[2018 Elice Machine Learning Basic Course]

Introduction to Linear Models for Classification

Heung-Il Suk

hisuk@korea.ac.kr

http://www.ku-milab.org

Department of Brain and Cognitive Engineering,
Korea University

May 29, 2018

Contents

Machine Learning Overview

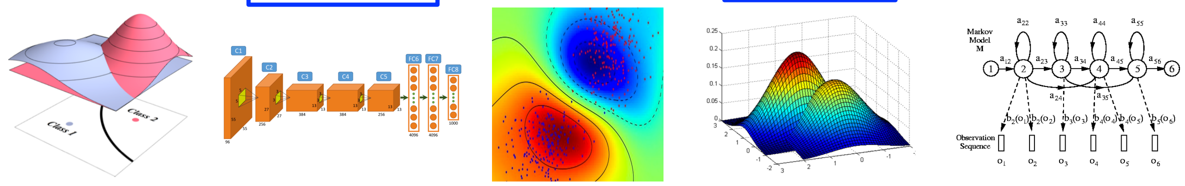
“Learning” in Machine Learning

- Most cases, data are cheap and abundant, but knowledge is expensive and scarce
- Learning: to build computer models that can analyze data and extract information automatically from them
- Induction: process of extracting general rules from a set of particular examples
- Build a model that is a good and useful approximation to the data

- *e.g.*, handwritten digit recognition ▸ Example
  - Collect a large set of N digits, *training set*, $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
  - Express the category of a digit using a *target vector* $\mathbf{t}$
  - Determine a function $f(\mathbf{x})$, *training* or *learning*: to generate an output vector $\mathbf{y}$, encoded in the same way as the target vector $\mathbf{t}$

$$\mathbf{x} \quad \Rightarrow \quad f(\mathbf{x}; \theta) \quad \Rightarrow \quad \mathbf{y}$$

$\boxed{p(y|\mathbf{x};\mu,\Sigma)}$ $\boxed{S_j^{\downarrow}\left[F_j\left[\sum_j w_{ij} * X_j\right]\right]}$ $\boxed{\sum_{m\in S} a_m k(\mathbf{x},\mathbf{x}_m)+b}$ $\boxed{\sum_{k=1}^{K}\pi_k N(\mathbf{x}|\mu_k,\Sigma_k)}$ $\boxed{p(S_t|S_{t-1})p(\mathbf{x}_t|S_t)}$



Bayes classifier | Convolutional neural network | Support vector machine | Gaussian mixture model | Hidden Markov model

Generalization: the ability to categorize correctly new examples that possibly differ from those used for training

Feature Extraction/Representation

- To transform the original input variables into some new space of variables
- Hope to be easier to solve the problem in the new space
- To lessen computational burden (in real-time applications)
- Careful not to discard the useful discriminator information
- New test data must be preprocessed using the same steps as the training data

ML System Overview

Training session

1. Collecting training samples
2. Preprocessing
3. Feature extraction/representation
4. Feature selection
5. **Classifier/regressor learning**

Testing session

1. Given testing samples
2. Preprocessing
3. Feature extraction/representation
4. Feature selection
5. Outputs from classifier/regressor

General Learning Scheme in ML

Given *i.i.d.* samples $X = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, the aim is to build a good and useful approximation to y_n

$$\mathbf{x} \quad \Rightarrow \quad f(\mathbf{x}|\theta) \quad \Rightarrow \quad y$$

1. Model: $f(\mathbf{x}|\theta) \rightarrow$ enough capacity
2. Loss function: $J(\theta|X) = \sum_n L(y_n, f(\mathbf{x}_n|\theta)) \rightarrow$ sufficient training data
3. Learning: $\theta^* = \underset{\theta}{\operatorname{argmin}}\, J(\theta|X) \rightarrow$ good optimization method

Terminology

$$\mathbf{x} \quad \Rightarrow \quad f(\mathbf{x};\theta) \quad \Rightarrow \quad \mathbf{y}$$

- **Supervised learning**
  - Regression: continuous outputs
  - Classification: discrete or category outputs
- **Unsupervised learning**
  - Clustering
  - Density estimation
  - Visualization
- **Reinforcement learning**
  - Finding suitable actions to take in a given situation in order to maximize a reward
  - No optimal outputs are given, but must discover them by a process of trial and error

Linear Classification Models

Introduction

$$\mathbf{x} \quad \Rightarrow \quad f(\mathbf{x};\theta) \quad \Rightarrow \quad \mathbf{y}$$

Regression

- assign an input vector $\mathbf{x}$ to one or more continuous target variables t

Classification

- assign an input vector $\mathbf{x}$ to one of K discrete classes C_k, $k = 1, \ldots, K$

Linear Classification Models

$$\mathbf{x} \quad \Rightarrow \quad f(\mathbf{x};\theta) \quad \Rightarrow \quad \mathbf{y}$$

- Disjoint classes (common)
- Input space: divided into decision regions
- Decision surfaces are linear functions of an input $\mathbf{x}$
  - $D-1$ dimensional hyperplane within D dimensional input space
  - Straight line in $2D$
  - $2D$ plane in $3D$
  - Hyperplane in higher than $3D$
- Linearly separable: Data sets whose classes can be separated by linear decision surface

Class Label Representations

$$\mathbf{x} \quad \Rightarrow \quad f(\mathbf{x};\theta) \quad \Rightarrow \quad \mathbf{y}$$

- Two class ($K = 2$): binary representation
  - $t \in \{1(C_1), -1(C_2)\}$
  - $t \in \{1(C_1), 0(C_2)\}$: interpreting value of t as probability that class is C_1
- For $K > 2$: 1-of-K coding scheme
  - $\mathbf{t} \in \{0, 1\}^K$ is a vector of length K
  - *e.g.*, $\mathbf{t} = [0, 1, 0, 0, 0]^\top$: a pattern of class C_2 when $K = 5$
  - can interpret a value of t_k as probability of class C_k

Different Approaches to Classification

1. Discriminant function
   - Directly assign $\mathbf{x}$ to a specific class
   - *e.g.*, Fisher's linear discriminant, perceptron
2. Probabilistic models
   - Model $p(C_k|\mathbf{x})$ in inference stage (directly or by a Bayes rule)
   - Use it to make optimal decisions

Probabilistic Models

- Generative
  - Model class conditional densities by $p(\mathbf{x}|C_k)$ together with prior probabilities $P(C_k)$
  - Then use a Bayes rule to compute posterior

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)\,P(C_k)}{p(\mathbf{x})}$$

- Discriminative
  - Directly model conditional probabilities $p(C_k|\mathbf{x})$

- Separating inference from decision (to explicitly obtain posterior) is better
  - Minimize risk
  - Reject option (minimize expected loss)
  - Compensate for unbalanced data
    - Use modified balanced data & scale by class fractions
  - Combine models

Discriminant Functions

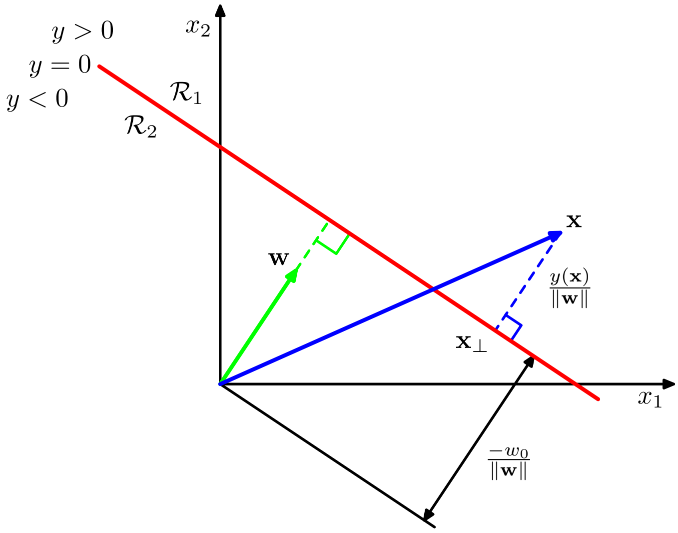
Geometry of Linear Discriminant Functions

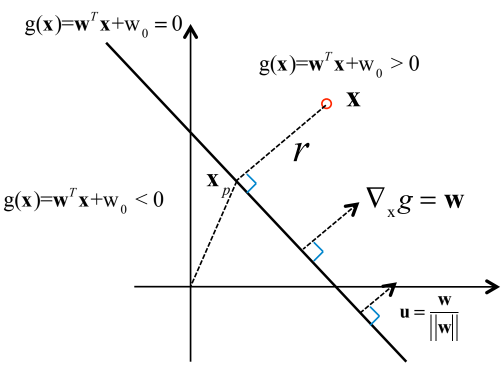
- Two-class linear discriminant function

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

($\mathbf{w}$: weight vector, w_0: bias/threshold)

  - Assign $\mathbf{x}$ to class C_1 if $y(\mathbf{x}) \geq 0$, otherwise class C_2
  - Decision boundary: $y(\mathbf{x}) = 0$
    - Geometrically, $\mathbf{w}$ determines the orientation of the decision surface

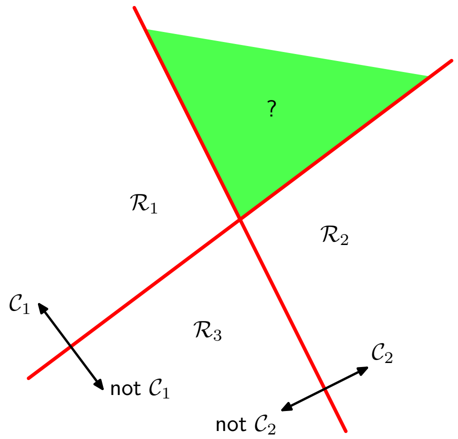




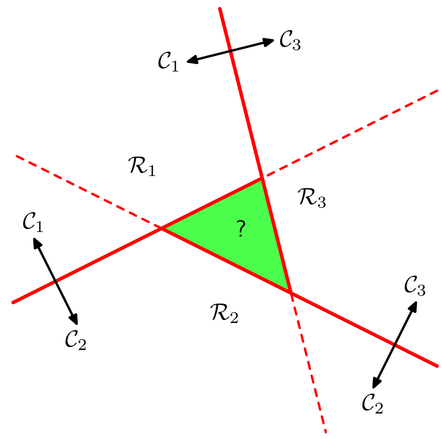
$$\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$$
$$g(\mathbf{x}) = \mathbf{w}^\top\left(\mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|}\right) + w_0$$
$$g(\mathbf{x}) = \left(\mathbf{w}^\top\mathbf{x}_p + w_0\right) + r\frac{\mathbf{w}^\top\mathbf{w}}{\|\mathbf{w}\|}$$
$$g(\mathbf{x}) = r\frac{\|\mathbf{w}\|^2}{\|\mathbf{w}\|} = r\|\mathbf{w}\|$$

Multiple Classes with Binary Classifiers

One-versus-Rest ($K-1$ classifiers)



One-versus-one ($K(K-1)/2$ classifiers)



Multiple Classes with K Discriminants

- Consider a single K class discriminant of the form

$$y_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + w_{k0}$$

- Assign a point $\mathbf{x}$ to class C_k if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$
  - Decision boundary between class C_k and C_j: $y_k(\mathbf{x}) = y_j(\mathbf{x})$
  - $D-1$ dimensional hyperplane defined by

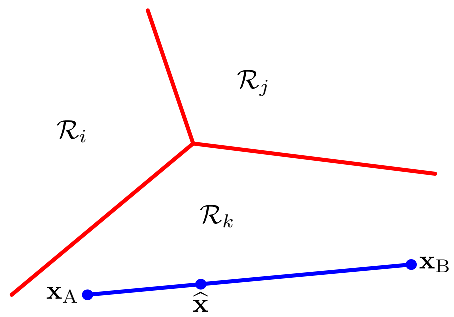
$$(\mathbf{w}_k - \mathbf{w}_j)^\top \mathbf{x} + (w_{k0} - w_{j0}) = 0$$

Decision regions of such a discriminant are always singly connected and convex

Convexity of Decision Regions

Proof!!!

$$\begin{aligned}
\hat{\mathbf{x}} &= \lambda \mathbf{x}_A + (1-\lambda)\mathbf{x}_B \\
& \quad (0 \le \lambda \le 1) \\
y_k(\hat{\mathbf{x}}) &= \lambda y_k(\mathbf{x}_A) + (1-\lambda) y_k(\mathbf{x}_B) \\
y_k(\mathbf{x}_A) &> y_j(\mathbf{x}_A) \\
y_k(\mathbf{x}_B) &> y_j(\mathbf{x}_B) \\
y_k(\hat{\mathbf{x}}) &> y_j(\hat{\mathbf{x}})
\end{aligned}$$



Learning Parameters of Linear Discriminant Functions

- Least Squares
- Fisher's Linear Discriminant
- Perceptrons

Least Squares for Classification

- Analogous to regression, there exists a simple closed-form solution for parameters
- Each C_k, $k = 1, \ldots, K$, is described by

$$y_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + w_{k0}$$

- By grouping into vector notation

$$\mathbf{y}(\mathbf{x}) = \tilde{\mathbf{W}}^\top \tilde{\mathbf{x}}$$

  - Augmented vectors: $\tilde{\mathbf{w}}_k = \left[w_{k0}, \mathbf{w}_k^\top\right]^\top$, $\tilde{\mathbf{x}} = \left[1, \mathbf{x}^\top\right]^\top$
  - $\tilde{\mathbf{W}} = [\tilde{\mathbf{w}}_1, \cdots, \tilde{\mathbf{w}}_K]$

- A new input vector $\mathbf{x}$ is assigned to class for which the output $y_k = \tilde{\mathbf{w}}_k^\top \tilde{\mathbf{x}}$ is the largest.

Given a training set $\{\mathbf{x}_n, \mathbf{t}_n\}$, $n = 1, \ldots, N$,

- Sum of squares error function

$$E_D\left(\tilde{\mathbf{W}}\right) = \frac{1}{2}\mathrm{Tr}\left\{\left(\mathbf{X}\tilde{\mathbf{W}} - \mathbf{T}\right)^\top \left(\mathbf{X}\tilde{\mathbf{W}} - \mathbf{T}\right)\right\}$$

$$\text{where } \left\{ \begin{array}{l} \mathbf{T} \equiv \left[\mathbf{t}_1^\top; \cdots; \mathbf{t}_N^\top\right] \in \mathbb{R}^{N \times K} \\ \mathbf{X} \equiv \left[\tilde{\mathbf{x}}_1^\top; \cdots; \tilde{\mathbf{x}}_N^\top\right] \in \mathbb{R}^{N \times (D+1)} \end{array} \right.$$

$$
\begin{aligned}
\mathsf{Tr}(A) &= \mathsf{Tr}\left(A^\top\right) \\
\mathsf{Tr}(A+B) &= \mathsf{Tr}(A) + \mathsf{Tr}(B) \\
\mathsf{Tr}(aA) &= a\mathsf{Tr}(A)
\end{aligned}
\qquad
\begin{aligned}
\nabla_A \mathsf{Tr}(AB) &= B^\top \\
\nabla_{A^\top} f(A) &= \left(\nabla_A f(A)\right)^\top \\
\nabla_A \mathsf{Tr}\left(ABA^\top C\right) &= CAB + C^\top AB^\top
\end{aligned}
$$

$$
\begin{aligned}
\nabla_A E_D\left(\tilde{\mathbf{W}}\right) &= \nabla_A \frac{1}{2}\mathsf{Tr}\left\{\left(\mathbf{X}\tilde{\mathbf{W}} - \mathbf{T}\right)^\top \left(\mathbf{X}\tilde{\mathbf{W}} - \mathbf{T}\right)\right\} \\
&= \frac{1}{2}\nabla_A\left[\mathsf{Tr}\left(\tilde{\mathbf{W}}^\top\mathbf{X}^\top\mathbf{X}\tilde{\mathbf{W}} - \tilde{\mathbf{W}}^\top\mathbf{X}^\top\mathbf{T} - \mathbf{T}^\top\mathbf{X}\tilde{\mathbf{W}} + \mathbf{T}^\top\mathbf{T}\right)\right] \\
&= \frac{1}{2}\left[\nabla_A \mathsf{Tr}\left(\tilde{\mathbf{W}}^\top\mathbf{X}^\top\mathbf{X}\tilde{\mathbf{W}}\right) - \nabla_A \mathsf{Tr}\left(-\tilde{\mathbf{W}}^\top\mathbf{X}^\top\mathbf{T}\right) - \nabla_A\left(\mathbf{T}^\top\mathbf{X}\tilde{\mathbf{W}}\right)\right] \\
&= \frac{1}{2}\left[\nabla_A \mathsf{Tr}\left(\tilde{\mathbf{W}}\tilde{\mathbf{W}}^\top\mathbf{X}^\top\mathbf{X}\right) - \nabla_A \mathsf{Tr}\left(\tilde{\mathbf{W}}\mathbf{T}^\top\mathbf{X}\right) - \nabla_A\left(\tilde{\mathbf{W}}\mathbf{T}^\top\mathbf{X}\right)\right] \\
&= \frac{1}{2}\left[\left\{\mathbf{X}^\top\mathbf{X}\tilde{\mathbf{W}} + \left(\mathbf{X}\mathbf{X}^\top\right)^\top\tilde{\mathbf{W}}\right\} - \mathbf{X}^\top\mathbf{T} - \left(\mathbf{T}^\top\mathbf{X}\right)^\top\right] \\
&= \mathbf{X}^\top\mathbf{X}\tilde{\mathbf{W}} - \mathbf{X}^\top\mathbf{T} = 0 \\
\tilde{\mathbf{W}} &= \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{T}
\end{aligned}
$$

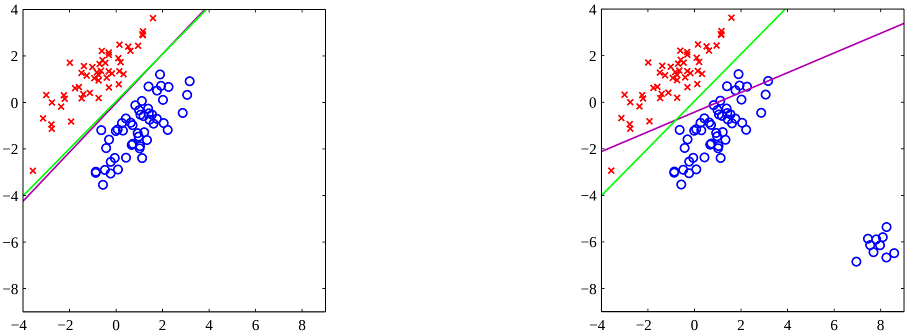
- Set derivative w.r.t. $\tilde{\mathbf{W}}$ to zero

$$
\tilde{\mathbf{W}} = \underbrace{\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top}_{\mathbf{X}^\dagger:\ \text{pseudo-inverse of } \mathbf{X}}\mathbf{T} = \mathbf{X}^\dagger\mathbf{T}
$$

- After rearranging,

$$
\mathbf{y}(\mathbf{x}) = \tilde{\mathbf{W}}^\top\mathbf{x} = \mathbf{T}^\top\left(\mathbf{X}^\dagger\right)^\top\mathbf{x}
$$

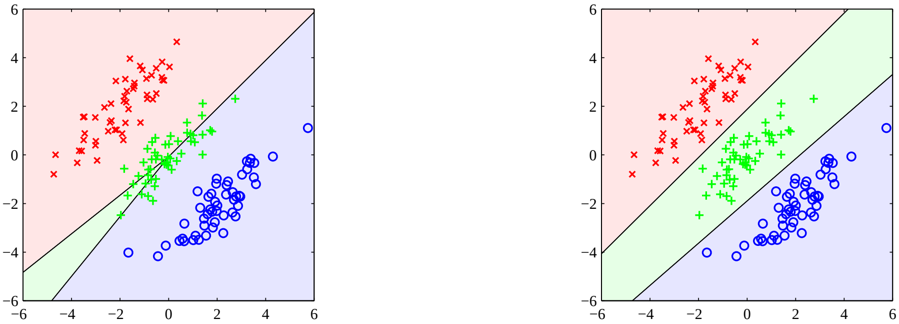
Least square is sensitive to outliers!!!



Magenta(least squares); Green(logistic regression)

- Sum of squared errors penalizes predictions that are "too correct" or long way from decision boundary

Disadvantage of Least Squares!!!



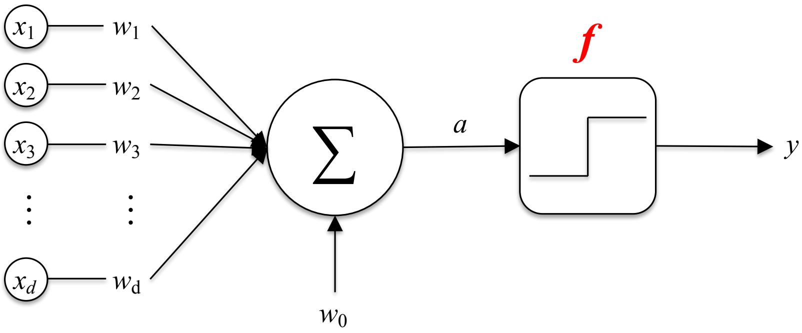
(Least squares) (Logistic regression)

The region of input space assigned to the green class is too small and so most of the points from this class are misclassified.

- (Recall) The decision boundary corresponds to maximum likelihood solution under a Gaussian conditional distribution.
- However, binary target vectors clearly have a distribution that is far from Gaussian.

Perceptron [Rosenblatt, 1962]

$$y(\mathbf{x}) = f\left(\mathbf{w}^\top \mathbf{x} + w_0\right) \quad \text{where } f(a) = \begin{cases} +1 & a \geq 0 \\ -1 & a < 0 \end{cases}$$



[Parameters Learning]

Error function minimization

- Error function: number of misclassifications
- This error function is a piecewise constant function of $\mathbf{w}$ with discontinuities (c.f., regression)
- No closed-form solution (no derivatives exist for non-smooth functions)
- Take an iterative approach

Perceptron Criterion

- Seeking $\mathbf{w}$ such that

$$\left\{\begin{array}{l} \mathbf{x}_n \in C_1 \ (t_n = +1) \text{ will have } \mathbf{w}^\top \mathbf{x}_n \geq 0 \\ \mathbf{x}_n \in C_2 \ (t_n = -1) \text{ will have } \mathbf{w}^\top \mathbf{x}_n < 0 \end{array}\right\} \Rightarrow \mathbf{w}^\top \mathbf{x}_n t_n \geq 0$$

  - Linearly bisecting the feature space

- For each misclassified sample, perceptron criterion tries to minimize

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

$\mathcal{M}$: a set of all misclassified samples

Perceptron Algorithm

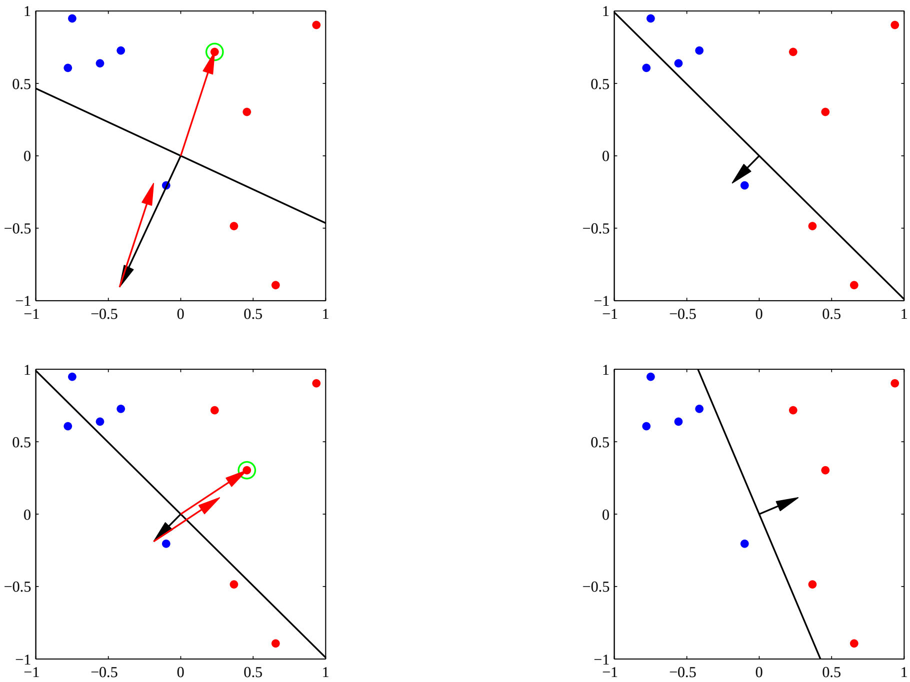
- Error function: $E_n(\mathbf{w}) = -\mathbf{w}^\top \mathbf{x}_n t_n \ (n \in \mathcal{M})$
- Stochastic gradient descent

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

η: learning rate, τ: step index

  - Since $y(\mathbf{x}, \mathbf{w})$ is unchanged if we multiply $\mathbf{w}$ by a constant, we can set η equal to 1 without loss of generality.

- Interpretation: cycle through the training samples in turn
  - If misclassified, for class C_1 add $\mathbf{x}_n$ to $\mathbf{w}$
  - If misclassified, for class C_2 subtract $\mathbf{x}_n$ from $\mathbf{w}$



Black arrow: $\mathbf{w}$ (points towards the decision region of the red class), green point: misclassified

Effect of a single update: reduce the error from a misclassified sample

$$
\begin{aligned}
-\left(\mathbf{w}^{(\tau+1)}\right)^{\top}\mathbf{x}_n t_n &= -\left(\mathbf{w}^{(\tau)}\right)^{\top}\mathbf{x}_n t_n - \underbrace{(\mathbf{x}_n t_n)^{\top}\mathbf{x}_n t_n}_{\|\mathbf{x}_n t_n\|^2>0} \\
&< -\left(\mathbf{w}^{(\tau)}\right)^{\top}\mathbf{x}_n t_n
\end{aligned}
$$

- Not imply that the contribution to the error function from the other misclassified samples will have been reduced
- No guarantee to reduce the total error function at each stage

If there exists an exact solution, it is guaranteed to find it in a finite number of steps.

Linear Basis Function Models

Linear Basis Function Models

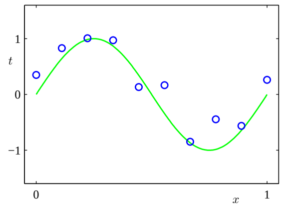
- Linear regression: simplest model for regression
  - Linear combination of input variables

$$y(\mathbf{x}, \mathbf{w}) = \sum_{d=1}^{D} w_d x_d + w_0$$

  - Limited as practical techniques for pattern recognition (*e.g.*, high dimensionality)
  - Nice analytical properties; foundation for more sophisticated models

- More useful form: polynomial curve fitting

$$y(x, \mathbf{w}) = \sum_{j=1}^{M-1} w_j x^j + w_0$$



- Linear combination of non-linear functions of input variables $\mathbf{x}$, called '*basis functions*'

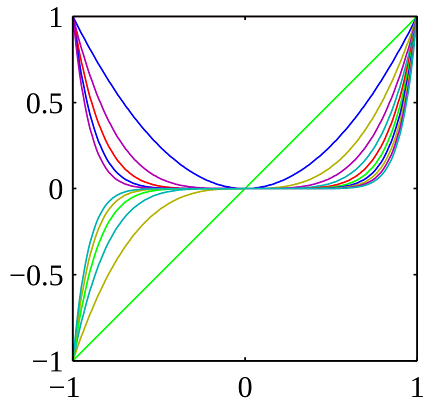
$$\begin{aligned} y(x, \mathbf{w}) &= \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) + w_0 = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) \\ &= \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) \end{aligned}$$

where $\mathbf{w} = [w_0, w_1, \ldots, w_{M-1}]$, $\boldsymbol{\phi}(\mathbf{x}) = [\phi_0 = 1, \phi_1, \ldots, \phi_{M-1}]$
  - $\boldsymbol{\phi}(\mathbf{x})$: fixed preprocessing or feature extraction

Linear functions of parameters (still analytic); Yet, non-linear with respect to the input variables

(Recap.) polynomial curve fitting

$$y(x, \mathbf{w}) = \sum_{j=1}^{M-1} w_j x^j + w_0$$



- Global function of the input variables: changes in one region of input space affect all other regions
- Difficult to formulate: number of polynomials/coefficients increases exponentially with M

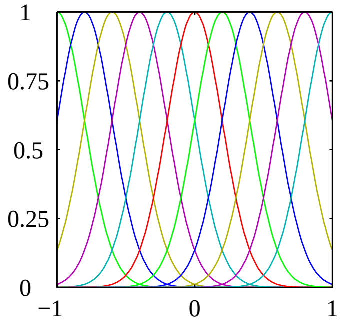
Divide the input space into regions and use different polynomials in each region!!!

Other Basis Functions

- (Gaussian) Radial Basis Functions (RBF)

$$\phi_j(x) = \exp\left\{\frac{(x-\mu_j)^2}{2s^2}\right\}$$

  - μ_j: governing the locations of the basis functions in input space
    - Can be arbitrary points in the data
  - s: governing the spatial scale
    - Can be chosen from the data set, *e.g.*, average variance



- Not required to have a probabilistic interpretation (normalization term is unimportant)

From Linear Regression to Linear Classification

- Linear regression model $y(\mathbf{x}, \mathbf{w})$

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

- For classification, we wish to obtain a discrete output or posterior probabilities in a range $(0, 1)$

$$y(\mathbf{x}) = f\left(\mathbf{w}^\top \mathbf{x} + w_0\right)$$

Generalized Linear Model

$$y(\mathbf{x}) = f\left(\mathbf{w}^\top \mathbf{x} + w_0\right)$$

- $f(\cdot)$: nonlinear, known as the activation function in machine learning
- f^{-1}: known as a link function in statistics
- Decision surfaces
  - $y(\mathbf{x}) =$ constant or $\mathbf{w}^\top \mathbf{x} + w_0 =$ constant
- Decision surfaces are linear functions of $\mathbf{x}$ even if $f(\cdot)$ is nonlinear (generalized linear model) [McCullagh and Nelder, 1989]
- Nonlinear in the parameter space $\mathbf{w}$ due to the nonlinear function $f(\cdot)$
  - Leads to more complex models for classification than regression

Probabilistic Models

Discriminative vs. Generative

Discriminative models (1-step)

- Directly infer posterior probabilities $P(C_k|\mathbf{x})$

Generative models (2-step)

1. Infer class-conditional densities $p(\mathbf{x}|C_k)$ and priors $P(C_k)$
2. Use a Bayes rule to determine posterior probabilities

$$P(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)\,P(C_k)}{p(\mathbf{x})}$$

In both cases, use a decision theory to assign each new $\mathbf{x}$ to a class

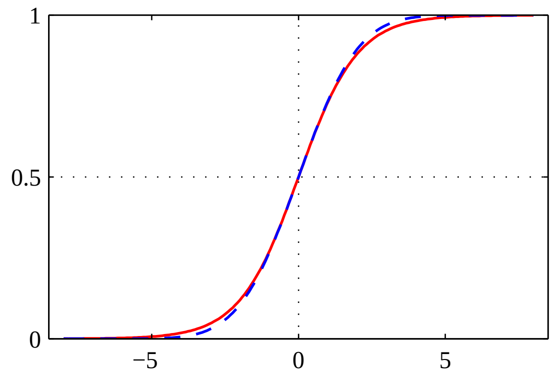
Posterior for class C_1 (in binary classification)

$$\begin{aligned} P(C_1|\mathbf{x}) &= \frac{p(\mathbf{x}|C_1)\,P(C_1)}{p(\mathbf{x}|C_1)\,P(C_1) + p(\mathbf{x}|C_2)\,P(C_2)} \\ &= \frac{1}{1+\exp(-a)} = \sigma(a) \end{aligned}$$

where $a = \ln \frac{p(\mathbf{x}|C_1)\,P(C_1)}{p(\mathbf{x}|C_2)\,P(C_2)}$

Logistic Sigmoid Function

- Sigmoid: "S"-shaped, squashing real axis into a finite interval
  - Maps real $a \in (-\infty, \infty)$ to a finite interval of $(0, 1)$



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

$\sigma(-a) = 1 - \sigma(a)$

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

$a = \ln\left(\frac{\sigma}{1-\sigma}\right)$ known as *logit* (inverse of the logistic sigmoid; log of the ratio of probabilities)

The dashed blue line is a scaled probit function (cdf of a zero-mean unit variance Gaussian).

Softmax Function

Generalization of a logistic sigmoid function for $K > 2$

$$P(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)\,P(C_k)}{\sum_j p(\mathbf{x}|C_j)\,P(C_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where $a_k = \ln p(\mathbf{x}|C_k)\,P(C_k)$

- Softmax: smoothed version of the 'max' function
  - If $a_k \gg a_j$ for all $j \neq k$, then $p(C_k|\mathbf{x}) \simeq 1$ and $p(C_j|\mathbf{x}) \simeq 0$

Probabilistic Discriminative Models

(Two-Class) Logistic Regression

- (From generative model) Posterior probability of class C_1: a logistic sigmoid acting on a linear function of the feature vector ϕ

$$P(C_1|\phi) = y(\phi) = \sigma\left(\mathbf{w}^\top \phi\right) \qquad P(C_2|\phi) = 1 - P(C_1|\phi)$$

  - $\sigma(\cdot)$: logistic sigmoid function

- In the terminology of statistics, known as '*logistic regression*'
  - This is a model for classification rather than regression.

For M-dimensional feature space ϕ

- M adjustable parameters
- *c.f.*, Generative with Gaussians: $M(M+5)/2+1$ growing quadratically
  - $2M$ parameters for means
  - $M(M+1)/2$ parameters for a shared covariance matrix
  - 2 parameters for class priors

[Determining Parameters of Logistic Regression]

For a dataset $\{\phi_n = \mathbf{x}_n, t_n\}$, where $t_n \in \{0, 1\}$ with $n = 1, \ldots, N$

- Likelihood function

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

  - $y_n = P(C_1|\phi_n) = \sigma(a_n)$, where $a_n = \mathbf{w}^\top \phi_n$
  - $\mathbf{t} = (t_1, \ldots, t_N)^\top$

- Taking negative logarithm $\rightarrow$ *cross-entropy error function*

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N} \left\{t_n \ln y_n + (1 - t_n) \ln (1 - y_n)\right\}$$

- Taking the gradient of the error function w.r.t. $\mathbf{w}$ [Quiz!!!]

$$\nabla E(\mathbf{w}) = ?$$

$$(\text{Tip}) \ \frac{\partial \sigma}{\partial a} = \sigma(1-\sigma)$$

[Quiz solution]

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

$$\text{where } y_n = \sigma(a_n) \qquad a_n = \mathbf{w}^\top \phi_n$$

$$\frac{\partial E_n}{\partial \mathbf{w}} = \frac{\partial E}{\partial y_n}\frac{\partial y_n}{\partial a_n}\frac{\partial a_n}{\partial \mathbf{w}} \quad \text{(chain rule)}$$

$$\frac{\partial E_n}{\partial y_n} = -\frac{t_n}{y_n} + \frac{1-t_n}{1-y_n} = \frac{y_n - t_n}{y_n(1-y_n)}$$

$$\frac{\partial y_n}{\partial a_n} = y_n(1-y_n)$$

$$\frac{\partial a_n}{\partial \mathbf{w}} = \phi_n$$

$$\therefore \frac{\partial E}{\partial \mathbf{w}} = \sum_{n=1}^{N} \frac{\partial E_n}{\partial \mathbf{w}} = \sum_{n=1}^{N} (y_n - t_n)\phi_n$$

- Taking the gradient of the error function w.r.t. $\mathbf{w}$

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} \underbrace{(y_n - t_n)}_{\text{error}} \phi_n \qquad \nabla E_n(\mathbf{w}) = (y_n - t_n)\phi_n$$

  - Contribution to gradient by data point n is given by an error between a target t_n and the prediction y_n times basis ϕ_n
  - The same form as the gradient of the sum-of-squares error function for the linear regression model

- In linear regression, the maximum likelihood solution, on the assumption of a *Gaussian noise model*, leads to a closed-form solution.
  - Due to a consequence of quadratic dependence of the log likelihood on the parameter vector $\mathbf{w}$

$$\nabla_{\mathbf{w}}^{R} \ln p(\mathbf{t}|\mathbf{w}, \beta) \Rightarrow \mathbf{w}_{ML}^{R} = \left(\Phi^\top \Phi + \lambda I\right)^{-1} \Phi^\top \mathbf{t}$$

- For logistic regression, there is no closed-form maximum likelihood solution.
  - Due to the *nonlinearity* of the logistic sigmoid function

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\,\phi_n = \sum_{n=1}^{N} \nabla E_n$$

- Iterative approach

$$\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} - \eta \nabla E(\mathbf{w})$$

- Alternative sequential iteration

$$\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} - \eta \nabla E_n(\mathbf{w})$$

▸ Iterative Reweighted Least Squares (IRLS)

Multiclass Logistic Regression

Work with a softmax function instead of logistic sigmoid

$$P(C_k|\mathbf{x}) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \quad \text{where } a_k = \mathbf{w}_k^\top \phi$$

- Likelihood function

$$p(\mathbf{T}|\mathbf{w}_1, \ldots, \mathbf{w}_K) = \prod_{n=1}^{N} \prod_{k=1}^{K} P(C_k|\mathbf{x}_n)^{t_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

  - $\mathbf{t}_n \in \{0, 1\}^K$: 1-of-$K$ coding scheme
  - $y_{nk} = y_k(\phi_n)$, $\mathbf{T} = [t_{nk}]$: $N \times K$ matrix of target variables

- Taking negative logarithm $\rightarrow$ *cross-entropy error function*

$$E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{w}_1, \ldots, \mathbf{w}_K) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

[Determining Parameters of Multiclass Logistic Regression]

- Taking the gradient of the error function w.r.t. $\mathbf{w}_j$ [Quiz!!!]

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = ?$$

$$\text{(Tip)} \quad \frac{\partial}{\partial x}\frac{f(x)}{g(x)} = \frac{f'(x)\,g(x) - f(x)\,g'(x)}{g(x)^2}$$

[Quiz solution]

$$E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{w}_1, \ldots, \mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

$$\frac{\partial E}{\partial \mathbf{w}_j} = \frac{\partial E}{\partial y_{nk}} \frac{\partial y_{nk}}{\partial a_{nj}} \frac{\partial a_{nj}}{\partial \mathbf{w}_j} \quad (\because \text{ chain rule})$$

$$\frac{\partial E}{\partial y_{nk}} = -\sum_{n=1}^{N}\sum_{k=1}^{K} \frac{t_{nk}}{y_{nk}} \qquad \frac{\partial a_{nj}}{\partial \mathbf{w}_j} = \phi_n$$

$$\begin{aligned}
\frac{\partial y_{nk}}{\partial a_{nj}} &= \left\{ \begin{array}{ll} (k=j) & \frac{\partial y_{nk}}{\partial a_{nk}} = \frac{\exp(a_k)}{\sum_i \exp(a_i)} - \left(\frac{\exp(a_k)}{\sum_i \exp(a_i)}\right)^2 = y_{nk}(1-y_{nk}) \\ (k \neq j) & \frac{\partial y_{nk}}{\partial a_{nj}} = -\frac{\exp(a_k)\exp(a_j)}{\left(\sum_i \exp(a_i)\right)^2} = -y_{nk}y_{nj} \end{array} \right\} \\
&= y_{nk}\left(I_{kj} - y_{nj}\right) \qquad \text{where, } I_{kj} = \left\{ \begin{array}{ll} 1 & \text{if } k=j \\ 0 & k \neq j \end{array} \right.
\end{aligned}$$

$$\begin{aligned}
\frac{\partial E}{\partial \mathbf{w}_j} &= -\sum_{n=1}^{N}\sum_{k=1}^{K} \frac{t_{nk}}{y_{nk}} y_{nk}\left(I_{kj} - y_{nj}\right)\phi_n \\
&= -\sum_{n=1}^{N}\sum_{k=1}^{K} \left(t_{nk}I_{kj} - t_{nk}y_{nj}\right)\phi_n = \sum_{n=1}^{N}\left(y_{nj} - t_{nj}\right)\phi_n
\end{aligned}$$

[Determining Parameters of Multiclass Logistic Regression]

- Taking the gradient of the error function w.r.t. $\mathbf{w}_j$

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1, \ldots, \mathbf{w}_K) = \sum_{n=1}^{N} \underbrace{(y_{nj} - t_{nj})}_{\text{error}} \phi_n$$

  - Contribution to gradient by data point n is given by an error between a target t_{nj} and prediction y_{nj} times basis ϕ_n

- Making use of it to give a sequential algorithm in which patterns are presented one at a time

$$\mathbf{w}_j^{(\text{new})} = \mathbf{w}_j^{(\text{old})} - \eta \nabla_j E_n(\mathbf{w}_1, \ldots, \mathbf{w}_K)$$

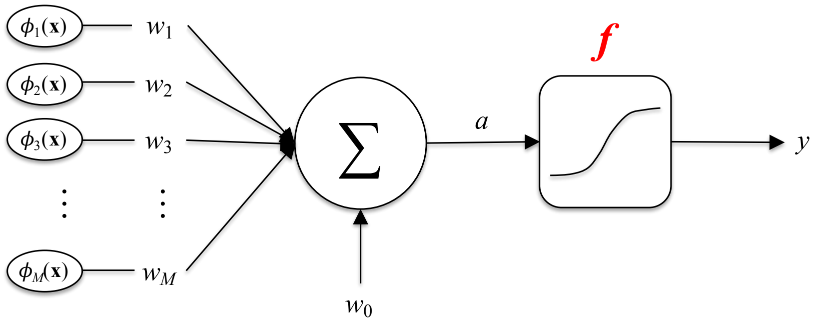
  - Update should be conducted for all parameters $\{\mathbf{w}_j\}$ simultaneously

Multiclass IRLS

Modified Perceptron: Two-Class

$$y(\mathbf{x}) = f\left(\underbrace{\mathbf{w}^\top \mathbf{x} + w_0}_{=a}\right)$$

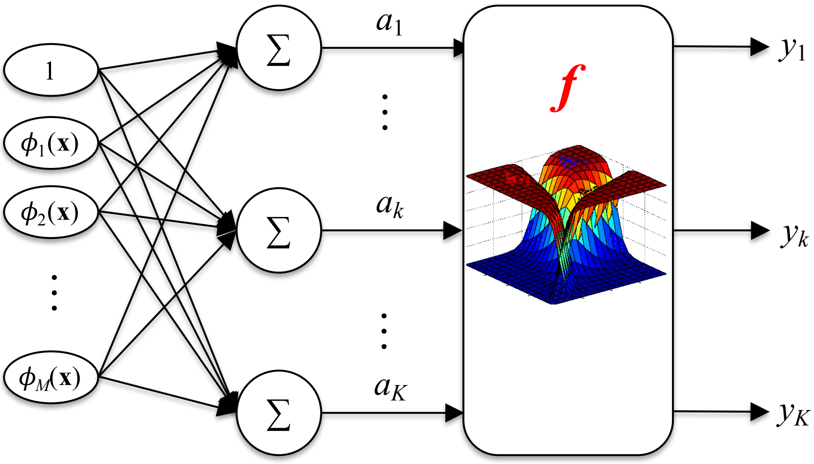
where $f(a) = \dfrac{1}{1 + \exp(-a)}$



Modified Perceptron: Multi-Class

$$y_k\left(\phi\left(\mathbf{x}\right)\right) = f\left(\underbrace{\mathbf{w}_k^\top \phi\left(\mathbf{x}\right) + w_0}_{=a_k}\right)$$

where $f\left(a_k\right) = \dfrac{\exp\left(a_k\right)}{\sum_j \exp\left(a_k\right)}$



Thank you for your attention!!!

(Q & A)

hisuk (AT) korea.ac.kr

http://www.ku-milab.org