residual error; Also possible oscillations
- ◆ Small $\eta \Rightarrow$ slow convergence but small residual error

The Newton's Method

◆ AKA: Newton-Raphson method

◆ A method that finds the root of: $f(x) = 0$

$$x_{t+1} = x_t - \frac{f(x_t)}{f'(x_t)}$$



The Newton's Method

- ◆ To maximize the conditional likelihood

$$\mathcal{L}(\mathbf{w}) = \sum_i [y_i \mathbf{w}^\top \mathbf{x}_i - \log(1 + \exp(\mathbf{w}^\top \mathbf{x}_i))]$$

- We need to find \mathbf{w}^* such that

$$\nabla \mathcal{L}(\mathbf{w}^*) = 0$$

- ◆ So we can perform the following iteration:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - H^{-1} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w})|_{\mathbf{w}_t}$$

- where H is known as the Hessian matrix:

$$H = \nabla_{\mathbf{w}}^2 \mathcal{L}(\mathbf{w})|_{\mathbf{w}_t}$$

Newton's Method for LR

◆ The update equation

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - H^{-1} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w})|_{\mathbf{w}_t}$$

□ where the gradient is:

$$\begin{aligned} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w})|_{\mathbf{w}_t} &= \sum_i (y_i - \mu_i) \mathbf{x}_i = X(\mathbf{y} - \boldsymbol{\mu}) \\ \mu_i &= \psi(\mathbf{w}_t^\top \mathbf{x}_i) \end{aligned}$$

□ The Hessian matrix is:

$$H = \nabla_{\mathbf{w}}^2 \mathcal{L}(\mathbf{w})|_{\mathbf{w}_t} = - \sum_i \mu_i (1 - \mu_i) \mathbf{x}_i \mathbf{x}_i^\top = -X R X^\top$$

$$\text{where } R_{ii} = \mu_i (1 - \mu_i)$$

Iterative reweighted least squares (IRLS)

◆ In least square estimate of linear regression, we have

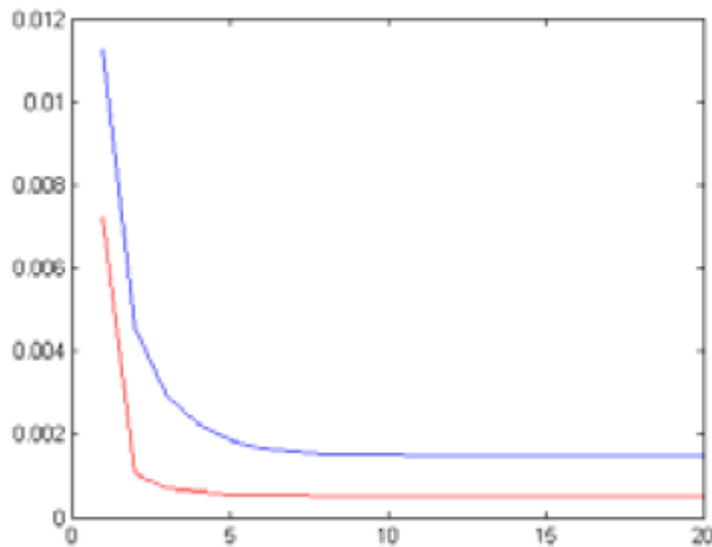
$$\mathbf{w} = (X X^\top)^{-1} X \mathbf{y}$$

◆ Now, for logistic regression

$$\begin{aligned}\mathbf{w}_{t+1} &= \mathbf{w}_t - H^{-1} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}_t) \\ &= \mathbf{w}_t - (X R X^\top)^{-1} X (\boldsymbol{\mu} - \mathbf{y}) \\ &= (X R X^\top)^{-1} \{X R X^\top \mathbf{w}_t - X (\boldsymbol{\mu} - \mathbf{y})\} \\ &= (X R X^\top)^{-1} X R \mathbf{z}\end{aligned}$$

where $\mathbf{z} = X^\top \mathbf{w}_t - R^{-1}(\boldsymbol{\mu} - \mathbf{y})$

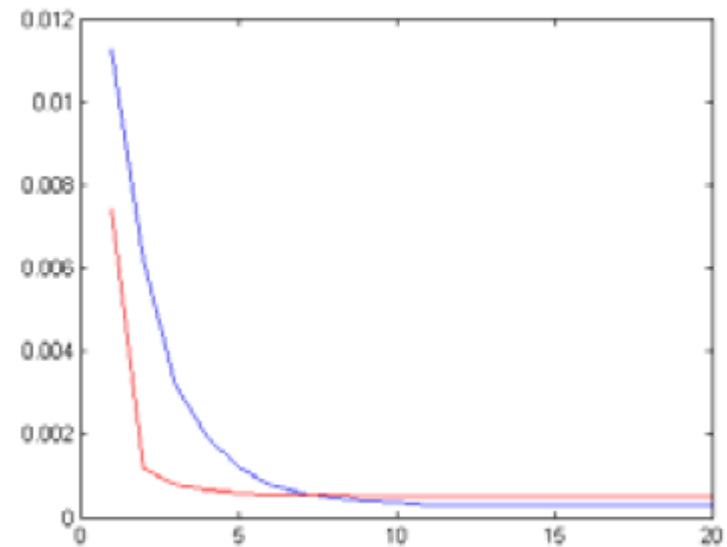
Convergence curves



rec.autos

vs.

rec.sports.baseball



comp.windows.x

vs.

rec.motorcycles

- Legend: X-axis: Iteration #; Y-axis: classification error
- In each figure, red for **IRLS** and blue for **gradient descent**

LR: Practical Issues

- ◆ IRLS takes $O(N + d^3)$ per iteration, where N is # training points and d is feature dimension, but converges in fewer iterations
- ◆ Quasi-Newton methods, that approximate the Hessian, work faster
- ◆ Conjugate gradient takes $O(Nd)$ per iteration, and usually works best in practice
- ◆ Stochastic gradient descent can also be used if N is large c.f. perceptron rule

Gaussian NB vs. Logistic Regression

GNB
Gaussian parameters

VS

LR
Regression parameters

- ◆ Representation equivalence
 - But only in some special case! (GNB with class independent variances)
- ◆ What's the differences?
 - LR makes no assumption about $P(X | Y)$ in learning
 - They optimize different functions, obtain different solutions

Generative vs. Discriminative

◆ Given **infinite data** (asymptotically)

- (1) If conditional independence assumption holds, discriminative and generative NB perform similar

$$\epsilon_{\text{Dis},\infty} \sim \epsilon_{\text{Gen},\infty}$$

- (2) If conditional independence assumption does NOT hold, discriminative outperform generative NB

$$\epsilon_{\text{Dis},\infty} < \epsilon_{\text{Gen},\infty}$$

Generative vs. Discriminative

- ◆ Given **finite data** (N data points, d features)

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

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

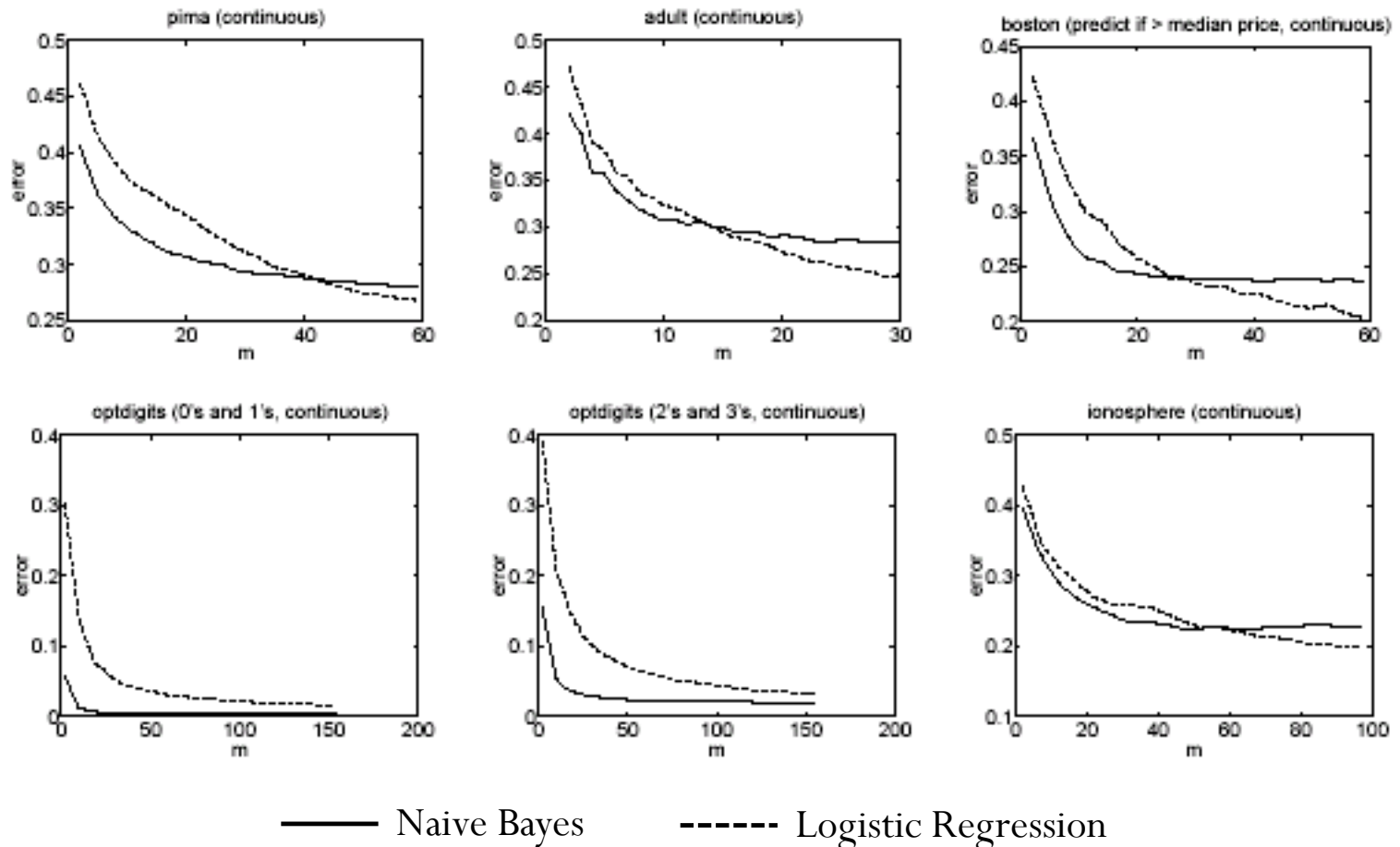
- Naive Bayes (generative) requires $N = O(\log d)$ to converge to its asymptotic error, whereas logistic regression (discriminative) requires $N = O(d)$.

- ◆ **Why?**

- “Independent class conditional densities” – parameter estimates are not coupled, each parameter is learnt independently, not jointly, from training data

Experimental Comparison

- ◆ UCI Machine Learning Repository 15 datasets, 8 continuous features, 7 discrete features



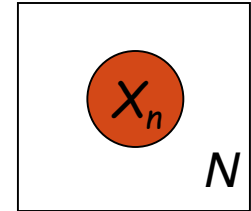
What you need to know

- ◆ LR is a linear classifier
 - Decision boundary is a hyperplane
- ◆ LR is learnt by maximizing conditional likelihood
 - No closed-form solution
 - Concave! Global optimum by gradient ascent methods
- ◆ GNB with class-independent variances representationally equivalent to LR
 - Solutions differ because of objective (loss) functions
- ◆ 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 $P(X | Y)$
- ◆ Convergence rates:
 - GNB (usually) needs less data
 - LR (usually) gets to better solutions in the limit

Exponential family

- ◆ For a numeric random variable \mathbf{X}

$$\begin{aligned} p(\mathbf{x}|\boldsymbol{\eta}) &= h(\mathbf{x}) \exp(\boldsymbol{\eta}^\top T(\mathbf{x}) - A(\boldsymbol{\eta})) \\ &= \frac{1}{Z(\boldsymbol{\eta})} h(\mathbf{x}) \exp(\boldsymbol{\eta}^\top T(\mathbf{x})) \end{aligned}$$



is an **exponential family distribution** with natural (canonical) parameter $\boldsymbol{\eta}$

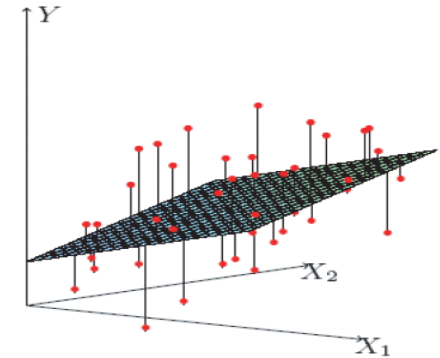
- ◆ Function $T(\mathbf{x})$ is a *sufficient statistic*.
- ◆ Function $A(\boldsymbol{\eta}) = \log Z(\boldsymbol{\eta})$ is the log normalizer.
- ◆ Examples: Bernoulli, multinomial, Gaussian, Poisson, gamma,...

Recall Linear Regression

- ◆ Let us assume that the target variable and the inputs are related by the equation:

$$y_i = \boldsymbol{\theta}^\top \mathbf{x}_i + \epsilon_i$$

white noise



where ϵ is an error term of unmodeled effects or random noise

- ◆ Now assume that ϵ follows a Gaussian $N(0, \sigma)$, then we have:

$$p(y_i | \mathbf{x}_i, \boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left(-\frac{(y_i - \boldsymbol{\theta}^\top \mathbf{x}_i)^2}{2\sigma^2} \right)$$

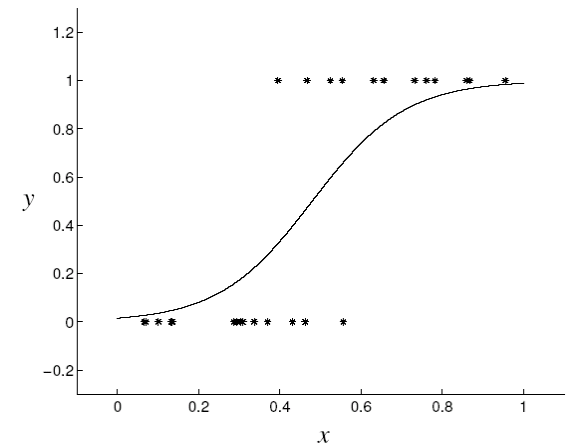
Recall: Logistic Regression (sigmoid classifier)

- ◆ The condition distribution: a Bernoulli

$$p(y|\mathbf{x}) = \mu(\mathbf{x})^y (1 - \mu(\mathbf{x}))^{1-y}$$

where μ is a logistic function

$$\mu(\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^\top \mathbf{x}}}$$



- ◆ We can use the brute-force gradient method as in LR
- ◆ But we can also apply generic laws by observing the $p(y|x)$ is an **exponential family function**, more specifically, a **generalized linear model**!

Example: Multivariate Gaussian Distribution

◆ For a continuous vector random variable $\mathbf{x} \in \mathbb{R}^d$:

$$p(\mathbf{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)$$

$$= \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}\text{tr}(\Sigma^{-1}\mathbf{x}\mathbf{x}^\top) + \boldsymbol{\mu}^\top \Sigma^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}^\top \Sigma^{-1}\boldsymbol{\mu} - \log|\Sigma|\right)$$

Moment parameter

◆ Exponential family representation

$$\boldsymbol{\eta} = \left[\Sigma^{-1}\boldsymbol{\mu}; -\frac{1}{2}\text{vec}(\Sigma^{-1}) \right] = [\boldsymbol{\eta}_1; \text{vec}(\boldsymbol{\eta}_2)], \quad \boldsymbol{\eta}_1 = \Sigma^{-1}\boldsymbol{\mu} \text{ and } \boldsymbol{\eta}_2 = -\frac{1}{2}\Sigma^{-1}$$

Natural parameter

$$T(\mathbf{x}) = [\mathbf{x}; \text{vec}(\mathbf{x}\mathbf{x}^\top)]$$

$$A(\boldsymbol{\eta}) = \frac{1}{2}\boldsymbol{\mu}^\top \Sigma^{-1}\boldsymbol{\mu} + \log|\Sigma| = -\frac{1}{2}\text{tr}(\boldsymbol{\eta}_2\boldsymbol{\eta}_1\boldsymbol{\eta}_1^\top) - \frac{1}{2}\log(-2|\boldsymbol{\eta}_2|)$$

$$h(\mathbf{x}) = (2\pi)^{-d/2}$$

- Note: a d -dimensional Gaussian is a $(d + d^2)$ -parameter distribution with a $(d + d^2)$ -element vector of sufficient statistics (but because of symmetry and positivity, parameters are constrained and have lower degree of freedom)

Example: Multinomial distribution

◆ For a binary vector random variable $\mathbf{x} \sim \text{multi}(\mathbf{x}|\pi)$:

$$\begin{aligned} p(\mathbf{x}|\pi) &= \prod_{i=1}^d \pi_i^{x_i} = \exp \left(\sum_i x_i \ln \pi_i \right) \\ &= \exp \left(\sum_{i=1}^{d-1} x_i \ln \pi_i + \left(1 - \sum_{i=1}^{d-1} x_i \right) \ln \left(1 - \sum_{i=1}^{d-1} \pi_i \right) \right) \\ &= \exp \left(\sum_{i=1}^{d-1} x_i \ln \frac{\pi_i}{1 - \sum_{i=1}^{d-1} \pi_i} + \ln \left(1 - \sum_{i=1}^{d-1} \pi_i \right) \right) \end{aligned}$$

◆ Exponential family representation

$$\boldsymbol{\eta} = [\ln(\pi_i/\pi_d); 0]$$

$$T(\mathbf{x}) = \mathbf{x}$$

$$A(\boldsymbol{\eta}) = -\ln \left(1 - \sum_{i=1}^{d-1} \pi_i \right) = \ln \left(\sum_{i=1}^d e^{\eta_i} \right)$$

$$h(\mathbf{x}) = 1$$

Why exponential family?

- ◆ Moment generating property (proof?)

$$\nabla_{\boldsymbol{\eta}} A(\boldsymbol{\eta}) = \nabla_{\boldsymbol{\eta}} \log Z(\boldsymbol{\eta}) = \cdots = \mathbb{E}_{p(\mathbf{x}|\boldsymbol{\eta})}[T(\mathbf{x})]$$

$$\nabla_{\boldsymbol{\eta}}^2 A(\boldsymbol{\eta}) = \cdots = \text{Var}[T(\mathbf{x})]$$

Moment estimation

- ◆ We can easily compute moments of any exponential family distribution by taking the derivatives of the log normalizer $A(\eta)$.
- ◆ The q^{th} derivative gives the q^{th} centered moment.

$$\nabla_{\eta} A(\eta) = \text{mean}$$

$$\nabla_{\eta}^2 A(\eta) = \text{variance}$$

$$\vdots$$

Moment vs canonical parameters

- ◆ The moment parameter μ can be derived from the natural (canonical) parameter

$$\nabla_{\eta} A(\eta) = \mathbb{E}_{p(\mathbf{x}|\eta)}[T(\mathbf{x})] \triangleq \mu$$

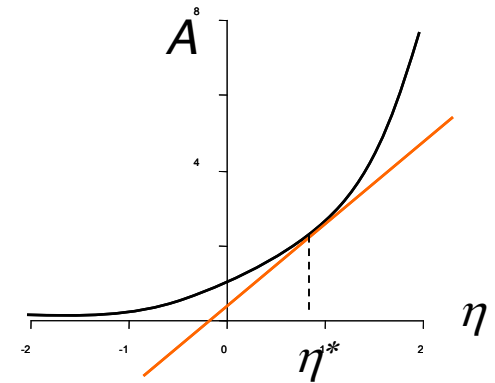
- ◆ $A(\eta)$ is convex since

$$\nabla_{\eta}^2 A(\eta) = \text{Var}[T(\mathbf{x})] > 0$$

- ◆ Hence we can invert the relationship and infer the canonical parameter from the moment parameter (1-to-1):

$$\eta \triangleq \psi(\mu)$$

- A distribution in the exponential family can be parameterized not only by η – the canonical parameterization, but also by μ – the moment parameterization.



IID Sampling for Exponential Family

- ◆ For exponential family distribution, we can obtain the sufficient statistics by inspection once represented in the standard form

$$p(\mathbf{x}|\boldsymbol{\eta}) = h(\mathbf{x}) \exp(\boldsymbol{\eta}^\top T(\mathbf{x}) - A(\boldsymbol{\eta}))$$

- Sufficient statistics: $T(\mathbf{x})$

- ◆ For IID sampling, the joint distribution is also an exponential family

$$\begin{aligned} p(D|\boldsymbol{\eta}) &= \prod_i h(\mathbf{x}_i) \exp(\boldsymbol{\eta}^\top T(\mathbf{x}_i) - A(\boldsymbol{\eta})) \\ &= \left(\prod_i h(\mathbf{x}_i) \right) \exp \left(\boldsymbol{\eta}^\top \sum_i T(\mathbf{x}_i) - N A(\boldsymbol{\eta}) \right) \end{aligned}$$

- Sufficient statistics: $\sum_i T(\mathbf{x}_i)$

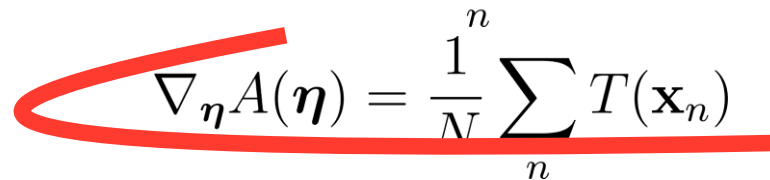
MLE for Exponential Family

- ◆ For *iid* data, the log-likelihood is

$$\mathcal{L}(\boldsymbol{\eta}; D) = \sum_n \log h(\mathbf{x}_n) + \left(\boldsymbol{\eta}^\top \sum_n T(\mathbf{x}_n) \right) - NA(\boldsymbol{\eta})$$

- ◆ Take derivatives and set to zero:

$$\nabla_{\boldsymbol{\eta}} \mathcal{L}(\boldsymbol{\eta}; D) = \sum_n T(\mathbf{x}_n) - N \nabla_{\boldsymbol{\eta}} A(\boldsymbol{\eta}) = 0$$


$$\nabla_{\boldsymbol{\eta}} A(\boldsymbol{\eta}) = \frac{1}{N} \sum_n T(\mathbf{x}_n)$$

$$\hat{\boldsymbol{\mu}}_{MLE} = \frac{1}{N} \sum_n T(\mathbf{x}_n) \quad \text{Only involve sufficient statistics!}$$

- ◆ This amounts to **moment matching**.
- ◆ We can infer the canonical parameters using $\hat{\boldsymbol{\eta}}_{MLE} = \psi(\hat{\boldsymbol{\mu}}_{MLE})$

Examples

◆ Gaussian: $\boldsymbol{\eta} = \left[\Sigma^{-1} \boldsymbol{\mu}; -\frac{1}{2} \text{vec}(\Sigma^{-1}) \right]$

$$T(\mathbf{x}) = [\mathbf{x}; \text{vec}(\mathbf{x}\mathbf{x}^\top)]$$

$$A(\boldsymbol{\eta}) = \frac{1}{2} \boldsymbol{\mu}^\top \Sigma^{-1} \boldsymbol{\mu} + \log |\Sigma|$$

$$h(\mathbf{x}) = (2\pi)^{-d/2}$$

$$\Rightarrow \hat{\boldsymbol{\mu}}_{MLE} = \frac{1}{N} \sum_n T_1(\mathbf{x}_n) = \frac{1}{N} \sum_n \mathbf{x}_n$$

◆ Multinomial:

$$\boldsymbol{\eta} = [\ln(\pi_i/\pi_d); 0]$$

$$T(\mathbf{x}) = \mathbf{x}$$

$$A(\boldsymbol{\eta}) = -\ln \left(1 - \sum_{i=1}^{d-1} \pi_i \right)$$

$$h(\mathbf{x}) = 1$$

$$\Rightarrow \hat{\boldsymbol{\mu}}_{MLE} = \frac{1}{N} \sum_n \mathbf{x}_n$$

◆ Poisson: $\eta = \log \lambda$

$$T(x) = x$$

$$A(\eta) = \lambda = e^\eta$$

$$h(x) = \frac{1}{x!}$$

$$\Rightarrow \hat{\mu}_{MLE} = \frac{1}{N} \sum_n x_n$$

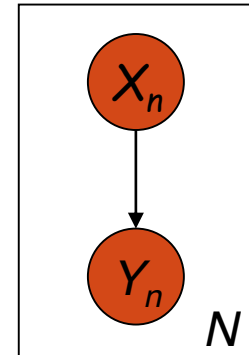
Generalized Linear Models (GLIMs)

◆ The graphical model

- Linear regression
- Discriminative linear classification
- Commonality:

$$\text{model} \quad \mathbb{E}_p[y] = \mu = f(\boldsymbol{\theta}^\top \mathbf{x})$$

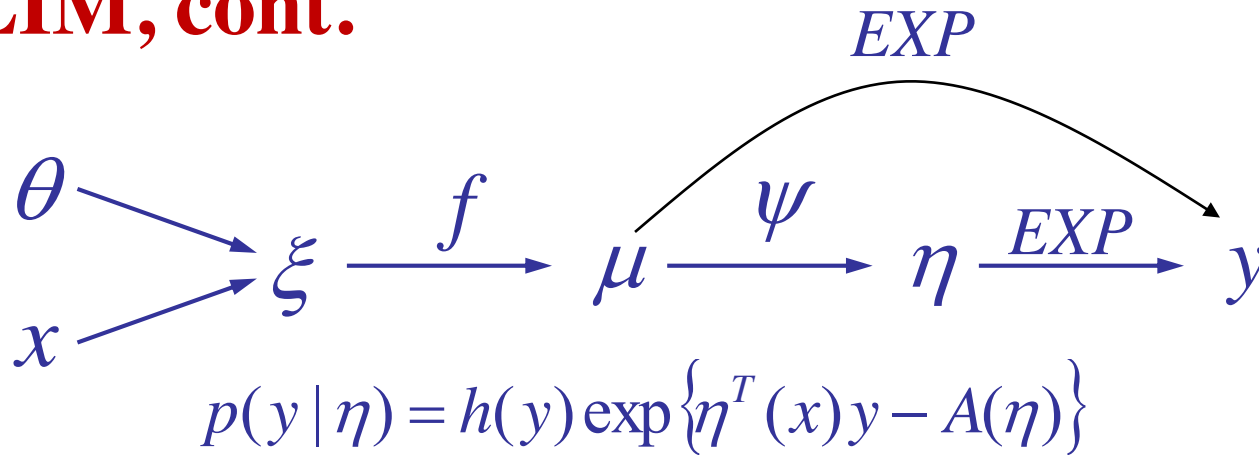
- What is $p()$? the cond. dist. of Y .
- What is $f()$? the response function.



◆ GLIM

- The observed input \mathbf{x} is assumed to enter into the model via a linear combination of its elements $\xi = \boldsymbol{\theta}^\top \mathbf{x}$
- The conditional mean μ is represented as a function $f(\xi)$ of ξ , where f is known as the response function
- The observed output y is assumed to be characterized by an exponential family distribution with conditional mean μ .

GLIM, cont.



$$\Rightarrow p(y | \eta, \phi) = h(y, \phi) \exp \left\{ \frac{1}{\phi} \left(\eta^T (x) y - A(\eta) \right) \right\}$$

◆ The choice of exp family is constrained by the nature of the data \mathcal{Y}

- Example:
 - y is a continuous vector \rightarrow multivariate Gaussian
 - y is a class label \rightarrow Bernoulli or multinomial

◆ The choice of the response function

- Following some mild constraints, e.g., $[0, 1]$. Positivity ...
- Canonical response function:
 - In this case $\theta^T \mathbf{x}$ directly corresponds to canonical parameter η .

$$\mathbf{f} = \psi^{-1}(\cdot)$$

MLE for GLIMs

◆ Log-likelihood

$$\mathcal{L}(\boldsymbol{\theta}; D) = \sum_n \log h(y_n) + \sum_n (\eta_n y_n - A(\eta_n))$$

where $\eta_n = \psi(\mu_n)$, $\mu_n = f(\xi_n)$ and $\xi_n = \boldsymbol{\theta}^\top \mathbf{x}_n$

◆ Derivative of Log-likelihood

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \mathcal{L} &= \sum_n \left(y_n \nabla_{\boldsymbol{\theta}} \eta_n - \frac{dA(\eta_n)}{d\eta_n} \nabla_{\boldsymbol{\theta}} \eta_n \right) \\ &= \sum_n (y_n - \mu_n) \nabla_{\boldsymbol{\theta}} \eta_n\end{aligned}$$

This is a fixed point function
because μ is a function of θ

MLE for GLIMs with canonical response

◆ Log-likelihood

$$\mathcal{L}(\boldsymbol{\theta}; D) = \sum_n \log h(y_n) + \sum_n (\boldsymbol{\theta}^\top \mathbf{x}_n y_n - A(\eta_n))$$

◆ Derivative of Log-likelihood

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \mathcal{L} &= \sum_n \left(\mathbf{x}_n y_n - \frac{dA(\eta_n)}{d\eta_n} \nabla_{\boldsymbol{\theta}} \eta_n \right) \\ &= \sum_n (y_n - \mu_n) \mathbf{x}_n \\ &= X(\mathbf{y} - \boldsymbol{\mu})\end{aligned}$$

This is a fixed point function
because μ is a function of θ

◆ Online learning for canonical GLIMs

- Stochastic gradient ascent = least mean squares (LMS) algorithm:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \rho(y_n - \mu_n^t) \mathbf{x}_n$$

where $\mu_n^t = f(\boldsymbol{\theta}_t^\top \mathbf{x}_n)$ and ρ is a step size

MLE for GLIMs with canonical response

◆ Log-likelihood

$$\mathcal{L}(\boldsymbol{\theta}; D) = \sum_n \log h(y_n) + \sum_n (\boldsymbol{\theta}^\top \mathbf{x}_n y_n - A(\eta_n))$$

◆ Derivative of Log-likelihood

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \mathcal{L} &= \sum_n \left(\mathbf{x}_n y_n - \frac{dA(\eta_n)}{d\eta_n} \nabla_{\boldsymbol{\theta}} \eta_n \right) \\ &= \sum_n (y_n - \mu_n) \mathbf{x}_n \\ &= X(\mathbf{y} - \boldsymbol{\mu})\end{aligned}$$

This is a fixed point function
because $\boldsymbol{\mu}$ is a function of $\boldsymbol{\theta}$

◆ Batch learning applies

- E.g., the Newton's method leads to an Iteratively Reweighted Least Square (IRLS) algorithm

What you need to know

- ◆ Exponential family distribution
- ◆ Moment estimation
- ◆ Generalized linear models
- ◆ Parameter estimation of GLIMs