EECS 545: Machine Learning

Lecture 4. Linear models of classification

Honglak Lee

1/19/2011

Announcement

- Ctools: Forums - general discussions
  – Find your study and project groups
- Ctools: Forums - FAQ
  – We will post answers to frequently asked questions about homework problems
- Project information (TBD)
  – We will post about project guidelines, dates, datasets, topics.

Outline

- Recap: Regression & Probability
- Linear models of classification
- Discriminant functions (read Bishop book)
- Probabilisitic discriminative models
  - Logistic regression
  - Softmax regression

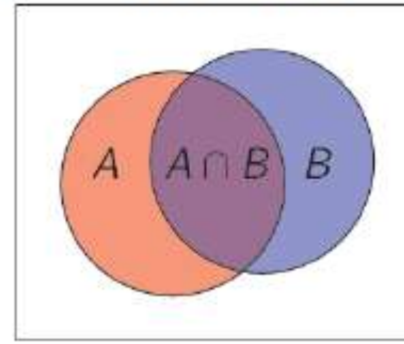
Recap: probability

Recap: Axioms of Probability

- $P(A) \geq 0 \ \forall A \in F$
- $P(\Omega) = 1$
- If A_1, A_2, ...are disjoint events, then

$$P(\cup_i A_i) = \sum_i P(A_i)$$

Additional Properties of Probability



- If $A \subseteq B \implies P(A) \leq P(B)$.
- $P(A \cap B) \leq \min(P(A), P(B))$.
- (Union Bound) $P(A \cup B) \leq P(A) + P(B)$.
- $P(\Omega \setminus A) = 1 - P(A)$.
- (Law of Total Probability) If $A_1, \ldots, A_k$ are a set of disjoint events such that $\cup_{i=1}^{k} A_i = \Omega$, then

$$\sum_{i=1}^{k} P(A_k) = 1.$$

Recap: Bayes' Rule

Using the chain rule we may see:

$$P(A|B)P(B) = P(A \cap B) = P(B|A)P(A)$$

Rearranging this yields **Bayes' rule**:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

Often this is written as:

$$P(B_i|A) = \frac{P(A|B_i)P(B_i)}{\sum_i P(A|B_i)P(B_i)}$$

Where B_i are a partition of Ω (note the bottom is just the law of total probability).

Recap: Likelihood Functions

- Why is Bayes' so useful in learning? Allows us to evaluate a parameter setting w:

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}$$

$$p(D) = \sum_{w} p(D|w)p(w)$$

- The likelihood function, $p(D|w)$, is evaluated for observed data D as a function of w. It expresses how probable the observed data set is for various parameter settings w.
- Bayes' in words: posterior $\propto$ likelihood $\times$ prior

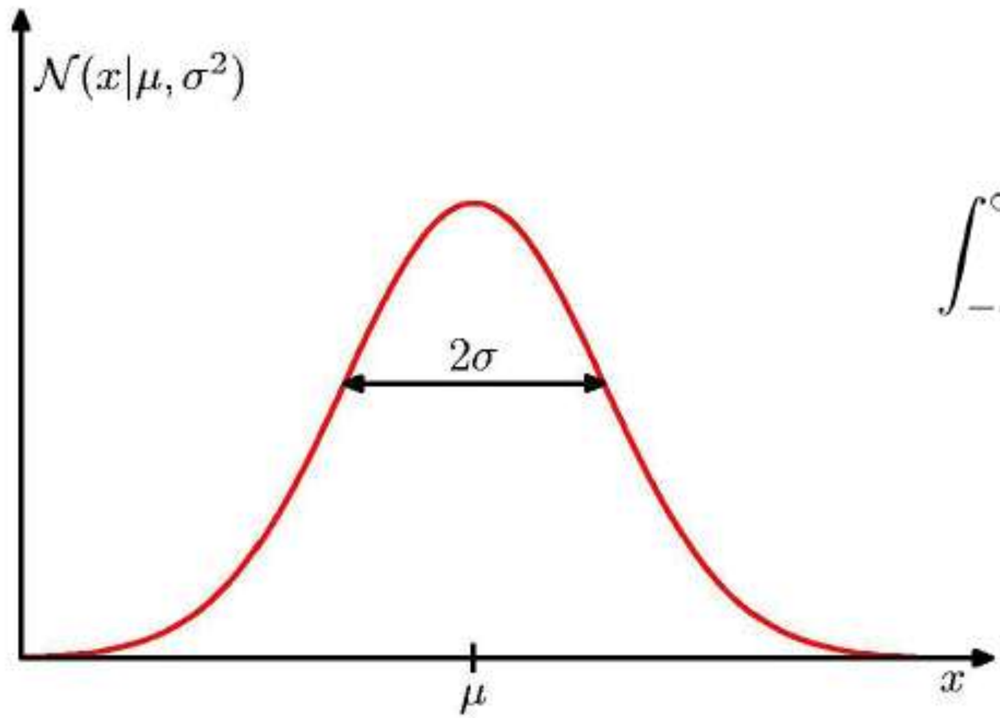
Recap: Maximum Likelihood

- Maximum likelihood:
  - choose parameter setting w that maximizes likelihood function $p(D|w)$.
  - Choose the value of w that maximizes the probability of observed data.
  - The negative log of the likelihood is called an error function.
  - Because negative logarithm is a monotonically decreasing function, maximizing likelihood is equivalent to minimizing the error.

Maximum Likelihood interpretation of least squares regression

Recap: The Gaussian Distribution

$$\mathcal{N}\left(x|\mu,\sigma^2\right) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

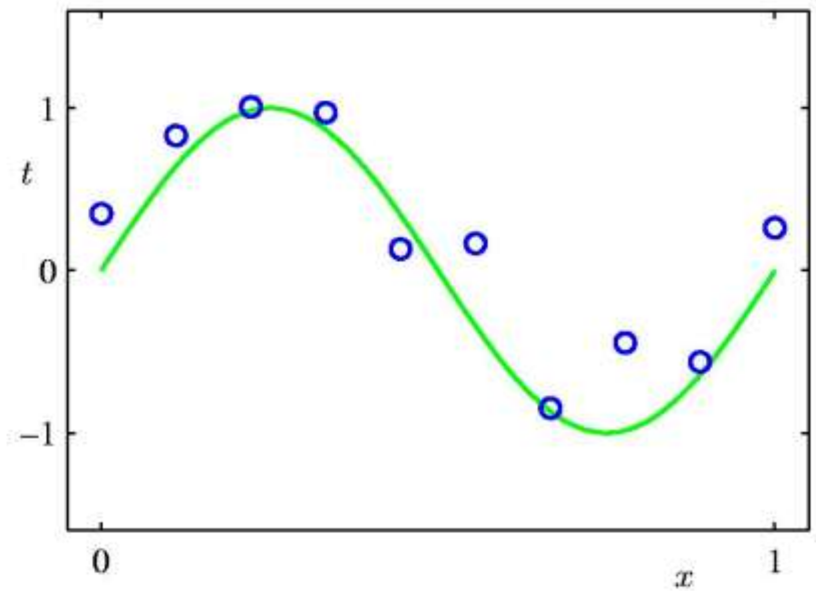


$$\mathcal{N}(x|\mu,\sigma^2) > 0$$

$$\int_{-\infty}^{\infty} \mathcal{N}\left(x|\mu,\sigma^2\right) \mathrm{d}x = 1$$

Regression

- Given a set of observations
  - $\mathbf{x} = \{ x_1 \, . \, . \, . \, x_N \}$
- And corresponding target values:
  - $\mathbf{t} = \{ t_1 \, . \, . \, . \, t_N \}$

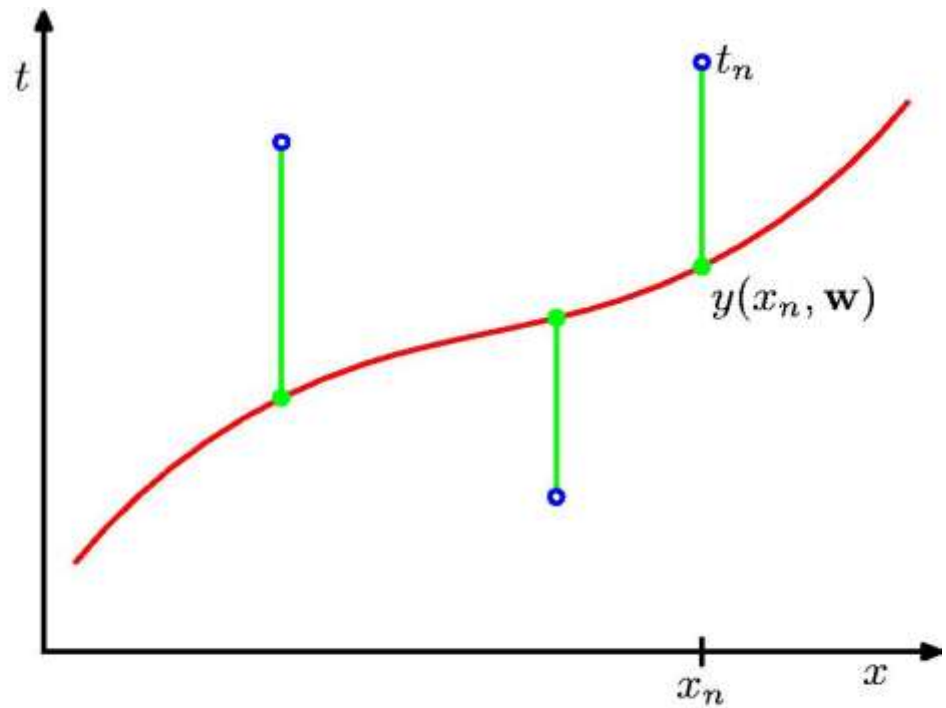


- We want to learn a function $y(x,w)=t$ to predict future values.

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Slide credit: Ben Kuipers

Sum-of-Squares Error Function



$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2$$

Maximum Likelihood w

- Assume a stochastic model:

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon \text{ where } \epsilon \sim \mathcal{N}(0, \beta^{-1})$$

- This gives a likelihood function:

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1})$$

- With inputs X={x_1 … x_N} and target values t={t_1 … t_N}, the data likelihood is:

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^T \phi(\mathbf{x}_n), \beta^{-1})$$

Slide credit: Ben Kuipers

Log Likelihood

- Given data likelihood

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^T \phi(\mathbf{x}_n), \beta^{-1})$$

- Log likelihood is:

$$\ln p(\mathbf{t}|\mathbf{w}, \beta) = \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(\mathbf{w})$$

- where:

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{t_n - \mathbf{w}^T \phi(x_n)\}^2$$

  - (Note: Bishop drops **X** from the notation.)

Slide credit: Ben Kuipers

Details of derivation

From $P(t|\mathbf{x}, \mathbf{w}) = \sqrt{\frac{\beta}{2\pi}} \exp(-\beta||t - \mathbf{w}^T\phi(x)||^2)$

We have:

$$\begin{aligned}
& \log P(t_1, ..., t_N|\mathbf{x}, \mathbf{w}) \\
= \quad & \log \prod_{i=1}^{N} \mathcal{N}(t_i, \mathbf{w}^T\phi(\mathbf{x}^{(i)}) \\
= \quad & \sum_{i=1}^{N} \log \left(\sqrt{\frac{\beta}{2\pi}} \exp(-\frac{\beta}{2}||t^{(i)} - \mathbf{w}^T\phi(\mathbf{x}^{(i)})||^2) \right) \\
= \quad & \sum_{i=1}^{N} \left(\frac{1}{2}\log\beta - \frac{1}{2}\log 2\pi - \frac{\beta}{2}||t^{(i)} - \mathbf{w}^T\phi(\mathbf{x}^{(i)})||^2 \right) \\
= \quad & \frac{N}{2}\log\beta - \frac{N}{2}\log 2\pi - \sum_{i=1}^{N} \frac{\beta}{2}||t^{(i)} - \mathbf{w}^T\phi(\mathbf{x}^{(i)})||^2
\end{aligned}$$

Classification

Classification

- The task of classification:
  - Given an input vector **x**, assign it to one of K distinct classes C_k where $k = 1, \ldots K$
- Representing the assignment:
  - For K=2
    - Let t=1 mean that **x** is in C_1.
    - Let t=0 mean that **x** is in C_2.
  - For K>2,
    - Use 1-of-K coding, e.g., $\mathbf{t} = (0, 1, 0, 0, 0)^T$
      - (This would also work for K=2, of course.)

Learning the Classifier

- From input vectors x = $\{x_1, \ldots x_N\}$
  - and corresponding target values $\mathbf{t} = \{t_1, \ldots t_N\}$.
- 1. Discriminant functions

  Learn a function $y(\mathbf{x})$ that maps $\mathbf{x}$ onto some C_j.
- 2. Learn the distributions $p(C_k \mid \mathrm{x})$.

  (a) Learn model parameters from the training set.

  Discriminative models

  (b) Learn class densities $p(x \mid C_k)$ and priors $p(C_k)$

  Generative models

Discriminant functions

Discriminating two classes

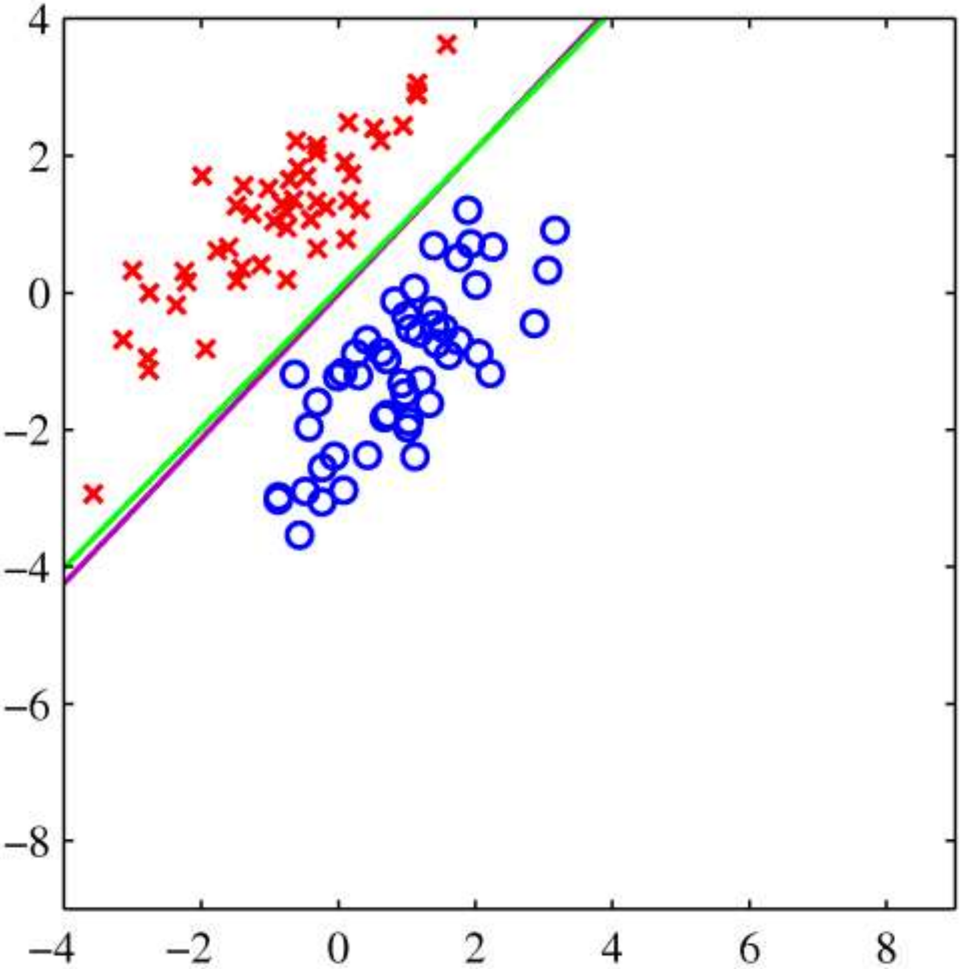
- Specify a weight vector w and a bias w_0.

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

- Assign x to C_1 if

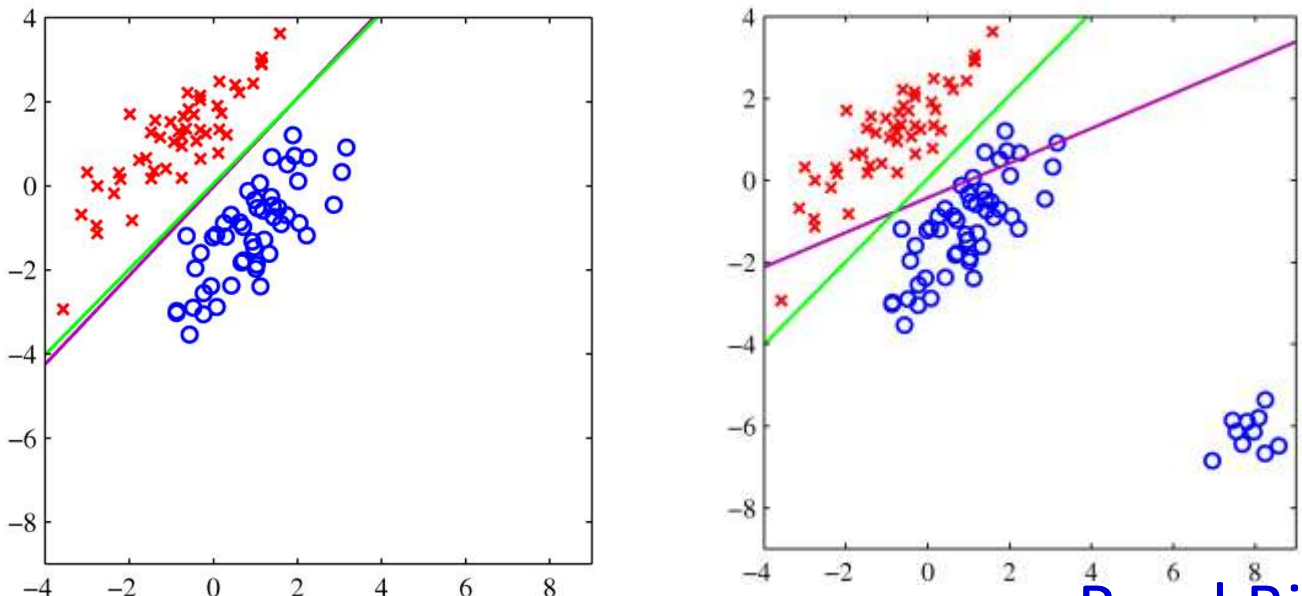
$$y(\mathbf{x}) \geq 0$$

  - and to C_0 otherwise.
- How to pick w?



How do we set the weights w?

- How about w that minimizes squared error?
    - (Predicted **t** versus actual, e.g., **t** = (0, 1, 0, 0, 0).)
  – Least squares is too sensitive to outliers.



Read Bishop book

Fisher's Linear Discriminant

- Use w to project x to one dimension.

$$\text{if } \mathbf{w}^T\mathbf{x} \geq -w_0 \text{ then } C_1 \text{ else } C_0$$

- Select w that best separates the classes.

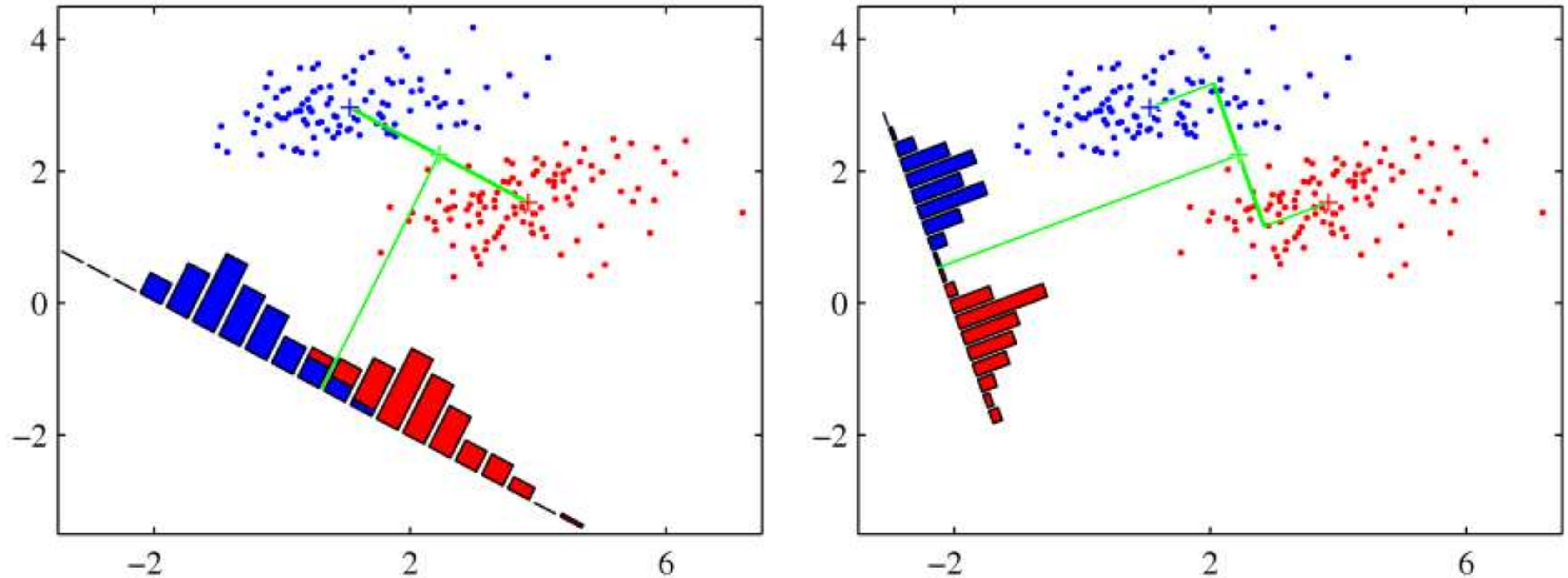
- What does that mean? Simultaneously,
  - Maximize class separation
  - Minimize class variances

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

Read Bishop book

Fisher's Linear Discriminant

- Maximizing separation alone doesn't work.
  - Minimizing class variance is a big help.



Read Bishop book

Objective function

- We want to maximize the "distance between classes"

$$m_2 - m_1 \equiv \mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)$$

- While minimizing the "distance within each class"

$$s_1^2 + s_2^2 \equiv \sum_{n \in C_1} (\mathbf{w}_1^T \mathbf{x}_n - m_k)^2 + \sum_{n \in C_2} (\mathbf{w}_2^T \mathbf{x}_n - m_k)^2$$

- Objective function: $$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

- Can be solved via eigenvalue problem.

Read Bishop book

The Perceptron

- A "generalized linear function"

$$y(\mathbf{x}) = f(\mathbf{w}^T \phi(\mathbf{x}))$$

- Where

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

- Uses target code: t=+1 for C_1, t=-1 for C_2.
- Means we always want:

$$\mathbf{w}^T \phi(\mathbf{x}_n) t_n > 0$$

The Perceptron Criterion

- Only count errors from misclassified points:

$$E_P(\mathbf{w}) = -\sum_{\mathbf{x}_n \in \mathcal{M}} \mathbf{w}^T \phi(\mathbf{x}_n) t_n$$

  – where *M* are the misclassified points.

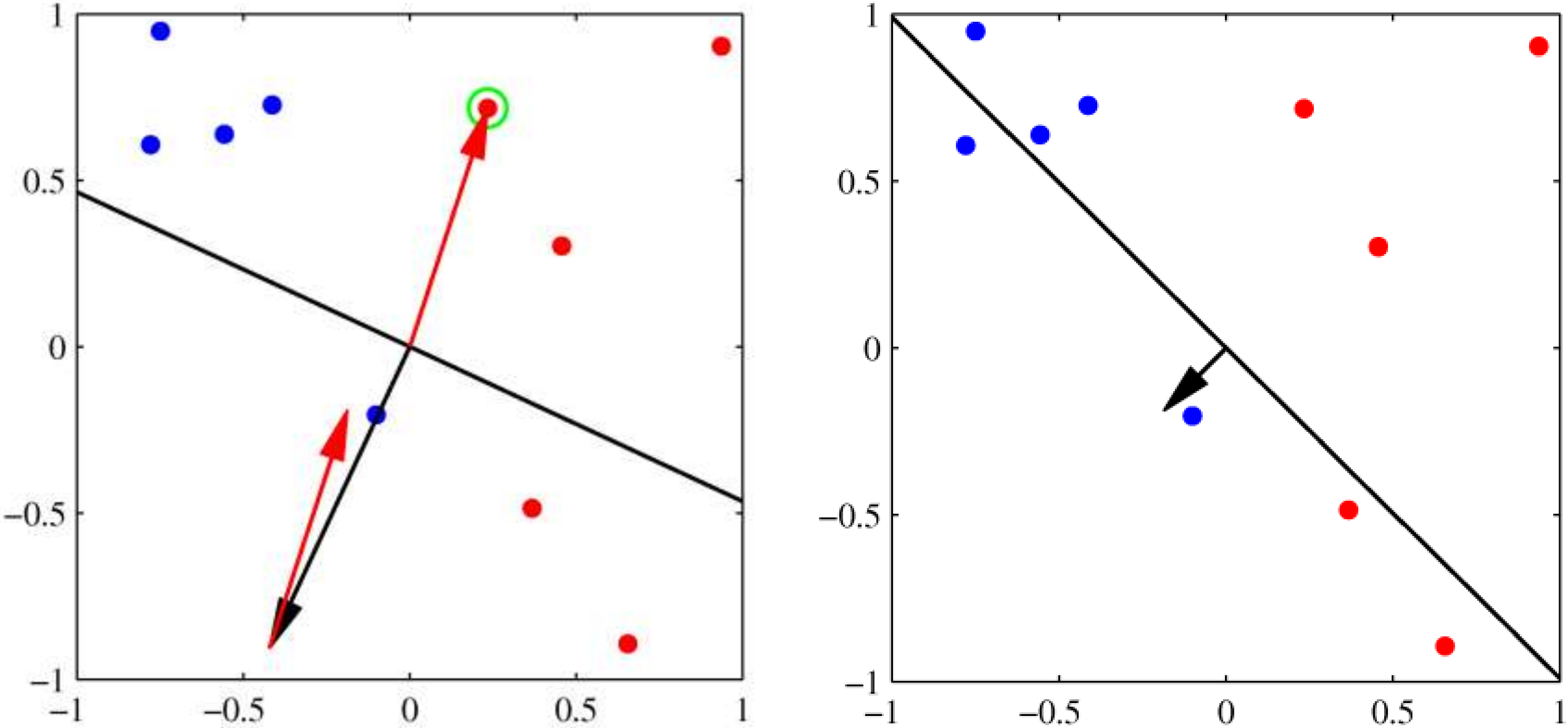
- Stochastic gradient descent:
  – Update the weight vector according to the misclassified points:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} - \eta \phi(\mathbf{x}_n) t_n$$

Note: update only for misclassified examples

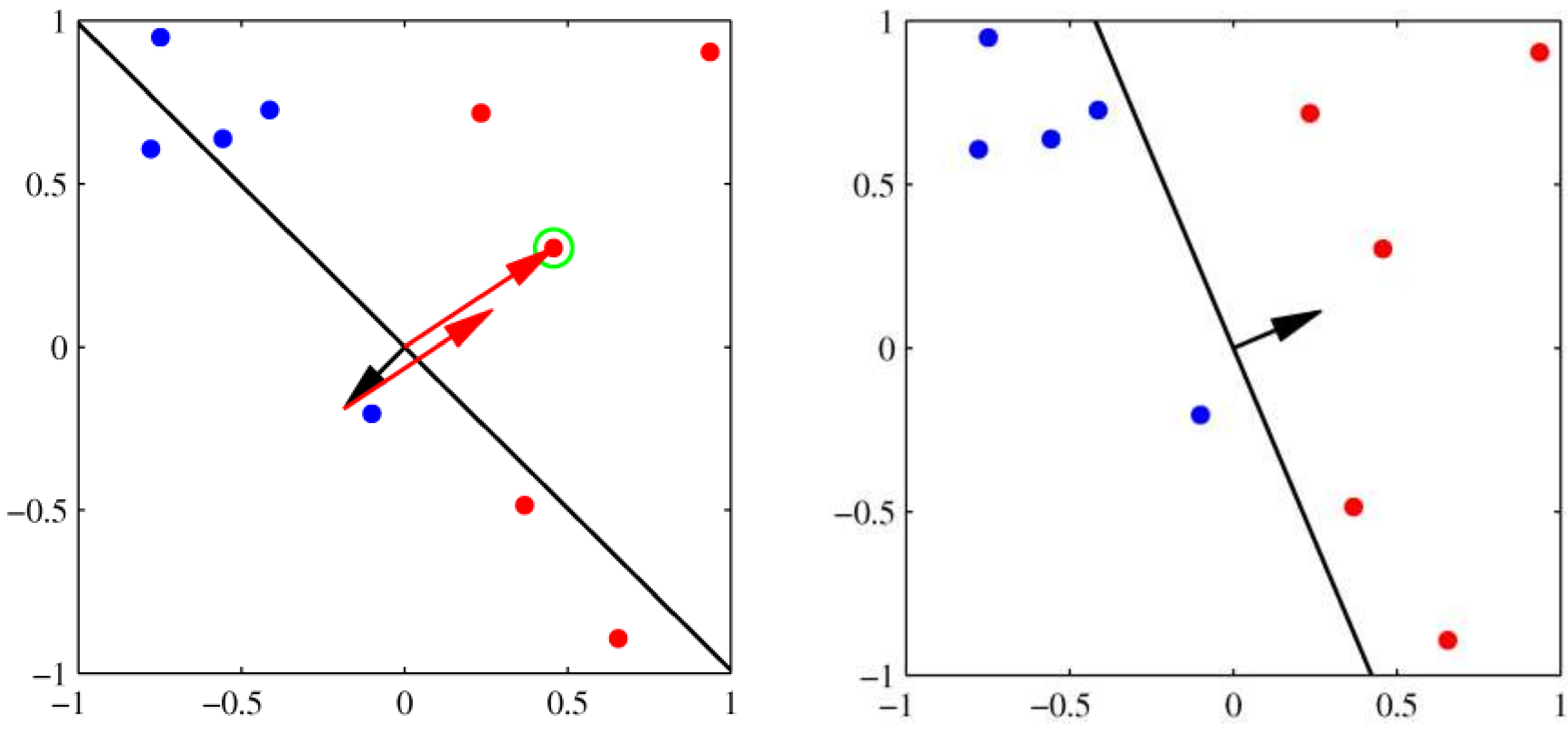
Perceptron Learning (1)

- If x_n is misclassified, add $\phi(\mathbf{x}_n)$ into w.



Perceptron Learning (2)

- If x_n is misclassified, add $\phi(\mathbf{x}_n)$ into w.



Perceptron Learning

- The Perceptron Convergence Theorem says
  - If there exists an exact solution
    - (i.e., if the training data is linearly separable)
  - then the learning algorithm will find it
    - In a finite number of steps.
- But:
    - It can be very slow.
    - If no solution, it won't converge, or stop.
    - Does not generalize well to K>2 classes.
    - Can't express many important concepts.

Probabilistic discriminative models: logistic regression

Logistic regression

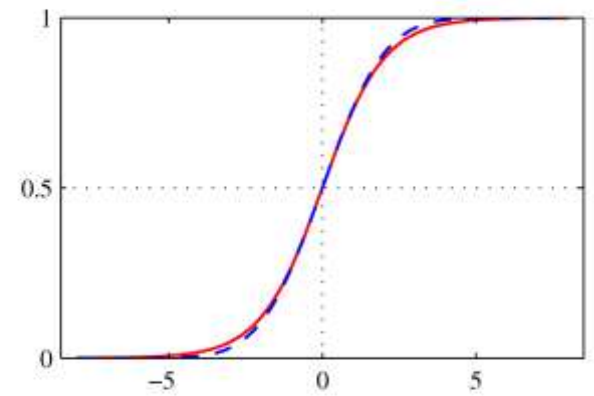
Main idea of probabilistic discriminative models

- Model decision boundary as a function of input x
  - Learn P(Ck|x) over data (e.g., maximum likelihood)
  - Directly predict class labels from inputs
- Next class: we will cover probabilistic generative models
  - Learn P(Ck,x) over data (maximum likelihood) and then use Bayes’ rule to predict P(Ck|x)

Sigmoid and Logit functions

- The *logistic sigmoid* function is:

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$



- Its inverse is the *logit* function:

$$a = \ln\left(\frac{\sigma}{1-\sigma}\right)$$

- Generalizes to *normalized exponential*, or *softmax*.

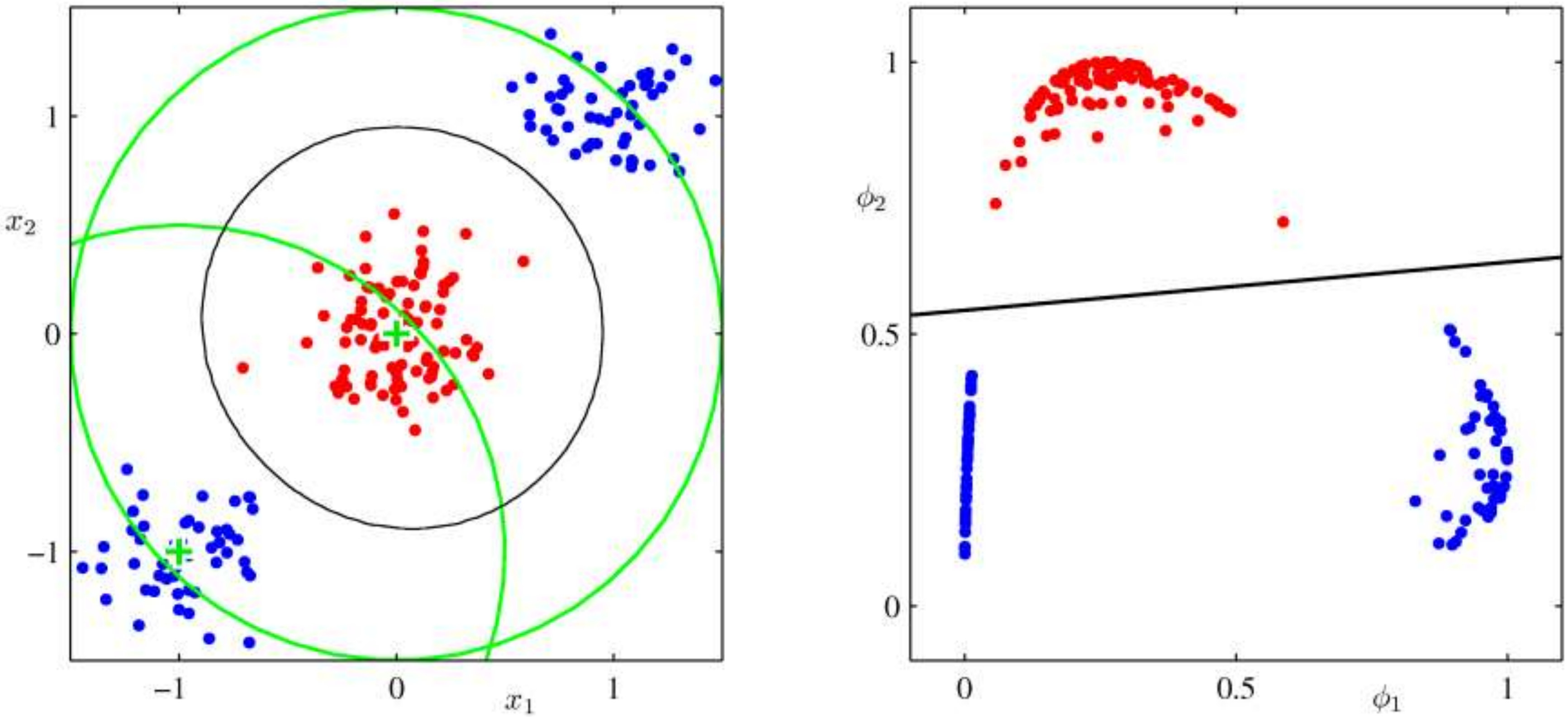
$$p_i = \frac{\exp(q_i)}{\sum_j \exp(q_j)}$$

Fixed Basis Functions

- Instead of working directly with the input x, we may apply to the input a fixed non-linear transformation to a vector $\phi(\mathbf{x})$
- This can make non-separable classes into linearly separable classes.
  – But it can’t eliminate overlap between classes.
  – And the non-linear transformation is fixed.

Making classes separable

- The effect of Gaussian kernel functions.



Likelihood function

- Depending on the label y, the likelihood of x is defined as:

$$\begin{aligned} P(t=1|x,w) &= \sigma(w^T\phi(x)) \\ P(t=0|x,w) &= 1-\sigma(w^T\phi(x)) \end{aligned}$$

- Therefore:

$$P(t|x,w) = \sigma(w^T\phi(x))^y \left(1-\sigma(w^T\phi(x))\right)^{1-y}$$

- Likelihood of data: $\{\langle\phi(\mathbf{x}_n), t_n\rangle\}$ where $t_n \in \{0,1\}$

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}$$

Logistic Regression

- For a data set $\{\langle\phi(\mathbf{x}_n), t_n\rangle\}$ where $t_n \in \{0,1\}$
- the likelihood function is

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}$$

- where

$$y_n = p(C_1|\phi(\mathbf{x}_n)) = \sigma(\mathbf{w}^T\phi(\mathbf{x}_n))$$

- Define an error function $E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w})$
  - (Minimizing *E*(**w**) maximizes likelihood.)

Derivation

- Taking log: $\log P(t|w) = \sum_{n=1}^{N} t_n \log y_n + (1-t_n)\log(1-y_n)$
- Gradient (matrix calculus)

$$\sigma_n \equiv \sigma(\mathbf{w}^T\phi(\mathbf{x}_n))$$

$$\begin{aligned}
&\nabla_{\mathbf{w}} \log P(\mathbf{t}|\mathbf{w}) \\
= &\sum_{n=1}^{N} \nabla_{\mathbf{w}} \left(t_n \log \sigma(\mathbf{w}^T\phi(\mathbf{x}_n)) + (1-t_n)\log(1-\sigma(\mathbf{w}^T\phi(\mathbf{x}_n)))\right) \\
= &\sum_{n=1}^{N} \left(t_n \frac{\sigma_n(1-\sigma_n)}{\sigma_n} - (1-t_n)\frac{\sigma_n(1-\sigma_n)}{1-\sigma_n}\right) \nabla_{\mathbf{w}}(\mathbf{w}^T\phi(\mathbf{x}_n)) \\
= &\sum_{n=1}^{N} t_n(1-\sigma_n)\nabla_{\mathbf{w}}(\mathbf{w}^T\phi(\mathbf{x}_n)) - (1-t_n)\sigma_n\nabla_{\mathbf{w}}(\mathbf{w}^T\phi(\mathbf{x}_n)) \\
= &\sum_{n=1}^{N} (t_n - \sigma_n)\phi(\mathbf{x}_n)
\end{aligned}$$

Logistic Regression: gradient descent

- Taking the gradient of *E*(w) gives us

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\phi(\mathbf{x}_n)$$

  – Recall

$$y_n = p(C_1|\phi(\mathbf{x}_n)) = \sigma(\mathbf{w}^T\phi(\mathbf{x}_n))$$

    - This is essentially the same gradient expression that appeared in linear regression with least-squares.
    - Note the error term between model prediction and target value: $\sigma(\mathbf{w}^T\phi(\mathbf{x}_n)) - t_n$

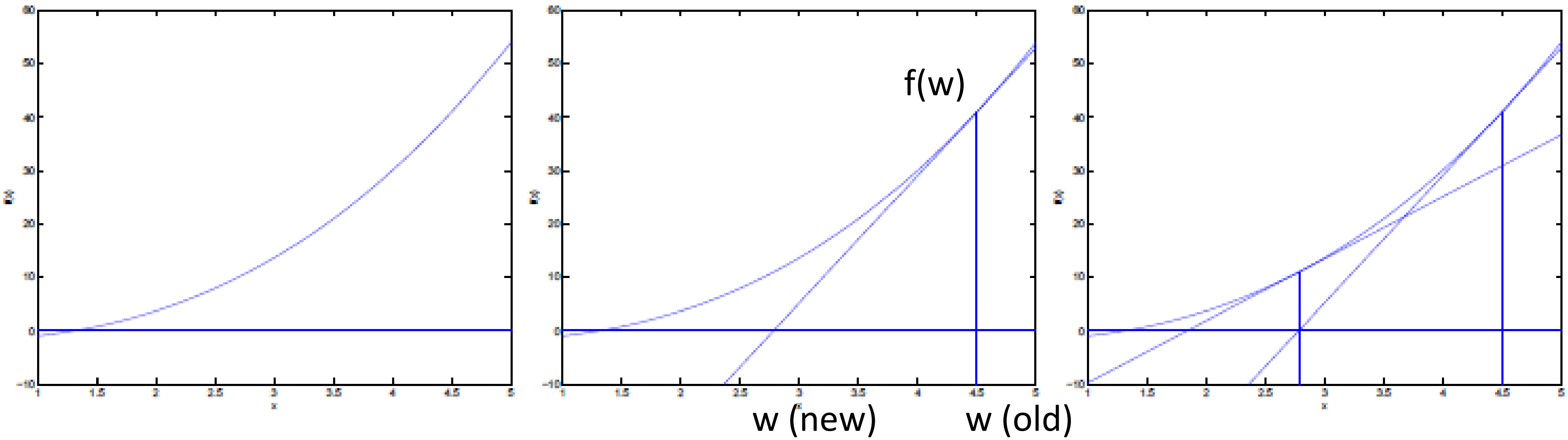
Newton's method

- Goal: Minimizing a general function $l(w)$ (one dimensional case)
  - Approach: solve for $f(w) = \frac{\partial l(w)}{\partial w} = 0$
  - So, how to solve this problem?
- Newton's method:
  - Repeat until convergence:

$$w := w - \frac{f(w)}{f'(w)}$$

Newton's method

- Interatively solve until we get $f(w) = 0$.



- Geometric intuition

$$w := w - \frac{f(w)}{f'(w)}$$

Current value

"Slope"

Newton's method

- Convering $l'(w) = f(w)$
  - Repeat until convergence:

$$w := w - \frac{l'(w)}{l'(w)}$$

- This method can be also extended for multivariate case:

$$w := w - H^{-1} \nabla_w l$$

where H is a Hessian matrix

$$H_{ij}(w) = \frac{\partial^2 l(w)}{\partial w_i \partial w_j}$$

Note: We already did this for least squares problem!

Logistic Regression

- For linear regression, least-squares has a closed-form solution:

$$\mathbf{w}_{ML} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{t}$$

- Generalizes to weighted-least-squares with an *N*x*N* diagonal weight matrix R.

$$\mathbf{w}_{WLS} = (\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{R}\mathbf{t}$$

- But, because $\nabla E(\mathbf{w}) = 0$ is non-linear,
- there is no exact solution. Must iterate.

Iterative Solution

- Apply Newton-Raphson method to iterate to a solution w to $\nabla E(\mathbf{w}) = 0$
- This involves least-squares with weights R:

$$R_{nn} = y_n(1 - y_n)$$

- Since R depends on w (and vice versa), we get *iterative reweighted least squares* (IRLS)

$$\mathbf{w}^{(new)} = (\mathbf{\Phi}^T \mathbf{R} \mathbf{\Phi})^{-1} \mathbf{\Phi}^T \mathbf{R} \mathbf{z}$$

  – where

$$\mathbf{z} = \mathbf{\Phi} \mathbf{w}^{(old)} - \mathbf{R}^{-1}(\mathbf{y} - \mathbf{t})$$

Bayesian Logistic Regression

- Possible, but computationally intractable.
  - Likewise the predictive distribution.
- The Laplace Approximation is helpful.
  - Given a distribution $p(z)$, take the Taylor series of $\ln p(z)$, at a point (the mode) where the linear term vanishes.
  - Use the quadratic term to define a Gaussian.

Exponential family distributions

Motivation

- We considered a binary classification problem where P(y|x) is a Bernoulli distribution
- We are interested in more general distribution
  - E.g., integer variables $y \in \{0,1,2,\dots,\infty\}$
  - E.g., multinomial variables $y \in \{0,1,2,\dots,K\}$
  - Q. is there a general way of parameterizing these distributions?
- Approach: exponential family distribution

Exponential family distribution

- Exponential family distribution

$$p(x|\eta) \quad = \quad h(\mathbf{x})g(\eta)\exp(\eta^T\mathbf{u}(\mathbf{x}))$$

  - η: natural parameters
  - x: data
  - u(x): sufficient statistic

The Exponential Family

- Distribution

$$p(\mathbf{x}|\boldsymbol{\eta}) = h(\mathbf{x})g(\boldsymbol{\eta})\exp\left\{\boldsymbol{\eta}^{\mathrm{T}}\mathbf{u}(\mathbf{x})\right\}$$

- where η is the *natural parameter* and

$$g(\boldsymbol{\eta})\int h(\mathbf{x})\exp\left\{\boldsymbol{\eta}^{\mathrm{T}}\mathbf{u}(\mathbf{x})\right\}\,\mathrm{d}\mathbf{x} = 1$$

- so g(η) can be interpreted as a normalization coefficient.

The Exponential Family

- Distribution

$$p(\mathbf{x}|\boldsymbol{\eta}) = h(\mathbf{x})g(\boldsymbol{\eta})\exp\left\{\boldsymbol{\eta}^{\mathrm{T}}\mathbf{u}(\mathbf{x})\right\}$$

  - η: natural parameters
  - x: data
  - u(x): sufficient statistic
- Normalization:

$$g(\boldsymbol{\eta})\int h(\mathbf{x})\exp\left\{\boldsymbol{\eta}^{\mathrm{T}}\mathbf{u}(\mathbf{x})\right\}\,\mathrm{d}\mathbf{x} = 1$$

  - so g(η) can be interpreted as a normalization coefficient.

The Exponential Family (2.1)

- The Bernoulli Distribution

$$\begin{aligned} p(x|\mu) &= \mathrm{Bern}(x|\mu) = \mu^x(1-\mu)^{1-x} \\ &= \exp\left\{x \ln \mu + (1-x)\ln(1-\mu)\right\} \\ &= (1-\mu)\exp\left\{\ln\left(\frac{\mu}{1-\mu}\right)x\right\} \end{aligned}$$

- Comparing with the general form we see that

$$\eta = \ln\left(\frac{\mu}{1-\mu}\right)$$ and so $$\mu = \underbrace{\sigma(\eta) = \frac{1}{1+\exp(-\eta)}}_{\text{Logistic sigmoid}}.$$

The Exponential Family (2.2)

- The Bernoulli distribution can hence be written as

$$p(x|\eta) = \sigma(-\eta)\exp(\eta x)$$

- where

$$\begin{aligned} u(x) &= x \\ h(x) &= 1 \\ g(\eta) &= 1 - \sigma(\eta) = \sigma(-\eta). \end{aligned}$$

The Exponential Family (3.1)

- The Multinomial Distribution

$$p(\mathbf{x}|\boldsymbol{\mu}) = \prod_{k=1}^{M} \mu_k^{x_k} = \exp\left\{\sum_{k=1}^{M} x_k \ln \mu_k\right\} = h(\mathbf{x})g(\boldsymbol{\eta})\exp\left(\boldsymbol{\eta}^{\mathrm{T}}\mathbf{u}(\mathbf{x})\right)$$

- where, $\mathbf{x} = (x_1, \ldots, x_M)^{\mathrm{T}}$, $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_M)^{\mathrm{T}}$ and

$$\begin{aligned} \eta_k &= \ln \mu_k \\ \mathbf{u}(\mathbf{x}) &= \mathbf{x} \\ h(\mathbf{x}) &= 1 \\ g(\boldsymbol{\eta}) &= 1. \end{aligned}$$

NOTE: The μ_k parameters are not independent since the corresponding μ_k must satisfy

$$\sum_{k=1}^{M} \mu_k = 1.$$

The Exponential Family (3.2)

- Let $\mu_M = 1 - \sum_{k=1}^{M-1} \mu_k$. This leads to

$$\eta_k = \ln\left(\frac{\mu_k}{1 - \sum_{j=1}^{M-1} \mu_j}\right) \quad \text{and} \quad \mu_k = \underbrace{\frac{\exp(\eta_k)}{1 + \sum_{j=1}^{M-1} \exp(\eta_j)}}_{\text{Softmax}}.$$

- Here the μ_k parameters are independent. Note that

$$0 \leqslant \mu_k \leqslant 1 \quad \text{and} \quad \sum_{k=1}^{M-1} \mu_k \leqslant 1.$$

The Exponential Family (3.3)

- The Multinomial distribution can then be written as

$$p(\mathbf{x}|\boldsymbol{\mu}) = h(\mathbf{x})g(\boldsymbol{\eta})\exp\left(\boldsymbol{\eta}^{\mathrm{T}}\mathbf{u}(\mathbf{x})\right)$$

- where

$$\begin{aligned}
\boldsymbol{\eta} &= (\eta_1, \ldots, \eta_{M-1}, 0)^{\mathrm{T}} \\
\mathbf{u}(\mathbf{x}) &= \mathbf{x} \\
h(\mathbf{x}) &= 1 \\
g(\boldsymbol{\eta}) &= \left(1 + \sum_{k=1}^{M-1} \exp(\eta_k)\right)^{-1}.
\end{aligned}$$

The Exponential Family (4)

- The Gaussian Distribution

$$
\begin{aligned}
p(x|\mu,\sigma^2) &= \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} \\
&= \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}x^2 + \frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}\mu^2\right\} \\
&= h(x)g(\boldsymbol{\eta}) \exp\left\{\boldsymbol{\eta}^{\mathrm{T}}\mathbf{u}(x)\right\}
\end{aligned}
$$

- where

$$
\boldsymbol{\eta} = \begin{pmatrix} \mu/\sigma^2 \\ -1/2\sigma^2 \end{pmatrix} \qquad h(\mathbf{x}) = (2\pi)^{-1/2}
$$

$$
\mathbf{u}(x) = \begin{pmatrix} x \\ x^2 \end{pmatrix} \qquad g(\boldsymbol{\eta}) = (-2\eta_2)^{1/2} \exp\left(\frac{\eta_1^2}{4\eta_2}\right).
$$

ML for the Exponential Family (1)

- From the definition of $g(\boldsymbol{\eta})$ we get

$$\nabla g(\boldsymbol{\eta}) \underbrace{\int h(\mathbf{x}) \exp\left\{\boldsymbol{\eta}^{\mathrm{T}} \mathbf{u}(\mathbf{x})\right\} \,\mathrm{d}\mathbf{x}}_{1/g(\boldsymbol{\eta})} + \underbrace{g(\boldsymbol{\eta}) \int h(\mathbf{x}) \exp\left\{\boldsymbol{\eta}^{\mathrm{T}} \mathbf{u}(\mathbf{x})\right\} \mathbf{u}(\mathbf{x}) \,\mathrm{d}\mathbf{x}}_{\mathbb{E}[\mathbf{u}(\mathbf{x})]} = 0$$

- Thus

$$-\nabla \ln g(\boldsymbol{\eta}) = \mathbb{E}[\mathbf{u}(\mathbf{x})]$$

ML for the Exponential Family (2)

- Given a data set, $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ the likelihood function is given by

$$p(\mathbf{X}|\boldsymbol{\eta}) = \left(\prod_{n=1}^{N} h(\mathbf{x}_n)\right) g(\boldsymbol{\eta})^N \exp\left\{\boldsymbol{\eta}^{\mathrm{T}} \sum_{n=1}^{N} \mathbf{u}(\mathbf{x}_n)\right\}.$$

- Thus we have

$$-\nabla \ln g(\boldsymbol{\eta}_{\mathrm{ML}}) = \frac{1}{N} \underbrace{\sum_{n=1}^{N} \mathbf{u}(\mathbf{x}_n)}_{\text{Sufficient statistic}}$$

Conjugate priors

- For any member of the exponential family, there exists a prior

$$p(\boldsymbol{\eta}|\boldsymbol{\chi}, \nu) = f(\boldsymbol{\chi}, \nu) g(\boldsymbol{\eta})^{\nu} \exp\left\{\nu \boldsymbol{\eta}^{\mathrm{T}} \boldsymbol{\chi}\right\}.$$

- Combining with the likelihood function, we get

$$p(\boldsymbol{\eta}|\mathbf{X}, \boldsymbol{\chi}, \nu) \propto g(\boldsymbol{\eta})^{\nu+N} \exp\left\{\boldsymbol{\eta}^{\mathrm{T}} \left(\sum_{n=1}^{N} \mathbf{u}(\mathbf{x}_n) + \nu \boldsymbol{\chi}\right)\right\}.$$

Prior corresponds to º pseudo-observations with value Â.

Next class

- Exponential family distribution
  - Generalized linear models
- Probabilistic Generative models for classification