LINEAR MODELS FOR CLASSIFICATION (PART 2)

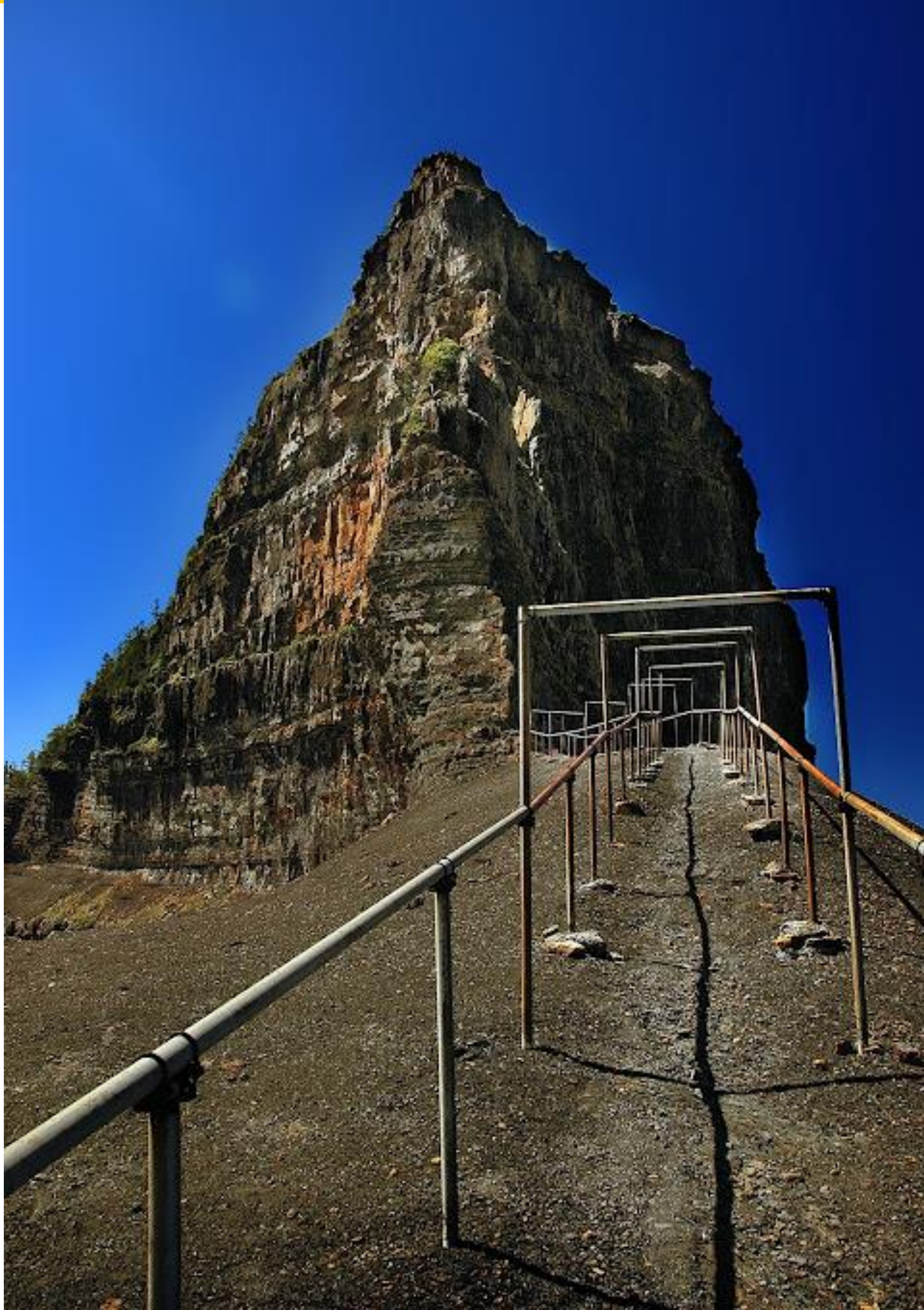
Hsin-Min Lu

盧信銘

台大資管系

Topics

- Introduction
- Discriminant Function
- Probabilistic Generative Models
- Probabilistic Discriminative Models
- Bayesian Logistic Regression



PROBABILISTIC DISCRIMINATIVE MODELS

Topics in Linear Classification using Probabilistic Discriminative Models

- Generative vs. Discriminative
- Nonlinear basis functions in linear classification
- Logistic Regression
  - Two-class, Multi-class
  - Parameter estimation using
    - Maximum Likelihood
    - Iterative Reweighted Least Squares
- Probit Regression
- Canonical Link Functions

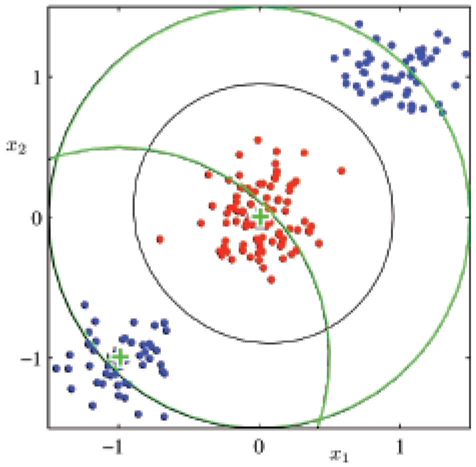
Generative vs Discriminative

- Probabilistic generative models (linear)
  - 2-class: $p(C_1|\mathrm{x})$ written as σ operating on linear function of x
    i.e., $\mathrm{w}^t\mathrm{x} + \mathrm{w}_0$, for wide choice of forms for $p(\mathrm{x}|C_k)$
  - Multiclass: $p(C_k|\mathrm{x})$ given by softmax of linear function of x
  - MLE used to get parameters of $p(\mathrm{x}|C_k)$ and priors $p(C_k)$
  - Can generate synthetic data from marginal $p(\mathrm{x})$
- Probabilistic discriminative models
  - Direct approach
  - Maximize likelihood function of conditional distribution $p(C_k|\mathrm{x})$
- Advantages
  - Fewer adaptive parameters
  - Improved performance when $p(\mathrm{x}|C_k)$ assumptions are poor approximations

Nonlinear Basis Functions in Linear Models

- Although we use linear classification models
  - Linear-separability in feature space does not imply linear-separability in input space
  - Developing useful features to enable linear separable feature space is called “feature engineering.”

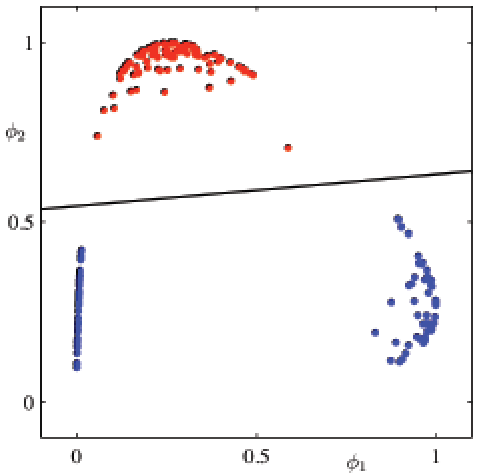
Original Input Space



not linearly separable

Nonlinear transformation of inputs using vector of basis functions $\phi(x)$

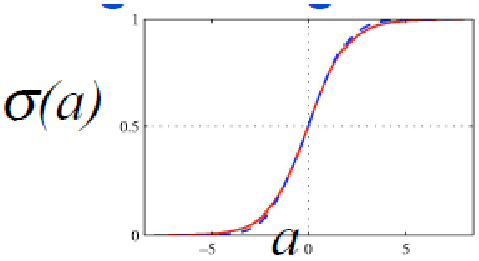
Feature Space (ϕ_1, ϕ_2)



linearly separable

2. Logistic Regression

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



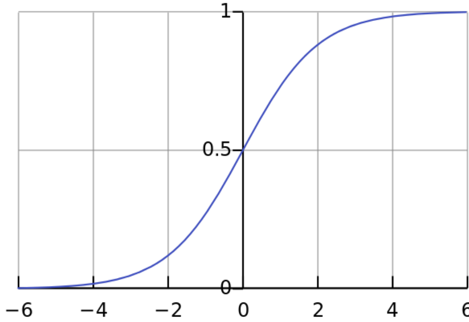
- Feature vector ϕ , two-classes *C1* and *C2*
- *A posteriori* probability *p(C₁|ϕ)* can be written as

$$p(C_1|\phi) = y(\phi) = \sigma(w^T\phi)$$

  where ϕ is a *M*-dimensional feature vector
  $\sigma(\cdot)$ is the logistic sigmoid function

- Goal is to determine the *M* parameters
- Known as logistic regression in statistics
  - Although a model for classification rather than for regression

Logistic Sigmoid



- $\sigma(a) = \frac{1}{1+\exp(-a)}$
- $1 - \sigma(a) = \frac{1+\exp(-a)-1}{1+\exp(-a)} = \frac{\exp(-a)}{1+\exp(-a)} = \frac{1}{\exp(a)+1} = \sigma(-a)$
- Symmetric: $\sigma(-a) = 1 - \sigma(a)$
- $a = \ln\left(\frac{\sigma}{1-\sigma}\right)$ is the log odds ratio
- $\ln\left(\frac{\sigma}{1-\sigma}\right) = \ln\frac{p(C_1|x)}{p(C_2|x)}$
- $\frac{\partial\sigma(a)}{\partial a} = (-1)(1+\exp(-a))^{-2}(-\exp(-a))$
- $= \frac{\exp(-a)}{[1+\exp(-a)]^2} = \sigma(a)\frac{\exp(-a)}{1+\exp(-a)} = \sigma(a)(1-\sigma(a))$

Fewer Parameters in Linear Discriminative Model

- Discriminative approach (Logistic Regression)
  - For *M* -dim space: *M* adjustable parameters
- Generative based on Gaussians (Bayes/NB)
  - *2M* parameters for mean
  - *M(M+1)/2* parameters for shared covariance matrix
  - Two class priors
  - Total of *M(M+5)/2 + 1* parameters
    - Grows quadratically with *M*
  - If features assumed independent (naïve Bayes) still needs *M+3* parameters

Determining Logistic Regression parameters

- Maximum Likelihood Approach for Two classes

  Data set $\{\phi_n, t_n\}$

  where $t_n \in \{0,1\}$ and $\phi_n = \phi(x_n), n = 1, \dots, N$

  Since *t*$_n$ is binary we can use Bernoulli

  Let *y*$_n$ be the probability that *t*$_n$ =*1*

- Likelihood function associated with *N* observations

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

- where $t = (t_1, \dots, t_N)^T$, and
- $y_n = p(C_1|\phi_n)$

Error Function for Logistic Regression

Likelihood function

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

Error function is the negative of the log-likelihood

$$\begin{aligned} E(w) &= -\ln p(t|w) \\ &= -\sum_{n=1}^{N}\{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\} \end{aligned}$$

Gradient of Error Function

- Error function:
- $E(w) = -\ln p(t|w) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\}$
- where $y_n = \sigma(w^T \phi_n)$
- Want to compute $\nabla E(w) = \frac{\partial E(w)}{\partial w}$
- Recall: $\frac{\partial \sigma(a)}{\partial a} = \sigma(a)\big(1 - \sigma(a)\big)$,
- Let $z = z_1 + z_2$,
- where $z_1 = t \ln \sigma(w^T \phi)$, $z_2 = (1 - t)\ln[1 - \sigma(w^T \phi)]$.
- $\frac{\partial z_1}{\partial w} = \frac{t\, \sigma(w^T \phi)\big(1 - \sigma(w^T \phi)\big)\phi}{\sigma(w^T \phi)} = t[1 - \sigma(w^T \phi)]\phi$
- $\frac{\partial z_2}{\partial w} = \frac{(1-t)(-1)\sigma(w^T \phi)(1 - \sigma(w^T \phi)\phi}{1 - \sigma(w^T \phi)} = (t - 1)\sigma(w^T \phi)\phi$
- $\frac{\partial z}{\partial w} = \frac{\partial z_1}{\partial w} + \frac{\partial z_2}{\partial w} = \big(t - \sigma(w^T \phi)\big)\phi$

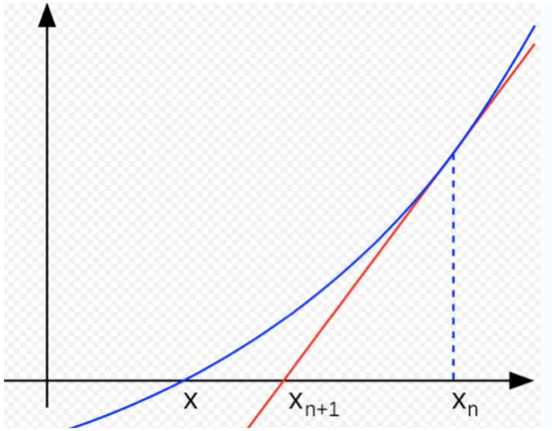
Gradient of Error Function (Cont'd.)

- Error function:
- $E(w) = -\ln p(t|w) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$
- where $y_n = \sigma(w^T\phi_n)$
- Let $z = z_1 + z_2$,
- where $z_1 = t\ln\sigma(w^T\phi)$, $z_2 = (1-t)\ln[1-\sigma(w^T\phi)]$.
- $\frac{\partial z}{\partial w} = \frac{\partial z_1}{\partial w} + \frac{\partial z_2}{\partial w} = \left(t - \sigma(w^T\phi)\right)\phi$
- $\nabla E(w) = \frac{\partial E(w)}{\partial w} = \sum_{n=1}^{N}(y_n - t_n)\phi_n$
- Contribution to gradient by data point *n* is error between target t_n and prediction $y_n = \sigma(w^T\phi_n)$ times basis ϕ_n

Finding the Minimal Value of $E(w)$

- It is clear from the previous slide that we cannot not solve $\nabla E(w) = \sum_{n=1}^{N}(y_n - t_n)\phi_n = 0$ for w directly.
- Need numerical algorithm for this task.
- Solving for $\nabla E(w) = 0$ is sometimes called the root finding problem.
- One well known approach is called the Newton-*Raphson* Method.
- Start with x_n, use the tangent line at $f(x_n)$ to determine x_{n+1}, and repeat.

Newton's Method



$$f'(x_n) = \frac{\text{rise}}{\text{run}} = \frac{\Delta \text{y}}{\Delta \text{x}} = \frac{f(x_n) - 0}{x_n - x_{n+1}}.$$

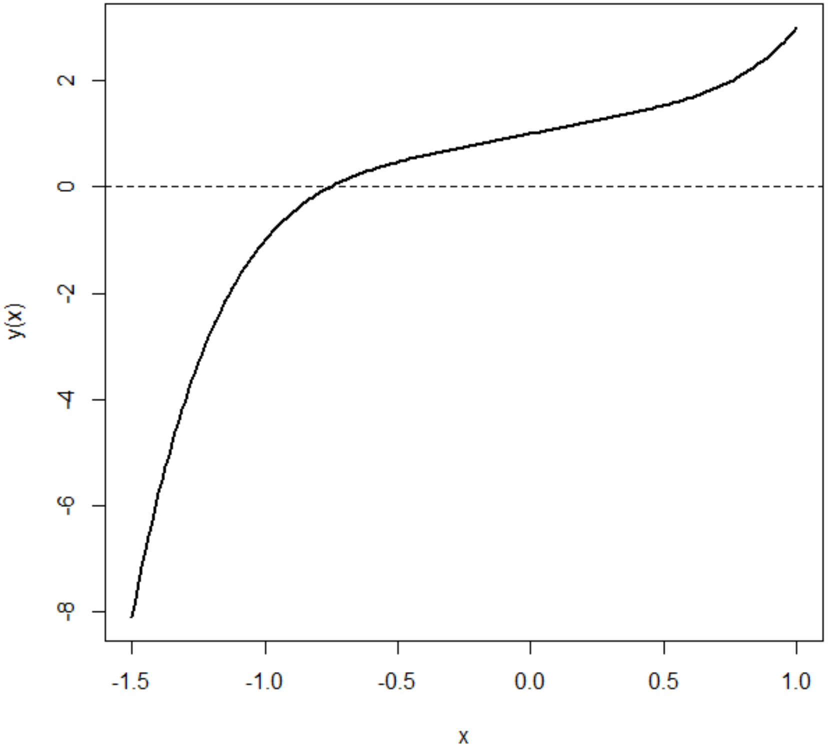
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Since we are solving for the derivative of *E*(w) Hessian of E(w) is needed

Newton-*Raphson* Method (One Dimensional)

- Example: Given $f(x) = x^5 + x - 1$
- Use $x_{n+1} = x_n - \frac{f(x_n)}{f\prime(x_n)}$ to approximate the root
- $f'(x) = 5x^4 + 1$, we start from $x_1 = 1$,

- starting from x= 1
- step 1 : 0.5
- step 2 : -0.6666667
- step 3 : -0.7681159
- step 4 : -0.7551625
- step 5 : -0.7548778
- step 6 : -0.7548777
- step 7 : -0.7548777
- step 8 : -0.7548777
- step 9 : -0.7548777
- step 10 : -0.7548777



Iterative Reweighted Least Squares (IRLS)

- Efficient approximation using *Newton-Raphson* iterative optimization
- $w^{(new)} = w^{(old)} - H^{-1} \nabla E(w)$
  - where *H* is the Hessian matrix
- $H = \frac{\partial E(w)}{\partial w \partial w^T} = \nabla \nabla E(w)$
- Elements in H are the second derivatives of *E(*w*)* with respect to the components of w

Two applications of IRLS

- IRLS is applicable to both Linear Regression and Logistic Regression
- We discuss both, for each we need
  1. Error function *E(*w*)*
     - Linear Regression: Sum of Squared Errors
     - Logistic Regression: Bernoulli Likelihood Function
  2. Gradient $\nabla E(w)$
  3. Hessian $H = \nabla\nabla E(w)$
  4. Newton-Raphson update

$$w^{(new)} = w^{(old)} - H^{-1}\nabla E(w)$$

IRLS for Linear Regression

- Recall the linear regression model is:
- $y(x,w) = \sum_{j=0}^{M-1} w_j \phi_j(x) = w^T \phi(x)$
- Data set: $X = \{x_n, t_n\}\ n = 1, \dots, N$

1. Error Function: Sum of Squared Errors

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - w^T\phi(x_n)\}^2 = \frac{1}{2}(\boldsymbol{t} - \Phi w)^T(\boldsymbol{t} - \Phi w)$$

$$= \frac{1}{2}[\boldsymbol{t}^T\boldsymbol{t} - 2w^T\Phi^T\boldsymbol{t} + w^T\Phi^T\Phi w]$$

2. Gradient of Error Function is:

$$\nabla E(w) = \Phi^T\Phi w - \Phi^T\boldsymbol{t}$$

3. Hessian is:

$$H = \nabla\nabla E(w) = \sum_{n=1}^{N}\phi_n\phi_n^T = \Phi^T\Phi$$

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathrm{x}_1) & \phi_1(\mathrm{x}_1) & \dots & \phi_{M-1}(\mathrm{x}_1) \\ \phi_0(\mathrm{x}_2) & & & \\ & & & \\ \phi_0(\mathrm{x}_N) & & & \phi_{M-1}(\mathrm{x}_N) \end{pmatrix}$$

Φ is the *N* x *M* design matrix whose *n*th row is given by ϕ_n^T

4. Newton-Raphson for Linear Regression

$$w^{(new)} = w^{(old)} - H^{-1}\nabla E(w)$$

Substituting: $H = \Phi^T\Phi$ and $\nabla E(w) = \Phi^T\Phi w - \Phi^T\boldsymbol{t}$

$$w^{(new)} = w^{(old)} - (\Phi^T\Phi)^{-1}\{\Phi^T\Phi w^{(old)} - \Phi^T\boldsymbol{t}\}$$

$$= (\Phi^T\Phi)^{-1}\Phi^T\boldsymbol{t}$$

which is the standard least squares solution

Since it is independent of w, Newton-Raphson gives exact solution in one step

IRLS for Logistic Regression

- Posterior probability of class C_1 is

$$p(C_1|\phi) = y(\phi) = \sigma(w^T\phi)$$

- Likelihood Function
  - For data set $\{\phi_n, t_n\}$, $t_n \in \{0,1\}$, $\phi_n = \phi(x_n)$
  - $p(t|w) = \prod_{n=1}^{N} y_n^{t_n}\{1 - y_n\}^{1-t_n}$
- Error Function

$$E(w) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\}$$

$y_n = \sigma(w^T\phi_n)$

IRLS for Logistic Regression

- Gradient of Error Function:

$$\nabla E(w) = \sum_{n=1}^{N} (y_n - t_n)\phi_n = \Phi^T(y - t)$$

- $y_n = \sigma(w^T \phi_n)$
- Hessian: $H = \nabla\nabla E(w) = \sum_{n=1}^{N} y_n(1 - y_n)\phi_n \phi_n^T = \Phi^T R \Phi$
- R is NxN diagonal matrix with elements: $R_{nn} = y_n(1 - y_n)$
- Hessian is not constant and depends on w through R
- Since H is positive-definite (i.e., for arbitrary u, u^THu>0), error function is a concave function of w and so has a unique minimum

IRLS for Logistic Regression

- Newton-Raphson update:

$$w^{(new)} = w^{(old)} - H^{-1}\nabla E(w)$$

- Substituting $H = \Phi^T R \Phi$ and $\nabla E(w) = \Phi^T(y - t)$
  - $w^{(new)} = w^{(old)} - (\Phi^T R \Phi)^{-1}\Phi^T(y - t)$
  - $= (\Phi^T R \Phi)^{-1}\{\Phi^T R \Phi w^{(old)} - \Phi^T(y - t)\}$
  - $= (\Phi^T R \Phi)^{-1}\Phi^T R\left\{\Phi w^{(old)} - R^{-1}(y - t)\right\}$
  - $= (\Phi^T R \Phi)^{-1}\Phi^T R z$
- z is a *N*-dimensional vector with elements
- $z = \Phi w^{(old)} - R^{-1}(y - t)$

Update formula is a set of normal equations
Since Hessian depends on w
Apply them iteratively each time using the new weight vector

Multi-class Logistic Regression

- Work with soft-max function instead of logistic sigmoid
- $P(C_k|\phi) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$
- where $a_k = w_k^T \phi$

Multi-class Likelihood Function

- *1*-of –*K* Coding Scheme
  - For feature vector ϕ_n, target vector t$_n$ belonging to class C_k is a binary vector with all elements zero except for element *k*

$$p(T|w_1, \dots, w_k) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|\phi_n)^{t_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

  - where $y_{nk} = y_k(\phi_n)$
  - subject to (s.t.) $\sum_k t_{nk} = 1$ (because of 1-of-K coding)
  - T is a *N x K* matrix of elements with elements t_{nk}

Multi-class Error Function

- Error Function: negative log-likelihood

$$E(w_1, \dots, w_k) = -\ln p(T|w_1, \dots, w_k) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

- Derivatives of soft-max:
- $y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$, $a_k = w_k^T \phi$
- $\frac{\partial y_k}{\partial a_j} = y_k\left(I_{kj} - y_j\right)$, where I_{kj} are elements of the identity matrix. [Exercise]

Multi-class Error Function

- Error Function: negative log-likelihood

$$E(w_1,\dots,w_k) = -\ln p(T|w_1,\dots,w_k) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\ln y_{nk}$$

- $y_{nk}(\phi) = \frac{\exp(a_{nk})}{\sum_j \exp(a_{nj})},\ a_{nk} = w_k^T\phi_n$
- Want: $\frac{\partial E(\cdot)}{\partial w_j} = \sum_{n=1}^{N}\sum_{k=1}^{K}\frac{-\partial t_{nk}\ln y_{nk}}{\partial y_{nk}}\frac{\partial y_{nk}}{\partial a_{nj}}\frac{\partial a_{nj}}{\partial w_j} =$
- $= \sum_{n=1}^{N}\sum_{k=1}^{K} -\frac{t_{nk}}{y_{nk}}\left[y_{nk}\left(I_{jk}-y_{nj}\right)\right]\phi_n$
- $= \sum_{n=1}^{N}\sum_{k=1}^{K}\left(y_{nj}t_{nk} - I_{jk}t_{nk}\right)\phi_n$
- $$= \sum_{n=1}^{N}\underbrace{(y_{nj} - t_{nj})}_{\text{Prediction error of the j-th outcome}}\phi_n \quad \left(since \sum_k t_{nk} = 1\right)$$

IRLS Algorithm for Multi-class

- Hessian matrix comprises blocks of size *M x M*
  - Block *j,k* is given by

$$\nabla_{w_k}\nabla_{w_j}E(w_1,\ldots,w_k) = -\sum_{n=1}^{N} y_{nk}(I_{kj}-y_{nj})\phi_n\phi_n^T$$

  - Hessian matrix is positive-definite, therefore error function has a unique minimum
  - Note the total size of H is $MK \times MK$
- Batch Algorithm based on Newton-Raphson

3. Probit Regression

- Logistic transformation is good for exponential family
- Not suitable for some types of probability-based inference models (e.g. Gaussian Mixture)
- Alternative discriminative model is based on probit function (which is the CDF of a zero-mean Gaussian)
    - Note that a CDF also goes between 0 and 1

Probit Activation Function

- Two-class case, Generalized Linear Model

  $$p(t = 1|a) = f(a)$$

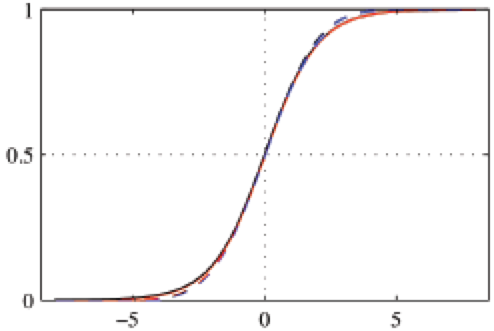
  - where $a = w^T\phi$ and $f(.)$ is the activation function
- Consider stochastic (noisy) threshold model
  - For input ϕ_n, evaluate $a_n = w^T\phi_n$ and draw a random variable $\theta \sim N(\theta; 0, 1)$.
  - Assign target value as $\begin{cases} t_n = 1 \;\; if\ a_n \geq \theta \\ t_n = 0 \;\; otherwise \end{cases}$
  - Then $p(t_n = 1|a_n) = p(\theta \leq a_n)$

    $$= \int_{-\infty}^{a_n} N(\theta; 0, 1)d\theta = \Phi(a_n)$$

Probit Function

It is the CDF of a zero-mean unit-variance Gaussian

$$\boxed{\Phi(a) = \int_{-\infty}^{a} N(\theta|0,1)d\theta}$$



It has a *sigmoidal* shape & related to the *erf* function which is usually tabulated

$$\mathrm{erf}(a) = \frac{2}{\sqrt{\pi}}\int_{0}^{a} \exp\left(\frac{-\theta^2}{2}\right) d\theta$$

It represents the probability that the error lies between $\pm\, a$

With the relationship

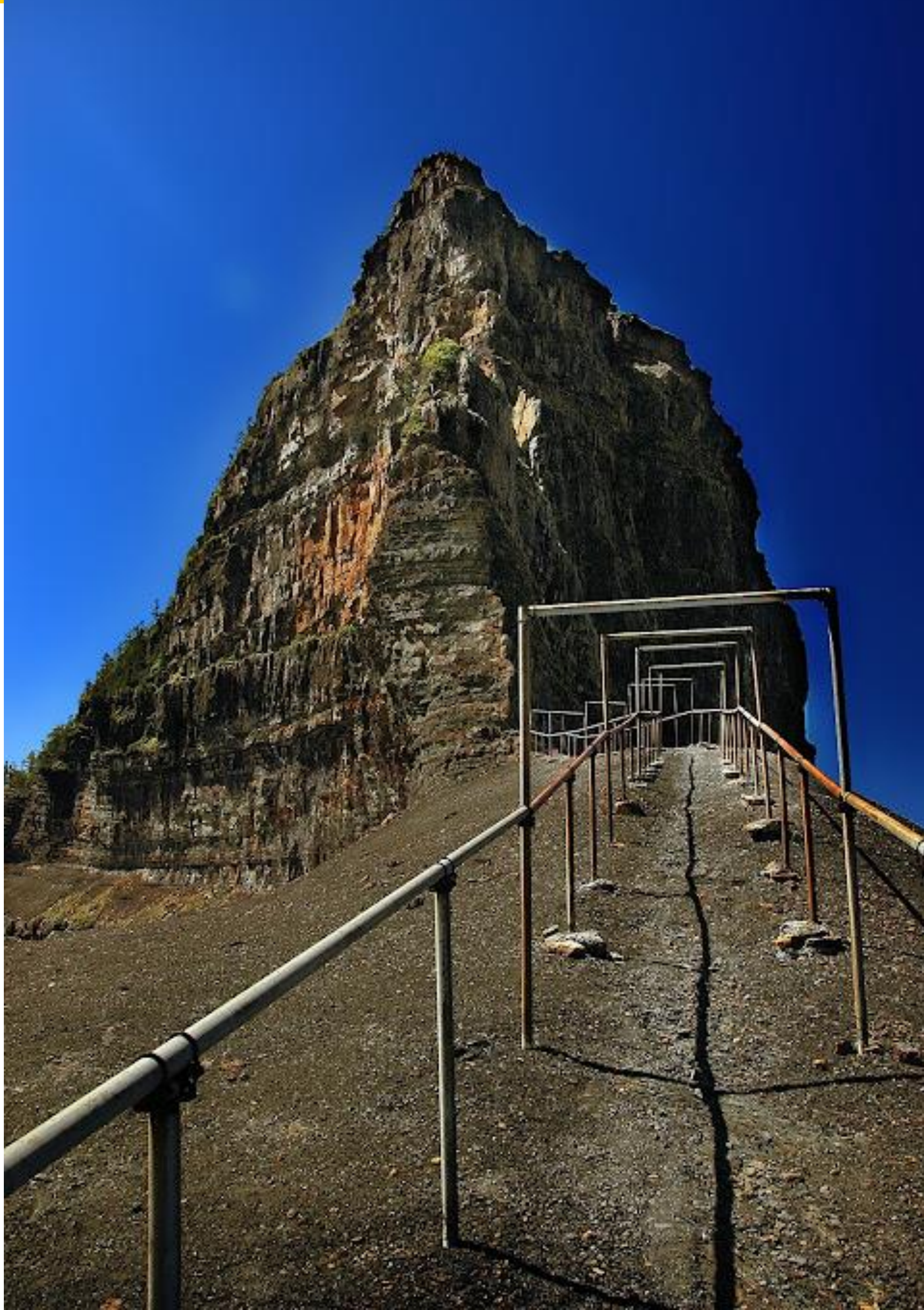
$$\Phi(a) = \frac{1}{2}\left\{1 + \frac{1}{\sqrt{2}}\mathrm{erf}(a)\right\}$$

Probit Regression

- Generalized Linear Model based on Probit activation function
- Parameters determined using maximum likelihood
- Results similar to Logistic Regression
  - Probit is more sensitive to outliers

Topics

- Introduction
- Discriminant Function
- Probabilistic Generative Models
- Probabilistic Discriminative Models
- Bayesian Logistic Regression



BAYESIAN LOGISTIC REGRESSION

Recap of Logistic Regression

- Feature vector ϕ , two-classes C_1 and C_2
- *A posteriori* probability $p(C_1|\phi)$ can be written as $p(C_1|\phi) = y(\phi) = \sigma(w^T\phi)$

  where ϕ is a *M*-dimensional feature vector

  $\sigma(\cdot)$ is the logistic sigmoid function
- Goal is to determine the *M* parameters
- Potential problem:
  - Overfitting with large number of features.
  - Overfitting for nearly separable datasets.

Roadmap of Bayesian Logistic Regression

- Logistic regression is discriminative probabilistic linear classification $p(C_1|x) = \sigma(w^T\phi)$
- Exact Bayesian inference for Logistic Regression is intractable

1. Evaluation of posterior distribution $p(w|t)$
   - Prior $p(w) = N(w|m_0, s_0)$ times likelihood (a product of sigmoids) $p(t|w) = \prod_{n=1}^{N} y_n^{t_n}\{1-y_n\}^{1-t_n}$
   - Need to apply Laplace approximation
2. Evaluation of predictive distribution ($p(w|t) \approx q(w)$) $p(C_1|\phi, t) \cong \int \sigma(w^T\phi)q(w)dw$
   - Approximate Sigmoid by Probit

Video for additional discussion (Laplace Approx.): https://www.youtube.com/playlist?list=PLvVPTibZrkBpC2Ga8xizhOH1ObiYz8wzq

Evaluation of Posterior Distribution

- Gaussian prior $p(w) = N(w|m_0, S_0)$
  - where m_0 and S_0 are hyper-parameters
- Posterior distribution $p(w|t) \propto p(w)p(t|w)$
  - where $t = (t_1, \dots, t_N)^T$
- Substituting $p(t|w) = \prod_{n=1}^{N} y_n^{t_n}\{1 - y_n\}^{1-t_n}$
  - where $y_n = \sigma(w^T \phi_n)$
- $\ln p(w|t) = -\frac{1}{2}(w - m_0)^T S_0^{-1}(w - m_0)$ $+ \sum_{i=1}^{n}[(t_n \ln y_n) + (1 - t_n)\ln(1 - y_n)] + const$
- Usually set $m_0 = 0$, $S_0 = \frac{1}{\lambda} I$

Finding Posterior Mode: $p(w|t)$

- $\ln p(w|t) = -\frac{\lambda}{2} w^T w$
  $+ \sum_{i=1}^{n} [(t_n \ln y_n) + (1 - t_n) \ln(1 - y_n)] + const$
- To be consistent with our previous discussion, define $E(w) = -\ln p(w|t)$
- Apply $w^{\tau+1} = w^{\tau} - H^{-1} \nabla E_n$ *(Newton-Raphson Numerical Optimization)*
- $\nabla E(w) = \lambda w + \sum_{n=1}^{N} (y_n - t_n) \phi_n = \lambda w + \Phi^T (y - t)$
- Hessian: $\nabla \nabla E(w) = \lambda I + \sum_{n=1}^{N} y_n (1 - y_n) \phi_n \phi_n^T$
  $= \Phi^T R \Phi + \lambda I$
  - Φ is NxM design matrix whose nth row is ϕ_n^T
  - R is NxN diagonal matrix with element $R_{nn} = y_n (1 - y_n)$

Hessian is not constant and depends on w through R
Since H is positive-definite (i.e., for arbitrary u, $u^T H u > 0$), error function is a concave function of w and so has a unique minimum

Newton-Raphson Update

- $w^{(new)} = w^{(old)} - H^{-1} \nabla E(w)$
- $\nabla E(w) = \lambda w^{(old)} + \Phi^T (y - t)$
- Hessian: $\nabla \nabla E\left(w^{(old)}\right) = \Phi^T R \Phi + \lambda I$
- ➔ $w^{(new)} = w^{(old)} - (\Phi^T R \Phi + \lambda I)^{-1} \left[\lambda w^{(old)} + \Phi^T (y - t)\ \right]$
- Iterate until converge ➔ w_0 (mode of $p(w|t)$)