Lectures 6 and 7: Logistic Regression & Decision-based Learning (Perceptrons) & Decision Trees

Prof. Krishna R. Pattipati
Dept. of Electrical and Computer Engineering
University of Connecticut
Contact: krishna@engr.uconn.edu (860) 486-2890

Fall 2018
October 22 & 29, 2018

Reading List

- Murphy Chapters 8, 21.6-21.8
- Ripley (LP formulation of Perceptron)
- Bishop, Chapter 4, Chapter 10.6
- S.Y. Kung, Fuzzy Perceptrons
- Sergios Theodiridis, Chapter 7
- Duda, Hart and Stork, Chapter 8.1-8.4

Lecture Outline

- Classification as a Function Approximation Problem
  - Discuss Generative versus Discriminative Learning
- Approximating Posterior Probabilities
  - Gaussian Case: Two Class and Multiple Class Cases
- Decision Based Learning
- Perceptron Convergence Theorem
- Optimal Learning Rate & Normalized Perceptron
- Extension to Multiple Classes
- Fuzzy Updates, LVQ as a Perceptron Update
- Decision Trees and Regularization
- Bagging, Random Forests and Gradient-based Boosting
- Summary

❑ Classification as a Function Approximation Problem

Define a target vector $\underline{z}_k = \underline{e}_k = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \Rightarrow z = k$; Alternate $\underline{z}_k = \begin{bmatrix} -1/(C-1) \\ \vdots \\ 1 \\ \vdots \\ -1/(C-1) \end{bmatrix}$

want to find a mapping $\hat{\underline{z}} = \underline{y}(\underline{x})$ ə Mean Square Error is a min.

$$J = \frac{1}{2} E\left\{(\underline{y}(\underline{x}) - \underline{z})^T (\underline{y}(\underline{x}) - \underline{z})\right\} = \frac{1}{2} \int_{\underline{x}} \sum_{\underline{z}} (\underline{y}(\underline{x}) - \underline{z})^T (\underline{y}(\underline{x}) - \underline{z}) p(\underline{x}, \underline{z}) d\underline{x}$$

$$= \frac{1}{2} \int_{\underline{x}} \sum_{\underline{z}} (\underline{y}(\underline{x}) - \underline{z})^T (\underline{y}(\underline{x}) - \underline{z}) P(\underline{z}/\underline{x}) p(\underline{x}) d\underline{x}$$

$$\nabla_{\underline{y}} J = 0 \Rightarrow \sum_{\underline{z}} (\underline{y}(\underline{x}) - \underline{z}) P(\underline{z}/\underline{x}) = 0 \quad \boxed{\Rightarrow \underline{y}(\underline{x}) = \sum_{\underline{z}} \underline{z} P(\underline{z}/\underline{x}) = E[\underline{z}/\underline{x}] = \hat{\underline{z}}(\underline{x})}$$

conditional mean

Classification as a Function Approximation Problem-2

- **MMSE estimate = Conditional Mean**

$$\underline{y}(\underline{x}) = \sum_{k=1}^{C} \underline{e}_k P(z=k \mid \underline{x}) = \underline{P} = \begin{bmatrix} P(z=1 \mid \underline{x}) \\ P(z=2 \mid \underline{x}) \\ \\ P(z=C \mid \underline{x}) \end{bmatrix}$$

$$\textit{Alternate form for } \underline{z}_k : \underline{y}(\underline{x}) = \sum_{k=1}^{C} \underline{z}_k P(z=k \mid \underline{x}) = \begin{bmatrix} \frac{1}{C-1}[CP(z=1 \mid \underline{x}) - 1] \\ \frac{1}{C-1}[CP(z=2 \mid \underline{x}) - 1] \\ . \\ \frac{1}{C-1}[CP(z=C \mid \underline{x}) - 1] \end{bmatrix}$$

$$\underline{z}_k = \begin{bmatrix} -1/(C-1) \\ \vdots \\ 1 \\ \vdots \\ -1/(C-1) \end{bmatrix}$$

$$\textit{Binary} \Rightarrow \begin{bmatrix} P(z=1 \mid \underline{x}) - P(z=2 \mid \underline{x}) \\ P(z=2 \mid \underline{x}) - P(z=1 \mid \underline{x}) \end{bmatrix}$$

Minimum Distance versus Maximum a Posteriori Decisions

- Minimum distance versus Maximum a Posteriori decisions

Select class i if $\|\underline{P} - \underline{z}_k\|^2$ is a min

$$d_k = (\underline{P} - \underline{z}_k)^T (\underline{P} - \underline{z}_k) = \|\underline{P}\|^2 - 2\underline{P}^T \underline{z}_k + \underline{z}_k{}^T \underline{z}_k = 1 - 2P(z = k \mid \underline{x}) + \|\underline{P}\|^2$$

$$Alternate: d_k = (\underline{y}_k - \underline{z}_k)^T (\underline{y}_k - \underline{z}_k) = \frac{C}{C-1}[1 - P(z = k \mid \underline{x}) + C\|\underline{P}\|^2]$$

$$\min_k d_k \Rightarrow \max_k P(z = k \mid \underline{x})$$

- Learning posterior probabilities directly … **Discriminative Learning**

$P(z = k \mid \underline{x}) \sim$ **posterior probability of class** k **given** $\underline{x}$

$$P(z = k \mid \underline{x}) = \frac{P(\underline{x} \mid z = k)P(z = k)}{\sum_{i=1}^{C} P(\underline{x} \mid z = i)P(z = i)}$$

MAP Decisions: Gaussian Case

- *Gaussian case*

$$P(z=k \mid \underline{x}) = \frac{\frac{1}{|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(\underline{x}-\underline{\mu}_k)^T \Sigma_k^{-1}(\underline{x}-\underline{\mu}_k)} P(z=k)}{\sum_{i=1}^{C} \frac{1}{|\Sigma_i|^{1/2}} e^{-\frac{1}{2}(\underline{x}-\underline{\mu}_i)^T \Sigma_i^{-1}(\underline{x}-\underline{\mu}_i)} P(z=i)}$$

$$= \frac{e^{[-\frac{1}{2}(\underline{x}-\underline{\mu}_k)^T \Sigma_k^{-1}(\underline{x}-\underline{\mu}_k) - \frac{1}{2}\ln|\Sigma_k| + \ln P(z=k)]}}{\sum_{i=1}^{C} e^{[-\frac{1}{2}(\underline{x}-\underline{\mu}_i)^T \Sigma_i^{-1}(\underline{x}-\underline{\mu}_i) - \frac{1}{2}\ln|\Sigma_i| + \ln P(z=i)]}}$$

$$= \frac{e^{y(\underline{x},\underline{\theta}_k)}}{\sum_{i=1}^{C} e^{y(\underline{x},\underline{\theta}_i)}}$$

Why learn $\underline{\theta}$? $y(\underline{x}) = \underline{x}^T A_i \underline{x} + \underline{b}_i^T \underline{x} + c_i$

Why not learn weights directly?

$$\underline{\theta} = \Big[\underbrace{\{\underline{\mu}_k\}, \{\Sigma_k\}, P\{z=k\}}_{\underline{\theta}_k}\Big]$$

"soft max"
"multiple logistic model"
"logistic discrimination"

$y(\underline{x}, \underline{\theta}_k) = g_k(\underline{x})$
discriminant function

MAP Decisions : Gaussian with Equal Covariances

- *Equal covariance case*

$$P(z=k \mid \underline{x}) = \frac{e^{\underline{\mu}_k{}^T \Sigma^{-1} x - \frac{1}{2}\underline{\mu}_k{}^T \Sigma^{-1} \underline{\mu}_k + \ln P(z=k)}}{\sum_{i=1}^{C} e^{\underline{\mu}_i{}^T \Sigma^{-1} x - \frac{1}{2}\underline{\mu}_i{}^T \Sigma^{-1} \underline{\mu}_i + \ln P(z=i)}}$$

$$= \frac{e^{y(\underline{x}, \underline{w}_k)}}{\sum_{i=1}^{C} e^{y(\underline{x}, \underline{w}_i)}}$$

where $y(\underline{x}, \underline{w}_k) = \underline{w}_k{}^T \underline{x} - w_{k0}$

$$\underline{w}_k = \Sigma^{-1} \underline{\mu}_k \qquad w_{k0} = \frac{1}{2} \underline{\mu}_k{}^T \Sigma^{-1} \underline{\mu}_k - \ln P(z=k)$$

MAP Decisions: Gaussian, Equal Σ, Two Class Case

- *Two class case*

  Let the classes be $z \in \{0,1\}$

Note : If $z \in \{-1,1\}$,

$$P(z \mid \underline{x}, \underline{w}) = \frac{1}{1+e^{-zy}} = g(y,z)$$

$$\mu = P(z=1 \mid \underline{x}, \underline{w}_1, \underline{w}_0) = \frac{e^{y(\underline{x},\underline{w}_1)}}{e^{y(\underline{x},\underline{w}_1)} + e^{y(\underline{x},\underline{w}_0)}} = \frac{1}{1+e^{-(y(\underline{x},\underline{w}_1)-y(\underline{x},\underline{w}_0))}}$$

$$= \frac{1}{1+e^{-y}} = g(y) = \mu \qquad \frac{dg}{dy} = g(1-g)$$

where $y = y(\underline{x},\underline{w}_1) - y(\underline{x},\underline{w}_0) = \underline{w}_1^T \underline{x} - w_{10} - \underline{w}_0^T \underline{x} + w_{00}$

$$\frac{\mu}{1-\mu} = e^y$$

$$= (\underline{w}_1 - \underline{w}_0)^T \underline{x} - (w_{10} - w_{00}) = \tilde{\underline{w}}^T \underline{x} - \tilde{w}_0 = \underline{w}^T \underline{x}_a \qquad \underline{x}_a = \begin{bmatrix} -1 \\ \underline{x} \end{bmatrix}$$

$$P(z \mid \underline{x}, \underline{w}) = \mu^z (1-\mu)^{1-z} = \left(\frac{1}{1+e^{-y}}\right)^z \left(\frac{1}{1+e^{y}}\right)^{1-z} = \frac{1}{1+e^{-(2z-1)y}} = e^{yz} g(-y)$$

One of the classes can be taken as a reference and set its weights to 0!

The parameters $\tilde{\underline{w}}, \tilde{w}_0$ are directly estimated from the data. Assume $\underline{x}$ is augmented.

Maximum Likelihood Estimation of Weights

- Suppose have data

$$D = \left\{ \{\underline{x}^1, z^1\}\ \{\underline{x}^2, z^2\}, \cdots\cdots, \{\underline{x}^N, z^N\} :\ z^n \in \{0,1\} \right\}$$

- Likelihood

$$L(\underline{w}) = P(\{z^n\}_{n=1}^N \mid \{\underline{x}^n\}_{n=1}^N, \underline{w}) = \prod_{n=1}^N P(z^n \mid \underline{x}^n, \underline{w}) = \prod_{n=1}^N (\mu_n)^{z^n} (1-\mu_n)^{1-z^n}$$

Recall convexity of Cross Entropy, $-J_n$

$$J(\underline{w}) = -\ln L(\underline{w}) = NLL(\underline{w}) = -\sum_{n=1}^N J_n ; J_n = [z^n \ln \mu_n + (1-z^n)\ln(1-\mu_n)] \ldots . - ve\ Cross\ Entropy$$

$$\nabla_{\underline{w}} J = -\sum_{n=1}^N \underbrace{\frac{\partial J_n}{\partial \mu_n}}_{\frac{z^n-\mu_n}{\mu_n(1-\mu_n)}} \underbrace{\frac{\partial \mu_n}{\partial y_n}}_{\mu_n(1-\mu_n)} \underbrace{\nabla_{\underline{w}} y_n}_{\underline{x}^n} = \sum_{n=1}^N \left(\mu_n - z^n\right) \underline{x}^n = \underbrace{X^T}_{(p+1)\text{x}N} \underbrace{\left(\underline{\mu} - \underline{z}\right)}_{N\text{x}1}$$

$$X = \begin{bmatrix} (\underline{x}^1)^T \\ (\underline{x}^2)^T \\ . \\ (\underline{x}^N)^T \end{bmatrix}; N\text{x}(p+1)$$

$$\nabla^2_{\underline{w}} J = X^T \Sigma X > 0 \ where\ \Sigma = Diag[\mu_n(1-\mu_n)]$$

Re *gularize by adding* $\lambda \underline{w}^T \underline{w}$ *or* $\lambda \| \underline{w} \|_1$ *to* $J(\underline{w})$

- Optimization Methods: SD, Newton, Levenberg-Marquardt, CG, QN,...

Iteratively Reweighted Least Squares (IRLS)

- Newton's Method

$$\underline{w}_{i+1} = \underline{w}_i - \left(\nabla^2_{\underline{w}_i} J\right)^{-1} \nabla_{\underline{w}_i} J$$

$$= \underline{w}_i + \left(X^T \Sigma_i X\right)^{-1} X^T (\underline{z} - \underline{\mu}_i)$$

$$= \left(X^T \Sigma_i X\right)^{-1} \left[X^T \Sigma_i X \underline{w}_i + X^T (\underline{z} - \underline{\mu}_i)\right]$$

$$= \left(X^T \Sigma_i X\right)^{-1} X^T \Sigma_i \underbrace{\left[X \underline{w}_i + \Sigma_i^{-1} (\underline{z} - \underline{\mu}_i)\right]}_{\underline{y}_i}$$

$$y_{in} = \underline{w}_i^T \underline{x}^n + \frac{z^n - \mu_{in}}{\mu_{in}(1-\mu_{in})}$$

- This corresponds to the following **weighted least squares** problem at iteration *i*

$$\min J_i = \frac{1}{2} \left(\underline{y}_i - X \underline{w}\right)^T \Sigma_i \left(\underline{y}_i - X \underline{w}\right)$$

- Weighted least squares can be implemented in recursive way…. One sample at a time.

Multi-class Case

- Convert the problem to one versus rest problem (one-hot coding)
- Multiclass problems
  - *M* class problem: *M* two-class tasks or *M(M-1)/2* two-class tasks
  - Obtain a separate classifier for each pair
  - What if more than one decision rule classifies? **Majority rule**
- Error Correcting Output Codes
  - Each class: an *n* bit pattern chosen so that Hamming distance is as large as possible among classes
  - Learn each of the *n* bits via binary classifiers
  - Select nearest class (i.e., one with the least Hamming distance)
- Work Directly with Multi-class formulation
  - Direct extension of the binary case
  - Upper and lower bounds on sigmoid function
- Laplace approximation of posterior to quantify uncertainty in predictions

Multiple Class Case-1

- What to do in multiple class case ?

$$Let\ z_k^n = \begin{cases} 1\ if\ \ z^n = k \\ 0;\ otherwise \end{cases} \quad ....\ one\ of\ C\ coding$$

$$P(z_k^n = 1 \mid \underline{x}^n, \{\underline{w}_i\}_{i=1}^C) = \frac{e^{y(\underline{x}^n, \underline{w}_k)}}{\sum_{i=1}^{C} e^{y(\underline{x}^n, \underline{w}_i)}} = \frac{e^{\underline{w}_k^T \underline{x}^n}}{\sum_{i=1}^{C} e^{\underline{w}_i^T \underline{x}^n}} = \mu_{nk} \ ...\ soft\ \max\ function$$

$$Use\ class\ C\ as\ a\ reference.\ So,\ set\ \underline{w}_C = \underline{0} \Rightarrow W^T = \begin{bmatrix} \underline{w}_1^T \\ \underline{w}_2^T \\ . \\ \underline{w}_{C-1}^T \end{bmatrix}; (C-1)\mathrm{x}(p+1)$$

$$L = p(\{z^n\}_{n=1}^N \mid \{\underline{x}^n\}_{n=1}^N, W) = \prod_{n=1}^{N} \left(\prod_{k=1}^{C-1} (\mu_{nk})^{z_k^n} \right) \left(1 - \sum_{i=1}^{C-1} \mu_{ni} \right)^{\left(1 - \sum_{i=1}^{C-1} z_i^n\right)}$$

Multiple Class Case-2

$$J = -\ln L = \sum_{n=1}^{N}\left[\left\{-\sum_{k=1}^{C-1} z_k^n \ln\left(\frac{\mu_{nk}}{1-\sum_{i=1}^{C-1}\mu_{ni}}\right)\right\} - \ln\left(1-\sum_{i=1}^{C-1}\mu_{ni}\right)\right]$$

$$Note: \frac{\mu_{nk}}{1-\sum_{i=1}^{C-1}\mu_{ni}} = e^{\underline{w}_k^T \underline{x}^n} \text{ and } \left(1-\sum_{i=1}^{C-1}\mu_{ni}\right) = \left(1+\sum_{i=1}^{C-1} e^{\underline{w}_i^T \underline{x}^n}\right)^{-1}; \text{Recall } \underline{w}_C = \underline{0}$$

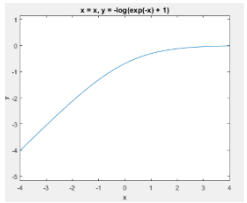
$$\Rightarrow J = \sum_{n=1}^{N}\left[\left(-\sum_{k=1}^{C-1} z_k^n \underline{w}_k^T \underline{x}^n\right) + \ln\left(1+\sum_{i=1}^{C-1} e^{\underline{w}_i^T \underline{x}^n}\right)\right]$$

$$\nabla_{\underline{w}_k} J = \sum_{n=1}^{N}\left(\mu_{nk} - z_k^n\right)\underline{x}^n \because \nabla_{\underline{w}_k} \ln\left(1+\sum_{i=1}^{C-1} e^{\underline{w}_i^T \underline{x}^n}\right) = \left(\frac{e^{\underline{w}_k^T \underline{x}^n}}{1+\sum_{i=1}^{C-1} e^{\underline{w}_i^T \underline{x}^n}}\right)\underline{x}^n = \mu_{nk}\underline{x}^n$$

$$\nabla^2_{\underline{w}_k \underline{w}_k} J = \sum_{n=1}^{N} \mu_{nk}\left(1-\mu_{nk}\right)\underline{x}^n\left(\underline{x}^n\right)^T; \nabla^2_{\underline{w}_k \underline{w}_j} J = -\sum_{n=1}^{N} \mu_{nk}\mu_{nj}\underline{x}^n\left(\underline{x}^n\right)^T; k \neq j$$

Big Matrix

Variational Upper Bound on Sigmoid



- Upper bounds on sigmoid function (helps when C is large)

Consider the problem of maximizing $\ln g(x) = -\ln(1+e^{-x})$ ... *concave in* x

Let $y = e^{-x} \Rightarrow \ln y = -x$

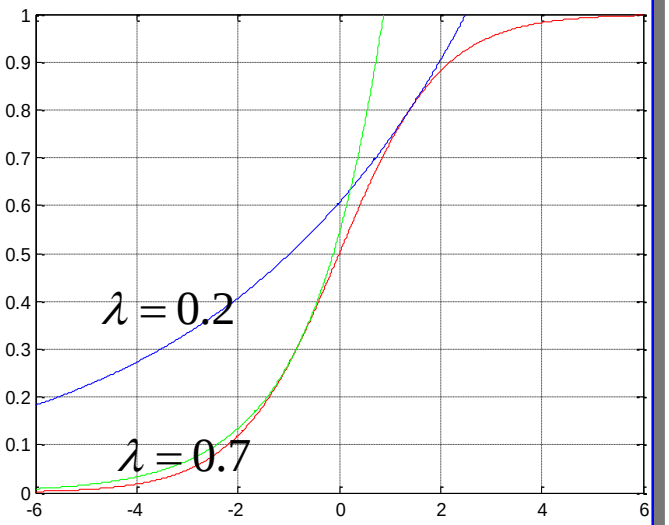
Consider $\max\{-\ln(1+y)\}$ *subject to* $\ln y + x = 0$

$$L(x, y, \lambda) = -\ln(1+y) + \lambda(\ln y + x) \Rightarrow y^* = \frac{\lambda}{1-\lambda}$$

$$L(x, y^*, \lambda) = \underbrace{(1-\lambda)\ln(1-\lambda) + \lambda \ln \lambda}_{-H(\lambda)} + \lambda x$$

$$\Rightarrow \ln g(x) \le \min_{\lambda} \left[\underbrace{(1-\lambda)\ln(1-\lambda) + \lambda \ln \lambda}_{-H(\lambda)} + \lambda x \right] \forall x$$

So, $g(x) \le \exp(\lambda x - H(\lambda)) \; \forall x, \lambda$



This can be used to obtain an upper bound on LL and use primal-dual methods for optimization. We will illustrate its application in the multi-class case.

Log-Sum-Exponent bound & Multi-class Case

- Recall

$$l = \ln p(\{z^n\}_{n=1}^N | \{\underline{x}^n\}_{n=1}^N, W) = \sum_{n=1}^{N}\left[\left(\sum_{k=1}^{C-1} z_k^n \underline{w}_k^T \underline{x}^n\right) - \ln\left(1+\sum_{i=1}^{C-1} e^{\underline{w}_i^T \underline{x}^n}\right)\right]$$

& $-\ln(1+e^x)$ *is concave in* x

Consider $\max -\{\ln(1+\sum_{i=1}^{C-1} e^{y_i})\} \Rightarrow$ *Lagrangian* $L(\{y_i\},\lambda,v) = -\ln(1+v) + \lambda(v - \sum_{i=1}^{C-1} e^{y_i})$

$$\Rightarrow v = (1-\lambda)/\lambda \Rightarrow -\ln(1+\sum_{i=1}^{C-1} e^{y_i}) \le \ln\lambda + 1 - \lambda(1+\sum_{i=1}^{C-1} e^{y_i})$$

$$so, -\ln\left(1+\sum_{i=1}^{C-1} e^{\underline{w}_i^T \underline{x}^n}\right) \le \ln\lambda_n + 1 - \lambda_n(1+\sum_{i=1}^{C-1} e^{\underline{w}_i^T \underline{x}^n})$$

$$So, l \le \sum_{n=1}^{N}\left[\left(\sum_{k=1}^{C-1} z_k^n \underline{w}_k^T \underline{x}^n\right) + \ln\lambda_n + 1 - \lambda_n(1+\sum_{i=1}^{C-1} e^{\underline{w}_i^T \underline{x}^n})\right] \cdots \text{maximize UB wrt } \{\underline{w}_k\} \text{ and minimize wrt } \lambda$$

The UB is a separable problem for each class given $\{\lambda_n\}$!

$$\tilde{l}_k = \sum_{n=1}^{N}\left[z_k^n \underline{w}_k^T \underline{x}^n - \lambda_n e^{\underline{w}_k^T \underline{x}^n}\right] \text{.....Needs NLP optimizer}$$

$$Next\ \lambda_n : \lambda_n = 1/\left(1+\sum_{i=1}^{C-1} e^{\underline{w}_i^T \underline{x}^n}\right)$$

} Iterate

Jensen + Log-normal + Variational in Multi-class Case

- Recall MAP formulation (Note: NLL)

$$J = -\ln p(\{z^n\}_{n=1}^N \mid \{\underline{x}^n\}_{n=1}^N, W) - \ln \underbrace{p(W)}_{prior}$$

$$= \sum_{n=1}^{N}\left[\left(-\sum_{k=1}^{C-1} z_k^n \underline{w}_k^T \underline{x}^n\right) + \ln\left(1+\sum_{i=1}^{C-1} e^{\underline{w}_i^T \underline{x}^n}\right)\right] - \frac{1}{2}\left(\sum_{k=1}^{C-1}\ln|\Sigma_{k0}| + (\underline{w}_k - \underline{w}_{k0})^T \Sigma_{k0}^{-1}(\underline{w}_k - \underline{w}_{k0}) + (p+1)\ln 2\pi\right)$$

Let $q(W) = \prod_{k=1}^{C-1} q_k(\underline{w}_k)$ *where* $q_k(\underline{w}_k) \sim N(\underline{w}_k; \underline{w}_{kN}, \Sigma_{kN})$; $p(\underline{w}_k) \sim N(\underline{w}_k; \underline{w}_{k0}, \Sigma_{k0})$

& $\ln(1+e^x)$ *is convex in* x

$$\Rightarrow E_q\left[\ln\left(1+\sum_{k=1}^{C-1} e^{\underline{w}_k^T \underline{x}^n}\right)\right] \le \ln\left(1+\sum_{k=1}^{C-1} E_{q_k}\left(e^{\underline{w}_k^T \underline{x}^n}\right)\right) = \ln\left(1+\sum_{k=1}^{C-1} e^{\left(\underline{w}_{kN}^T \underline{x}^n + \frac{1}{2}\underline{x}^{nT}\Sigma_{kN}\underline{x}^n\right)}\right)$$

Recall Moment Generating Function

$$so, \ln\left(1+\sum_{k=1}^{C-1} e^{\left(\underline{w}_{kN}^T \underline{x}^n + \frac{1}{2}\underline{x}^{nT}\Sigma_{kN}\underline{x}^n\right)}\right) \le \ln\lambda_n + 1 - \lambda_n\left(1+\sum_{k=1}^{C-1} e^{\left(\underline{w}_{kN}^T \underline{x}^n + \frac{1}{2}\underline{x}^{nT}\Sigma_{kN}\underline{x}^n\right)}\right)$$

$$So, J \le \sum_{n=1}^{N}\left[\left(-\sum_{k=1}^{C-1} z_k^n \underline{w}_{kN}^T \underline{x}^n\right) + \ln\lambda_n + 1 - \lambda_n\left(1+\sum_{k=1}^{C-1} e^{\left(\underline{w}_{kN}^T \underline{x}^n + \frac{1}{2}\underline{x}^{nT}\Sigma_{kN}\underline{x}^n\right)}\right)\right] +$$

$$\frac{1}{2}\sum_{k=1}^{C-1}\left(tr(\Sigma_{kN}\Sigma_{k0}^{-1}) + \ln|\Sigma_{k0}| + (\underline{w}_{kN} - \underline{w}_{k0})^T \Sigma_{k0}^{-1}(\underline{w}_{kN} - \underline{w}_{k0}) + (p+1)\ln 2\pi\right)$$

Jensen + Log-normal + Variational in Multi-class Case

$$J \le \sum_{n=1}^{N}\left[\left(-\sum_{k=1}^{C-1} z_k^n \underline{w}_{kN}^T \underline{x}^n\right) + \ln\lambda_n + 1 - \lambda_n\left(1+\sum_{k=1}^{C-1} e^{\left(\underline{w}_{kN}^T \underline{x}^n + \frac{1}{2}\underline{x}^{nT}\Sigma_{kN}\underline{x}^n\right)}\right)\right] +$$

$$\frac{1}{2}\sum_{k=1}^{C-1}\left(tr(\Sigma_{kN}\Sigma_{k0}^{-1}) + \ln|\Sigma_{k0}| + (\underline{w}_{kN}-\underline{w}_{k0})^T\Sigma_{k0}^{-1}(\underline{w}_{kN}-\underline{w}_{k0}) + (p+1)\ln 2\pi\right)$$

The LB is a separable problem for each class!

$$\tilde{J}_k = \sum_{n=1}^{N}\left[-z_k^n \underline{w}_{kN}^T \underline{x}^n - \lambda_n e^{\left(\underline{w}_{kN}^T \underline{x}^n + \frac{1}{2}\underline{x}^{nT}\Sigma_{kN}\underline{x}^n\right)}\right] + \left(tr(\Sigma_{kN}\Sigma_{k0}^{-1}) + \ln|\Sigma_{k0}| + (\underline{w}_{kN}-\underline{w}_{k0})^T\Sigma_{k0}^{-1}(\underline{w}_{kN}-\underline{w}_{k0})\right)$$

..... needs NLP

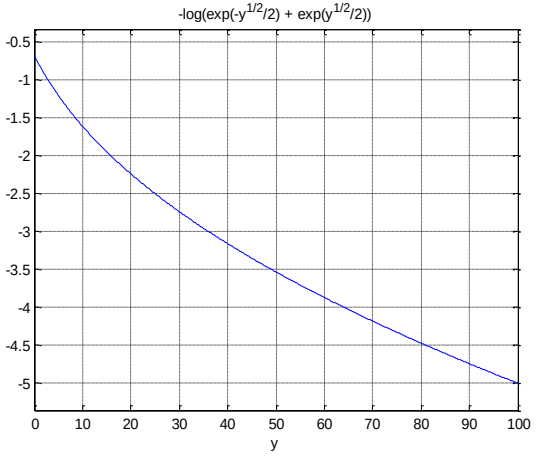
Alternately, can approximate exponential upto first order at current iteration and obtain closed form solutions.

$$Next\ \lambda_n : \lambda_n = 1/\left(1+\sum_{i=1}^{C-1} e^{\left(\underline{w}_{iN}^T \underline{x}^n + \frac{1}{2}\underline{x}^{nT}\Sigma_{kN}\underline{x}^n\right)}\right)$$

Variational Logistic Regression -1

- Lower bound on sigmoid function

$$\textit{Consider } \ln g(x) = -\ln(1+e^{-x}) = -\ln[e^{-x/2}(e^{x/2}+e^{-x/2})] = \frac{x}{2} - \ln(e^{x/2}+e^{-x/2})$$



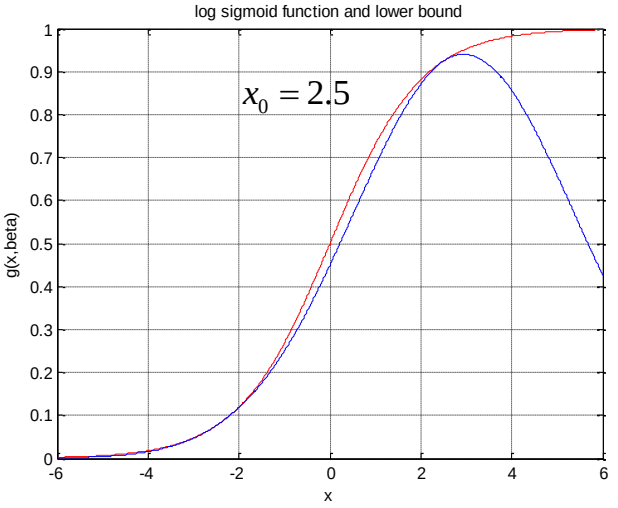
$$f(x) = -\ln(e^{x/2}+e^{-x/2}) \textit{ is convex in } x^2. \textit{Why?}$$

$$\textit{Let } x = \sqrt{y} \Rightarrow f(y) = -\ln(e^{\sqrt{y}/2}+e^{-\sqrt{y}/2}); y \geq 0$$

$$\frac{df}{dy} = \frac{-1}{4\sqrt{y}}\tanh(\frac{\sqrt{y}}{2}) < 0; \frac{d^2 f}{dy^2} = \frac{\frac{\sinh(\sqrt{y})}{y^{3/2}} - \frac{1}{y}}{8(\cosh(\sqrt{y})+1)} > 0 \; \forall y \geq 0$$

Recall for convex functions

$$f(y) \geq f(y_0) + \frac{df}{dy}|_{y=y_0}(y-y_0) \; \forall y_0 = x_0^2$$



$$\textit{So, } f(y) \geq -\ln(e^{\sqrt{y_0}/2}+e^{-\sqrt{y_0}/2}) - \frac{1}{4\sqrt{y_0}}\tanh(\frac{\sqrt{y_0}}{2})(y-y_0)$$

$$\Rightarrow f(x) \geq -\ln(e^{x_0/2}+e^{-x_0/2}) - \underbrace{\frac{1}{4x_0}\tanh(\frac{x_0}{2})}_{\lambda(x_0)}(x^2-x_0^2)$$

$$\ln g(x) \geq \frac{x-x_0}{2} + \underbrace{\frac{x_0}{2} - \ln(e^{x_0/2}+e^{-x_0/2})}_{\ln g(x_0)} - \lambda(x_0)(x^2-x_0^2)$$

$$\textit{So, } g(x) \geq g(x_0)\exp\{\frac{x-x_0}{2} - \lambda(x_0)(x^2-x_0^2)\}$$

Variational Logistic Regression - 2

- Binary (Two class) Case using local variational lower bound

Posterior distribution of z for a given x

$$y = \underline{w}^T \underline{x} \; or \; y = \underline{w}^T \phi(\underline{x})$$

$$p(z \mid \underline{w}) = g(y)^z (1 - g(y))^{1-z} = \left(\frac{1}{1+e^{-y}}\right)^z \left(\frac{e^{-y}}{1+e^{-y}}\right)^{1-z}$$

$$= e^{yz}\left(\frac{e^{-y}}{1+e^{-y}}\right) = e^{yz}\left(\frac{1}{1+e^{y}}\right) = e^{yz} g(-y)$$

$$\text{Recall } g(y) \geq g(y_0)\exp\{(y-y_0)/2 - \lambda(y_0)(y^2 - y_0^2)\}$$

$$\lambda(y_0) = \frac{1}{4y_0}\tanh(\frac{y_0}{2}) = \frac{1}{4y_0}\left(\frac{1-e^{-y_0}}{1+e^{-y_0}}\right) = \frac{1}{2y_0}[\frac{1}{2} - g(-y_0)] = \frac{1}{2y_0}[g(y_0) - \frac{1}{2}]$$

$$g(-y) \geq g(y_0)\exp\{-(y+y_0)/2 - \lambda(y_0)(y^2 - y_0^2)\} \forall y_0$$

$$p(z \mid \underline{w}) = e^{yz} g(-y) \geq e^{yz} g(y_0)\exp\{-(y+y_0)/2 - \lambda(y_0)(y^2 - y_0^2)\}$$

$$-\ln p(z \mid \underline{w}) \leq -z\underline{w}^T \underline{x} - \ln g(y_0) + (\underline{w}^T \underline{x} + y_0)/2 + \lambda(y_0)[(\underline{w}^T \underline{x})^2 - y_0^2]$$

Quadratic function in $\underline{w} \Rightarrow$ Gaussian posterior

$$Given\ Data, D = \{\underline{x}^n, z^n\}_{n=1}^N \; and \; prior \; p(\underline{w}) = N(\underline{w}; \underline{w}_0, \Sigma_0)$$

$$-\ln p(\underline{w} \mid D) \leq -\ln p(\underline{w}) - \sum_{n=1}^{N}\left\{\ln(g(y_{0n})) + z^n \underline{w}^T \underline{x}^n - (\underline{w}^T \underline{x}^n + y_{0n})/2 - \lambda(y_{0n})[(\underline{w}^T \underline{x}^n)^2 - y_{0n}^2]\right\}$$

Variational EM for Logistic Regression

- Variational EM for minimizing the upper bound on NLL

Variational E – step:

$$q(\underline{w}) = p(\underline{w} \mid \{y_{0n}\}^{old}) = N(\underline{w}; \underline{w}_N, \Sigma_N)$$

$$(\Sigma_N)^{-1} = (\Sigma_0)^{-1} + 2\sum_{n=1}^{N} \lambda(y_{0n}^{old}) \underline{x}^n \underline{x}^{n^T} = (\Sigma_0)^{-1} + \sum_{n=1}^{N} \frac{1}{y_{0n}} [g(y_{0n}) - \frac{1}{2}] \underline{x}^n \underline{x}^{n^T}$$

$$\underline{w}_N = \Sigma_N \left((\Sigma_0)^{-1} \underline{w}_0 + \sum_{n=1}^{N} (z^n - 1/2) \underline{x}^n \right)$$

Variational M – step: decouples for each y_{0n}

$$Q_i(y_{on}, y_{on}^{old}) = E\{\ln(g(y_{0n})) - y_{0n}/2 - \lambda(y_{0n})[(\underline{w}^T \underline{x}^n)^2 - y_{0n}^2]\}$$

$$\frac{dQ_i(y_{on}, y_{on}^{old})}{dy_{on}} = 0 \Rightarrow \frac{1}{g(y_{0n})} g(y_{0n})[1 - g(y_{0n})] - \frac{1}{2} + 2y_{0n}\lambda(y_{0n}) - \frac{d\lambda(y_{0n})}{dy_{on}} [E(\underline{w}^T \underline{x}^n)^2 - y_{0n}^2]$$

$$= [\frac{1}{2} - g(y_{0n})] + [g(y_{0n}) - \frac{1}{2}] - \frac{d\lambda(y_{0n})}{dy_{on}} [E(\underline{w}^T \underline{x}^n)^2 - y_{0n}^2] = 0$$

$$\Rightarrow E(\underline{w}^T \underline{x}^n)^2 - y_{0n}^2 = 0 \Rightarrow \left(y_{0n}^{new}\right)^2 = \left(\underline{x}^n\right)^T (\Sigma_N + \underline{w}_N \underline{w}_N^T) \underline{x}^n$$

This is still too much work! Are there simpler algorithms? Perceptrons

Laplace (Gaussian) Approximation of Posterior

Represent W by a vector $\underline{w} = \left[\underline{w}_1^T \; \underline{w}_2^T ....., \underline{w}_{C-1}^T \right]^T$, a $(p+1)$x$(C-1)$ vector

Let $\underline{w}^*$ be the optimal solution minimizing the NLL function, J

Let $\nabla^2_{\underline{w}\underline{w}} J = H$ be the Hessian evaluated at $\underline{w}^*$

- Laplace approximation of posterior weight distribution

  $p(\underline{w} \mid D) \sim N(\underline{w}; \underline{w}^*, H^{-1}); H =$ Hessian is the Information matrix

- Posterior predictive distribution of z for a given $\underline{x}$ ($\Rightarrow$testing)

$$P(z \mid \underline{x}, D) = \int_{\underline{w}} P(z \mid \underline{x}, \underline{w}) \, p(\underline{w} \mid D) d\underline{w}$$

(i) Plug-in MAP estimate $\Rightarrow p(\underline{w} \mid D) = \delta(\underline{w} - \underline{w}^*)$: $P(z \mid \underline{x}, D) \approx P(z \mid \underline{x}, \underline{w}^*)$

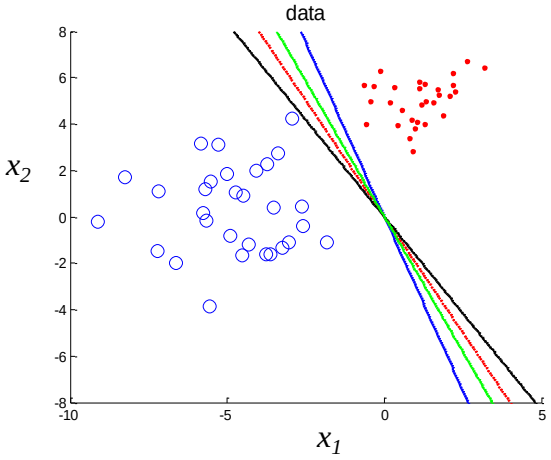
(ii) Monte Carlo approximation: $P(z \mid \underline{x}, D) \approx \frac{1}{M} \sum_{m=1}^{M} P(z \mid \underline{x}, \underline{w}^m)$

$\underline{w}^m$ is drawn from the posterior distribution, $p(\underline{w} \mid D)$

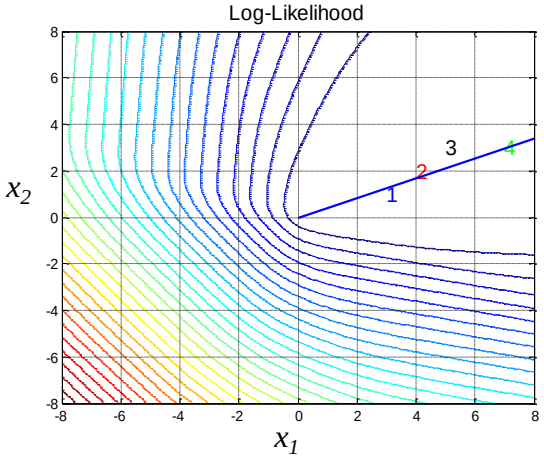
(iii) Probit approximation: $P(z \mid \underline{x}, D) \approx \int_{\underline{w}} P(z \mid \underline{x}, \underline{w}) N(\underline{w}; \underline{w}^*, H^{-1}) d\underline{w}$

Illustration of Logistic Regression in 2D

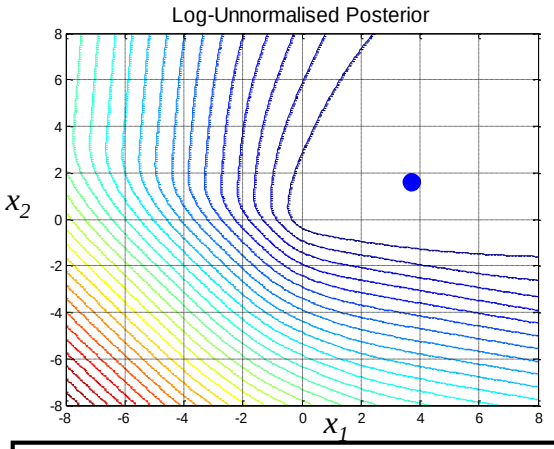
Linearly Separable Data



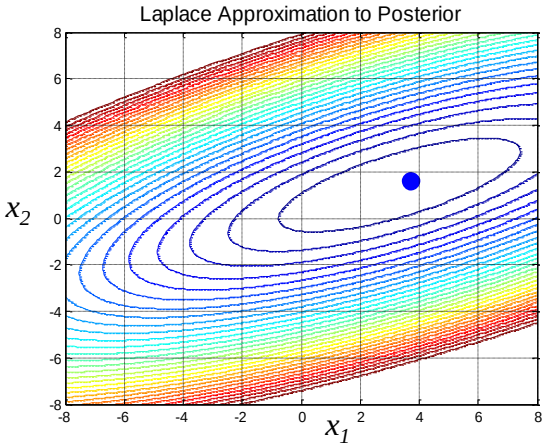
MLE



MAP for Prior $N(\underline{w},\underline{0}, 100I)$



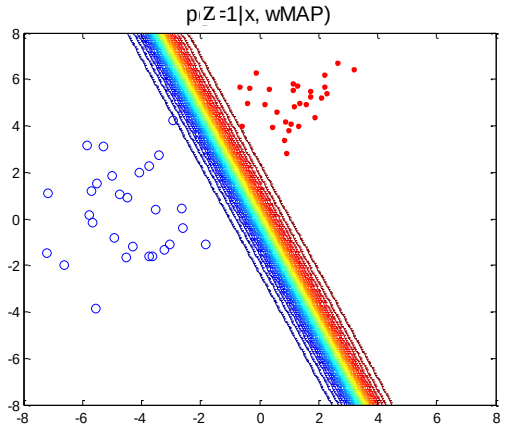
Laplace Approximation of Posterior



Generated using *logredLaplaceGirolamiDemo* from Murphy, Page 257

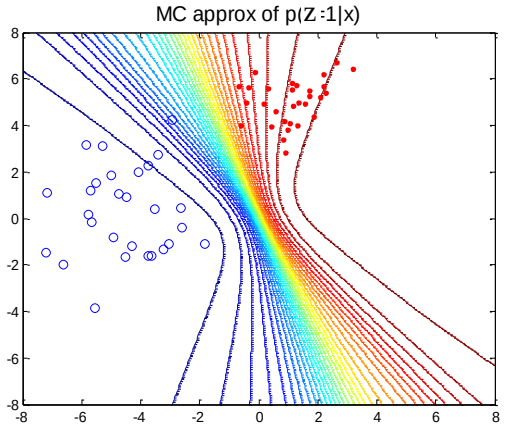
Contours of Posterior Predictive Distribution

Plug-in Approximation

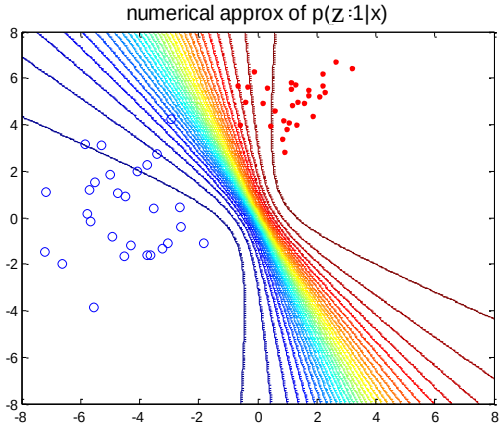


Contours of $P(z=1|\underline{x}, D)$

MC Approximation



Laplace + Probit Approximation of Posterior



- Plug-in underestimates uncertainty
- MC: Posterior mean decision boundary is equally far from both classes (similar to large margin classifiers, e.g., SVM)
- Laplace + probit is similar to MC

Probit Approximation of Sigmoid - 1

- What is probit? **CDF of normal distribution**

$$\Phi(a) \triangleq \int_{-\infty}^{a} N(x;0,1)dx \quad \textit{cumulative distribution function (CDF)}; \Phi(-\infty)=0; \Phi(\infty)=1$$

- Sigmoid function $g(a)$ approximates probit $\Phi(\lambda a)$ well if slopes are matched at the origin. Note $g(-\infty)=0$; $g(\infty)=1$

$$g'(0) = g(0)[1-g(0)] = \frac{1}{4}; \Phi'(0) = \frac{\lambda}{\sqrt{2\pi}} \Rightarrow \lambda = \sqrt{\frac{\pi}{8}}$$

- So,

$$P(z=1|\underline{x},D) \approx \int_{\underline{w}} P(z=1|\underline{x},\underline{w})N(\underline{w};\underline{w}^*,H^{-1})d\underline{w}$$

$$= \int_{\underline{w}} \left(\frac{1}{1+e^{-\underline{w}^T\underline{x}}}\right) N(\underline{w};\underline{w}^*,H^{-1})d\underline{w}$$

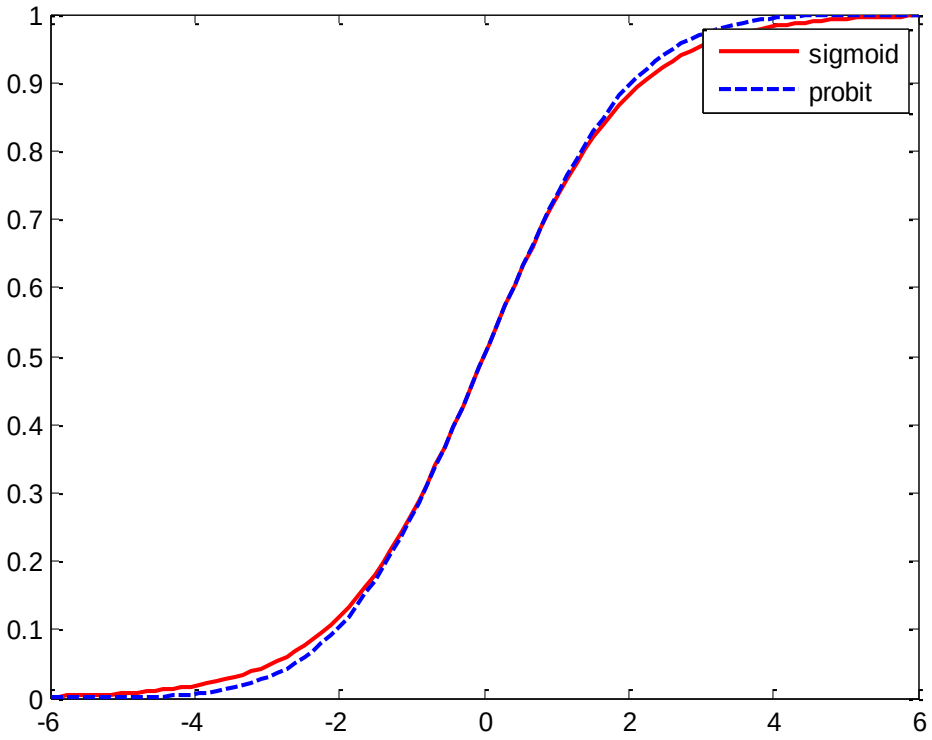
$$\textit{Define } x = z - \lambda a; z \sim N(z;0,1)$$
$$\Rightarrow p(x) = N(x;-\lambda\mu_a, 1+\lambda^2\sigma_a^2)$$
$$p(x|a) = N(x;-\lambda a,1)$$
$$P(x \le 0) = \int_a P(x \le 0|a)N(a;\mu_a,\sigma_a^2)da$$
$$= \int_a \Phi(\lambda a)N(a;\mu_a,\sigma_a^2)da = \Phi\left(\frac{\mu_a}{\left(\sigma_a^2+8/\pi\right)^{1/2}}\right)$$

$$\textit{Let } a = \underline{w}^T\underline{x} = \underline{x}^T\underline{w} \Rightarrow a = N(a; \underbrace{\underline{x}^T\underline{w}^*}_{\mu_a}, \underbrace{\underline{x}^T H^{-1}\underline{x}}_{\sigma_a^2}). \textit{ So, } P(z=1|\underline{x},D) \approx \int_a \underbrace{\left(\frac{1}{1+e^{-a}}\right)}_{g(a)} N(a;\mu_a,\sigma_a^2)da$$

$$\approx \int_a \Phi(\sqrt{\pi/8}\,a)N(a;\mu_a,\sigma_a^2)da = \Phi\left(\frac{\mu_a}{\left(\sigma_a^2+8/\pi\right)^{1/2}}\right) \approx g\left(\frac{\mu_a}{\left(1+\pi\sigma_a^2/8\right)^{1/2}}\right)$$

$$\int_{-\infty}^{\infty} \Phi(a+bx)\phi(x)\,dx = \Phi\left(\frac{a}{\sqrt{1+b^2}}\right)$$

Probit Approximation of Sigmoid - 2

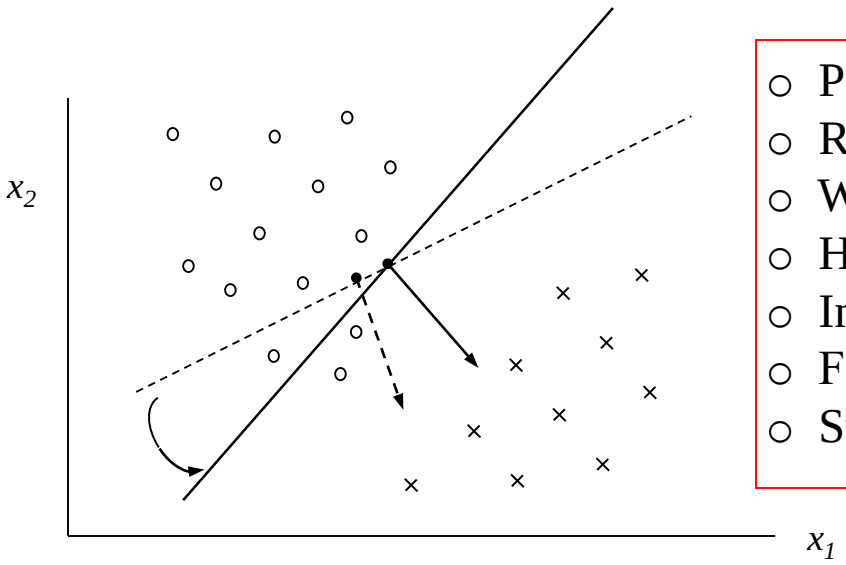


Non-parametric Learning of Discriminant Weights

Consider linear discriminants

$$y(\underline{x}, \underline{w}_k) = \underline{w}_k^{\,T}\,\underline{x} - w_{k0} \;(or)\; y(\underline{x}, \underline{w}_k) = \sum_{j=1}^{M} w_{kj}\varphi_{kj}(\underline{x}) - w_{k0}$$

Idea: Iteratively change $\underline{w}$



- Perceptron (linear) ↔ LVQ (non linear)
- Relaxation Procedures
- Widrow – Hoff (LMS, Adaline)
- Ho-Kashyap Procedure
- Incremental Gauss – Newton = RLS
- Fisher's linear discriminant
- Support Vector Machines

Two Class Case

Let us consider two class case first

Select class 1 if $\underline{w}_1^T \underline{x} > \underline{w}_2^T \underline{x}; \underline{x} = (-1\ x_1\ x_2 \cdots\cdots x_p)$

$$\Rightarrow (\underline{w}_1 - \underline{w}_2)^T \underline{x} > 0 \Rightarrow \underline{w}^T \underline{x} > 0$$

For simplicity write $y(\underline{x}, \underline{w}) = \underline{w}^T \underline{x}$

$$\underline{w}^T = (w_0\ w_1 \cdots\cdots w_p)$$
$$\underline{x} = (-1\ x_1\ x_2 \cdots\cdots x_p)$$

The output $g = \begin{cases} 1 \text{ if } y > 0 \\ 0 \text{ if } y < 0 \end{cases} \Rightarrow \quad g = I(y > 0) = I(\underline{w}^T \underline{x} > 0)$

- **Problem:** Want to find a $\underline{w}$ э

$\underline{w}^T \underline{x} > 0$ if $\underline{x}$ belongs to class 1 $\Rightarrow \theta < 90^\circ$ (acute angle)

$\underline{w}^T \underline{x} < 0$ if $\underline{x}$ belongs to class 2 $\Rightarrow \theta > 90^\circ$ (obtuse angle)

since $\underline{w}^T \underline{x} = \|w\| \|\underline{x}\| \cos\theta$

Linear Programming (LP)Formulation-1

- **Alternate problem**

  For class 2, let $\underline{x}^i = -\underline{x}^i \quad i = n_1 + 1, \cdots\cdots, N \quad \Rightarrow \underline{w}^T \underline{x}^i > 0 \;\; \forall \; i$

  In fact, one can find $\underline{w} \ni \underline{w}^T \underline{x}^i > b_i \quad \forall \; i$

  We have error if $\underline{w}^T \underline{x}^i < b_i$ or $b_i - \underline{w}^T \underline{x}^i > 0$

- **Why not minimize a measure of errors?**

$$\min_{\underline{w}} \sum_{i=1}^{N} \max(0, b_i - \underline{w}^T \underline{x}^i)$$

$$\Rightarrow \min_{\underline{w},\underline{u}} \sum_{i=1}^{N} u_i$$

$$u_i \geq 0$$

$$u_i - b_i + \underline{w}^T \underline{x}^i \geq 0 \quad i = 1, 2, \cdots\cdots, N$$

} LP

Since $\underline{w}$ is unrestricted in sign, define $\underline{w} = \underline{w}^+ - \underline{w}^-, \quad \underline{w}^+ \geq \underline{0}, \; \underline{w}^- \geq \underline{0}$

LP Formulation-2

Let

$$\underline{u}_a = \begin{bmatrix} \underline{u} \\ \underline{w}^+ \\ \underline{w}^- \end{bmatrix} \qquad A = \begin{matrix} \\ \uparrow \\ N \\ \downarrow \end{matrix} \overset{\begin{matrix} N & p+1 & p+1 \end{matrix}}{\begin{bmatrix} & \vdots & \left(\underline{x}^1\right)^T & \left(-\underline{x}^1\right)^T \\ I_N & \vdots & \left(\underline{x}^2\right)^T & \left(-\underline{x}^2\right)^T \\ & \vdots & \vdots & \vdots \\ & \vdots & \left(\underline{x}^N\right)^T & \left(-\underline{x}^N\right)^T \end{bmatrix}}; \quad \underline{c}^T = [\overbrace{1 1 \cdots 1}^{\underline{e}^T}\ \overbrace{0 0 \cdots}^{N}\ \overbrace{0 \cdots 0}^{p+1}\ \overbrace{}^{p+1}]$$

$$\underline{b} = [b_1, b_2, \ldots, b_N]^T$$

Then the LP becomes:

$$\left.\begin{array}{ll} & \min \underline{c}^T \underline{u}_a \\ s.t. & A\underline{u}_a \geq \underline{b} \\ & \underline{u}_a \geq \underline{0} \end{array}\right\} \quad \text{if } \sum_{i=1}^{N} u_i = 0 \Rightarrow \text{ Linearly separable}$$

Will get an answer even if it is not

Initial Basic Feasible Solution: $u_i = b_i, \underline{w}^+ = \underline{w}^- = \underline{0}$

Dual Problem

- **Dual of the LP problem**

$$\max \underline{\lambda}^T \underline{b} \qquad \sum_{i=1}^{N} \lambda_i b_i$$

$$\text{s.t. } A^T \underline{\lambda} \le \underline{c} \quad \Rightarrow \quad 0 \le \lambda_i \le 1$$

$$\underline{\lambda} \ge \underline{0} \qquad \sum_{i=1}^{N} \underline{x}^i \lambda_i = \underline{0} \Rightarrow \underline{\lambda} \in \mathbf{N}(\underline{x}^1 \ \underline{x}^2 \cdots\cdots \underline{x}^N)$$

$$A^T \underline{\lambda} = \begin{bmatrix} & I_N & & \\ \cdots & \cdots & \cdots & \cdots \\ \underline{x}^1 & \underline{x}^2 & \cdots & \underline{x}^N \\ -\underline{x}^1 & -\underline{x}^2 & \cdots & -\underline{x}^N \end{bmatrix} \underline{\lambda} = \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_N \\ \sum \underline{x}^i \lambda_i \\ -\sum \underline{x}^i \lambda_i \end{bmatrix} \le \begin{bmatrix} \underline{e} \\ \cdots \\ 0 \\ \cdots \\ 0 \end{bmatrix} \Rightarrow \left.\begin{array}{l} \sum \underline{x}^i \lambda_i \le \underline{0} \\ -\sum \underline{x}^i \lambda_i \le \underline{0} \end{array}\right\} \Rightarrow \sum_i \underline{x}^i \lambda_i = \underline{0}$$

Linear Separability $\Rightarrow \boxed{\underline{\lambda} = \underline{0}}$

This is also too much work! In 1950's, researchers were interested in finding iterative solutions.

Decision-Based Learning – 1

- Rosenblatt's Perceptron learning rule ~ Decision-based learning

  Note: Class 2 is scaled so that $\underline{x}^{i} = -\underline{x}^{i}$

  Idea: If $\underline{w}^{(n)^T}\,\underline{x}^{n} > 0$ at iteration n then do nothing

  else $\underline{w}^{(n+1)} = \underline{w}^{(n)} + \eta \underline{x}^{n} \qquad \eta > 0$

  Does it make sense?

  $$\underline{w}^{(n+1)T}\,\underline{x}^{n} = \underline{w}^{(n)T}\,\underline{x}^{n} + \eta \underline{x}^{nT}\,\underline{x}^{n} > \underline{w}^{(n)T}\,\underline{x}^{n} \quad \Rightarrow \text{ make it less negative}$$

- **Perceptron Algorithm ~ Supervisory learning (*Reinforcement learning*)**

  $$\underline{w} = \underline{0}$$

  Do until no misclassification (or misclassification error becomes constant)

  Do $n = 1,2,\cdots\cdots,N$

  If $\underline{w}^{T}\,\underline{x}^{n} < 0$

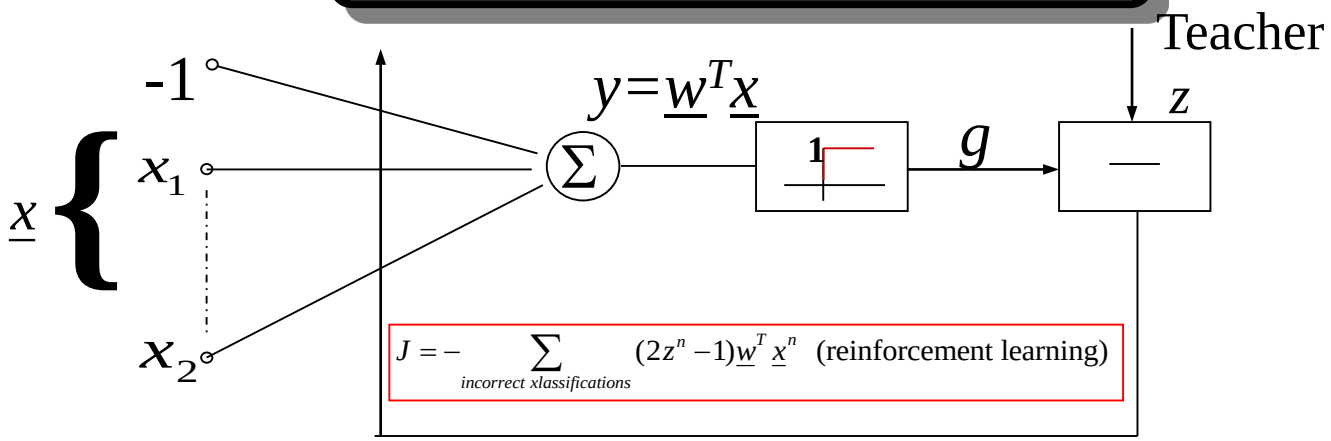
  $\underline{w} \leftarrow \underline{w} + \eta \underline{x}^{n}$

  End If

  End Do

  End Do

Decision Based Learning - 2



- ***Note:*** When you do not scale class 2 features, $\underline{w} \leftarrow \underline{w} + \eta(z^n - g^n)\underline{x}^n$

$$z^n = \begin{cases} 1 & \text{if class } 1 \\ 0 & \text{otherwise} \end{cases}$$

$\Rightarrow \underline{x}^n$ belongs to class 1 ($z^n = 1$) but assigned to class 2 ($g^n = 0$), then

$$\underline{w} \leftarrow \underline{w} + \eta \underline{x}^n \quad \text{(reinforcement learning)}$$

$\underline{x}^n$ belongs to class 2 ($z^n = 0$) but assigned to class 1 ($g^n = 1$), then

$$\underline{w} \leftarrow \underline{w} - \eta \underline{x}^n \quad \text{(antireinforcement learning)}$$

- You see the reason why we scaled $\underline{x}^n = -\underline{x}^n$ for class 2

Linear Perceptron Convergence Theorem-1

❑ Linear Perceptron Convergence Theorem

If classes are linearly separable, i.e., if there exists a vector $\underline{w}^k$ and a number $b > 0 \quad \ni \underline{w}^T \underline{x}^n \geq b \quad \forall\, n$ (when class 2 features are scaled), then the perceptron learning rule converges in finite number of iterations.

Proof 1:

$$\cos(\underline{u}, \underline{v}) = \frac{\underline{u}^T \underline{v}}{\|\underline{u}\|_2 \, \|\underline{v}\|_2} \leq 1$$

$$\underline{w}^{*T} \underline{w}^{(n+1)} = \underline{w}^{*T} (\underline{w}^{(n)} + \eta \underline{x}^n)$$

$$\underline{w}^{*T} \underline{w}^{(n+1)} \geq \underline{w}^{*T} \underline{w}^{(n)} + \eta b \qquad \text{Assuming } \underline{w}^{(0)} = \underline{0}.$$

$$\Rightarrow \underline{w}^{*T} \underline{w}^{(n+1)} \geq \eta n b \qquad \cdots\cdots\cdots\cdots \qquad (1)$$

Linear Perceptron Convergence Theorem-2

Proof 1 continued:

$$\left\|\underline{w}^{(n+1)}\right\|^2 = \left\|\underline{w}^{(n)} + \eta \underline{x}^n\right\|^2 = \left\|\underline{w}^{(n)}\right\|^2 + 2\eta \underline{w}^{(n)T} \underline{x}^n + \eta^2 \left\|\underline{x}^n\right\|^2 < \left\|\underline{w}^{(n)}\right\|^2 + \eta^2 \left\|\underline{x}^n\right\|^2$$

$$\Rightarrow \left\|\underline{w}^{(n+1)}\right\|^2 < n\eta^2\beta^2 \text{ where } \beta^2 = \max_i \| \underline{x}^i \|^2$$

$$\Rightarrow \cos(\underline{w}^{(n+1)}, \underline{w}^*) = \frac{\underline{w}^{*T} \underline{w}^{(n+1)}}{\left\|\underline{w}^*\right\|_2 \left\|\underline{w}^{(n+1)}\right\|_2} \geq \frac{\eta n b}{\sqrt{n\eta^2\beta^2}\left\|\underline{w}^*\right\|_2} \leq 1$$

$$\sqrt{n} b \leq \left\|w^*\right\|_2 \beta$$

$$n \leq \frac{\left\|w^*\right\|^2 \beta^2}{b^2}$$

Linear Perceptron Convergence Theorem-3

Proof 2:

$$\left\|\underline{w}^{(n+1)} - \underline{w}^{*}\right\|^2 = \left\|\underline{w}^{(n)} - \underline{w}^{*} + \eta \underline{x}^{n}\right\|^2 \qquad \text{know } \underline{w}^{(n)T}\,\underline{x}^{n} < 0$$

$$= \left\|\underline{w}^{(n)} - \underline{w}^{*}\right\|^2 + 2\eta(\underline{w}^{(n)} - \underline{w}^{*})^T \underline{x}^{n} + \eta^2 \left\|\underline{x}^{n}\right\|^2$$

$$< \left\|\underline{w}^{(n)} - \underline{w}^{*}\right\|^2 - 2\eta \mathrm{b} + \eta^2 \beta^2$$

$$< \left\|\underline{w}^{(n)} - \underline{w}^{*}\right\|^2 \text{ if } \eta < \frac{2b}{\beta^2}$$

- *For η sufficiently small, linear term dominates the quadratic term.*
- *Every monotonic sequence has a limit. So, the sequence converges.*

Optimal Learning Rate

❑ Optimal Learning Rate η_{opt}

$$\left\|\underline{w}^{(n+1)} - \underline{w}^{*}\right\|^2 = \left\|\underline{w}^{(n)} - \underline{w}^{*}\right\|^2 + \eta^2 \left\|\underline{x}^{n}\right\|^2 + 2\eta(\underline{w}^{(n)} - \underline{w}^{*})^T \underline{x}^{n}$$

$$= \left\|\underline{w}^{(n)} - \underline{w}^{*}\right\|^2 + \eta^2 \left\|\underline{x}^{n}\right\|^2 - 2\eta\left[\left|\underline{w}^{*T} \underline{x}^{n}\right| + \left|\underline{w}^{(n)T} \underline{x}^{n}\right|\right]$$

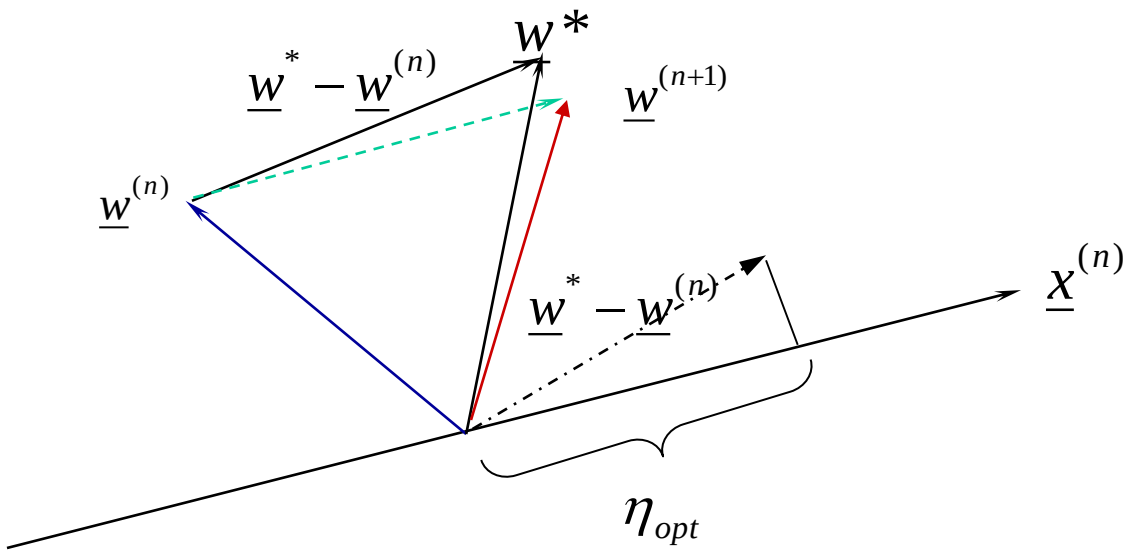
valid when don't scale also

$$\frac{\partial}{\partial \eta} = 0 \Rightarrow \quad \eta_{opt} = \frac{\left|\underline{w}^{*T} \underline{x}^{n}\right| + \left|\underline{w}^{(n)T} \underline{x}^{n}\right|}{\left\|\underline{x}^{n}\right\|^2}$$

$$= \frac{(\underline{w}^{*} - \underline{w}^{(n)})^T \underline{x}^{n}}{\left\|\underline{x}^{n}\right\|^2}$$

The optimal step size is chosen to move $\underline{w}^{(n+1)}$ as close to $\underline{w}^{*}$ as possible.

Relaxation Method



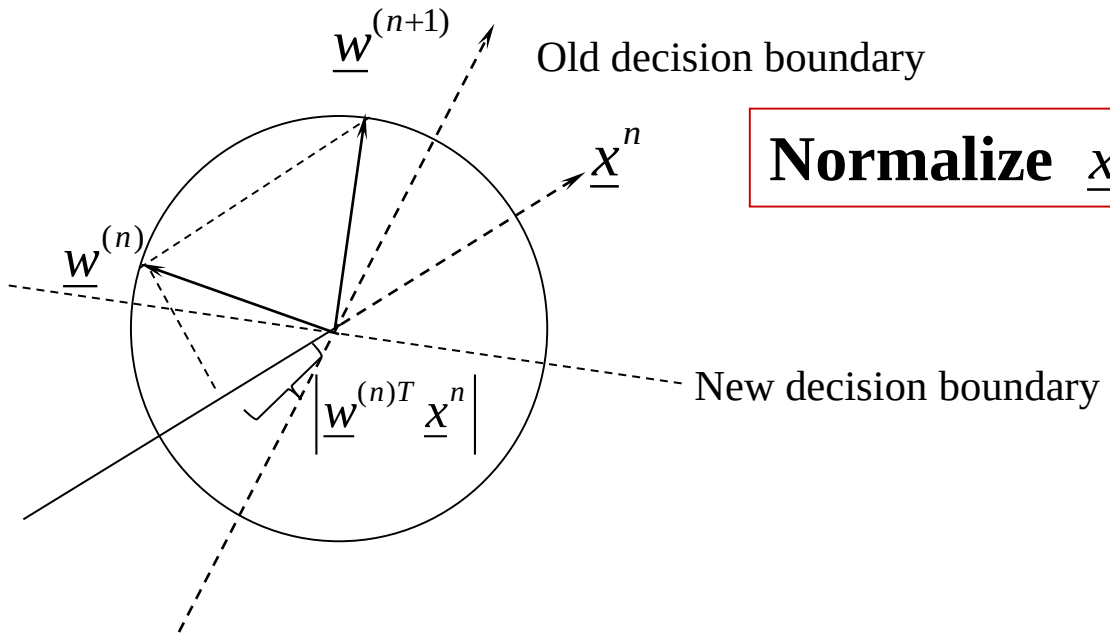
But do not know w^{*}. In *Relaxation method*, $w^{*T} x^{n}$ is replaced by a small positive number b .

$$\eta_n = \frac{b + \left| w^{(n)T} x^{n} \right|}{\left\| x^{n} \right\|^2}$$

Normalized Perceptron - 1

$\underline{w}^{(1)}$ is normalized (best method)

$$\eta_N = \frac{2\left|\underline{w}^{(n)T}\,\underline{x}^n\right|}{\left\|\underline{x}^n\right\|^2}$$



Normalized Perceptron - 2

Leads to finite convergence because

$$\left\|\underline{w}^{(n+1)} - \underline{w}^{*}\right\|^2 = \left\|\underline{w}^{(n)} - \underline{w}^{*}\right\|^2 - 4\left|\underline{w}^{(n)T}\,\underline{x}^{n}\right|^2 - 4\left|\underline{w}^{(n)T}\,\underline{x}^{n}\right|\left|\underline{w}^{*T}\,\underline{x}^{n}\right| + 4\left|\underline{w}^{(n)T}\,\underline{x}^{n}\right|^2$$

$$= \left\|\underline{w}^{(n)} - \underline{w}^{*}\right\|^2 - 4\left|\underline{w}^{(n)T}\,\underline{x}^{n}\right|\left|\underline{w}^{*T}\,\underline{x}^{n}\right|$$

$$< \left\|\underline{w}^{(n)} - \underline{w}^{*}\right\|^2$$

Also note that

$$\left\|\underline{w}^{(n+1)}\right\|^2 = \left\|\underline{w}^{(n)} + \eta_N\,\underline{x}^{n}\right\|^2 = \left\|\underline{w}^{(n)}\right\|^2 + 2\eta_N \underbrace{\underline{w}^{(n)T}\,\underline{x}^{n}}_{<0} + \eta_N{}^2\left\|\underline{x}^{n}\right\|^2 = \left\|\underline{w}^{(n)}\right\|^2$$

Extension to Multiple Classes

Extension to Multiple Classes $\{1, 2, \cdots\cdots\cdots, C\}$

$$k = \arg\max_i (\underline{w}_i^T \underline{x})$$

- Learning rule

  Suppose at present a pattern $\underline{x}^n$ belongs to class j

  If $k = j$ do nothing

  Else

  $$\underline{w}_j^{(n+1)} = \underline{w}_j^{(n)} + \eta \underline{x}^n \qquad \text{so that } \underline{w}_j^{(n+1)T} \underline{x}^n \uparrow$$

  $$\underline{w}_k^{(n+1)} = \underline{w}_k^{(n)} - \eta \underline{x}^n \qquad \text{so that } \underline{w}_k^{(n+1)T} \underline{x}^n \downarrow$$

  End If

Note: Leave other classes alone.

Note: Sometimes check $\underline{w}_k^T \underline{x}^n > \underline{w}_i^T \underline{x}^n + \varepsilon \quad \forall\ i \neq k$ for patterns belonging to class k, where ε is called a vigilance parameter.

Finite Convergence

$\underline{x}^n$ belongs to class j

$$\underline{w}_j^{*T}\,\underline{x}^n > \underline{w}_k^{*T}\,\underline{x}^n \;\; \forall\; k \neq j$$

This set of inequalities can be expressed as

$$\underline{q}^T = [\underline{w}_1^{*T}\; \underline{w}_2^{*T} \cdots\cdots \underline{w}_C^{*T}]$$

$C\,(p+1)$ dimensional vector

$$\underline{p}_{jk}^{\,T} = [0^T \cdots \overset{j}{\left(\underline{x}^n\right)^T} \cdots - \overset{k}{\left(\underline{x}^n\right)^T} \cdots 0^T \;]$$

$(p+1)$ dimensional vector

$\Rightarrow$ need $\underline{q}^T\,\underline{p}_{jk} > 0 \;\; \forall \;\; j \neq k$

If $\underline{q}^T\,\underline{p}_{jk} < 0$ for any $j \neq k$ $\Rightarrow$ teacher says it's wrong

$$\left.\begin{aligned} \underline{w}_j^{(n+1)} &= \underline{w}_j^{(n)} + \eta \underline{x}^n \\ \underline{w}_k^{(n+1)} &= \underline{w}_k^{(n)} - \eta \underline{x}^n \end{aligned}\right\} \Rightarrow \underline{q}^{new} = \underline{q}^{old} + \eta(z-g)\,\underline{p}_{jk}$$

Proof similar to binary case. See S. Y. Kung's Book "Digital Neural Networks" for proof of convergence

- Only weights j and k are updated

Key Properties of Perceptron

- **Key properties of Perceptron**:
  - It employs distributed decision-based credit assignment
  - Update weights only when misclassification occurs
  - **Distributed and localized**: reinforce correct class, penalize wrong decision class
  - Update depends on $y(\underline{x}, \underline{w})$

$$\Delta \underline{w} = \pm \eta \nabla_{\underline{w}} y(\underline{x}, \underline{w})$$

when $y(\underline{x}, \underline{w}) = \underline{w}^T \underline{x} \quad \Rightarrow \quad \Delta \underline{w} = \pm \eta \underline{x}$

$\Rightarrow$ Perceptron learning rule (Incremental gradient, Stochastic gradient)

Fuzzy Updates - 1

- Fuzzy updates
  - Suppose $\underline{x}$ belongs to class j

    In fuzzy update, one finds the **challenger to correct class** j

$$k = \arg\max_{i \neq j} y(\underline{x}, \underline{w}_i)$$

❖ Find a measure of misclassification

$$d = -y(\underline{x}, \underline{w}_j) + y(\underline{x}, \underline{w}_k)$$

$$r = \infty \Rightarrow$$
$$d = -y(\underline{x}, \underline{w}_j) + y(\underline{x}, \underline{w}_k)$$

more generally

$$d = -y(\underline{x}, \underline{w}_j) + \left[\frac{1}{C-1} \sum_{i \neq j} [y(\underline{x}, \underline{w}_i)]^r \right]^{1/r}, \; y > 0$$

$d > 0 \quad \Rightarrow$ misclassification

$d < 0 \quad \Rightarrow$ correct classification

Fuzzy Updates - 2

- Define a penalty function
  - Logistic

$$g(d) = \frac{1}{1+e^{-d/\xi}}$$

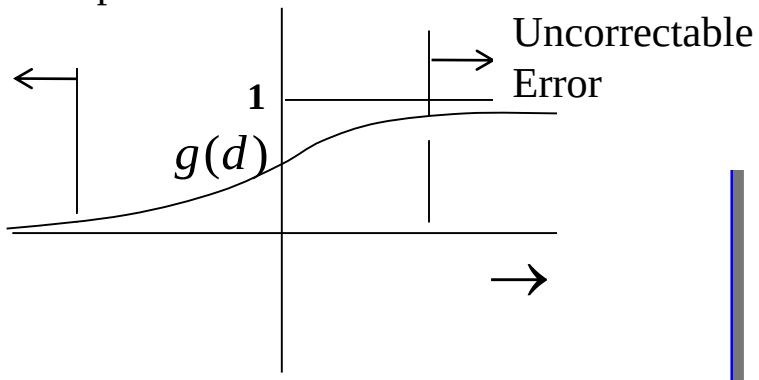
$\xi \downarrow 0 \Rightarrow$ step

Correct, but improve

Incorrect, but can be corrected

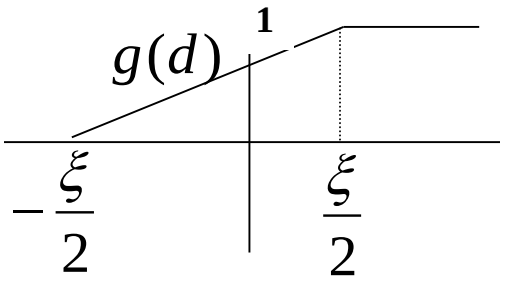
Correct, No action needed

Uncorrectable Error



  - Linear

$$g(d) = \begin{cases} 0 & d \le -\xi/2 \\ (d+\xi/2)/\xi & -\xi/2 < d < \xi/2 \\ 1 & d \ge \xi/2 \end{cases}$$



$$\underline{w}_j^{(n+1)} = \underline{w}_j^{(n)} + \eta g'(d) \nabla y(\underline{x}^n, \underline{w}_j)$$

$$\underline{w}_k^{(n+1)} = \underline{w}_k^{(n)} - \eta g'(d) \nabla y(\underline{x}^n, \underline{w}_k)$$

Fuzzy Updates - 3

✓ *Logistic:*

$$g'(d) = \frac{1}{\xi} g(1-g)$$

✓ *Linear:*

$$g'(d) = \begin{cases} 0 & d \le -\xi/2 \\ 1 & -\xi/2 < d < \xi/2 \\ 0 & \mathrm{d} \ge \xi/2 \end{cases}$$

Start with large ξ *and progressively reduce it.*

LVQ as Perceptron Update

- Another prominent example is the LVQ algorithm of Kohonen for **supervised** clustering

$$y(\underline{x}, \underline{w}_i) = -\frac{\|\underline{x} - \underline{w}_i\|^2}{2} \qquad i = 1, 2, \cdots\cdots, C$$

Assign pattern $\underline{x}$ to class k if

$$k = \arg\max_i -\frac{\|\underline{x} - \underline{w}_i\|^2}{2} = \arg\min_i \frac{\|\underline{x} - \underline{w}_i\|^2}{2}$$

$$\nabla_{w_i} y(\underline{x}, \underline{w}_i) = -(\underline{x} - \underline{w}_i) \quad \text{for the min function}$$

Suppose $\underline{x}^n$ belongs to class j, but the decision is class k

$$\underline{w}_j^{(n+1)} = \underline{w}_j^{(n)} + \eta(\underline{x}^n - \underline{w}_j^{(n)}) \qquad \text{(reinforcement learning)}$$

$$\underline{w}_k^{(n+1)} = \underline{w}_k^{(n)} - \eta(\underline{x}^n - \underline{w}_k^{(n)}) \qquad \text{(antireinforcement learning)}$$

Capacity Questions

- How many random patterns a Perceptron with p inputs can learn reliably in a 2 class case? $\approx 2p$

Suppose have N patterns and randomly assign them to one of two classes. They are linearly separable with probability $P(N, p)$

$$P(N,p) = \begin{cases} 1 & N \le p+1 \\ \frac{2}{2^N}\sum_{i=0}^{p}\binom{N-1}{i} & N > p+1 \\ \approx \Phi\left(\frac{2p-N}{\sqrt{N}}\right) & \text{for large } N \end{cases}$$

$N = 4; p = 2$

$$\Rightarrow P(N,p) = \frac{1}{8}(1+3+3)$$

$$= \frac{7}{8}$$

XOR Pattern cannot be correctly classified by a Perceptron

Decision Trees and Instance-based Classifiers

- Decision Trees and Instance-based Classifiers (e.g., K nearest neighbor classifier)
- Common learning paradigms in AI and statistics
- Idea: Exploit regularities or features in observations and use them to predict on the basis of previously encountered situations
  - Decision trees …… Make shared properties explicit
  - Instance-based …… Find distance from the new and stored cases $\Rightarrow$ Nearest neighbor $\{I^1, I^2, \ldots\ldots, I^N\}$
  - o Set of instances, examples, cases
  - o Each instance I^k is represented by $(x_1, x_2, \ldots\ldots, x_p)_k = \underline{x}^k$ attributes or features (continuous /discrete) and class z^k

Case-based reasoning

Illustrative Example

Case #	Outlook	Temp (°F)	Humidity (%)	Windy	Class (Take umbrella or Not)
1.	Rain	70	96	False	Yes
2.	Sunny	80	90	True	No
3.	Overcast	64	65	True	Yes
4.	Sunny	75	70	True	Yes
5.	Sunny	85	85	False	No
6.	Sunny	72	95	False	No
7.	Rain	75	80	False	Yes
8.	Sunny	69	70	False	Yes
9.	Overcast	83	78	False	Yes
10.	Rain	65	70	True	No
11.	Overcast	72	90	True	Yes
12.	Overcast	81	75	False	Yes
13.	Rain	68	80	False	Yes
14.	Rain	71	80	True	No

Partitioning Data based on Attribute Outcomes

$$\underline{x}^k = \left(x_1\ x_2\ x_3\ x_4\right)^k$$

Outlook (Discrete) | Temp | Humidity | Windy

$$x_1 \in \{\text{Sunny, Overcast, Rain}\}$$
$$x_2 \in R$$
$$x_3 \in R$$
$$x_4 \in \{\text{true, false}\}$$

$$\text{class} \in \{\text{yes, no}\} \quad \sim \quad \text{binary}\ \ Z = \{z_1\ z_2\}$$

Take umbrella or not

Decision tree divides the description (feature) space into regions, each associated with one of the classes.

Decision Regions in a Decision Tree

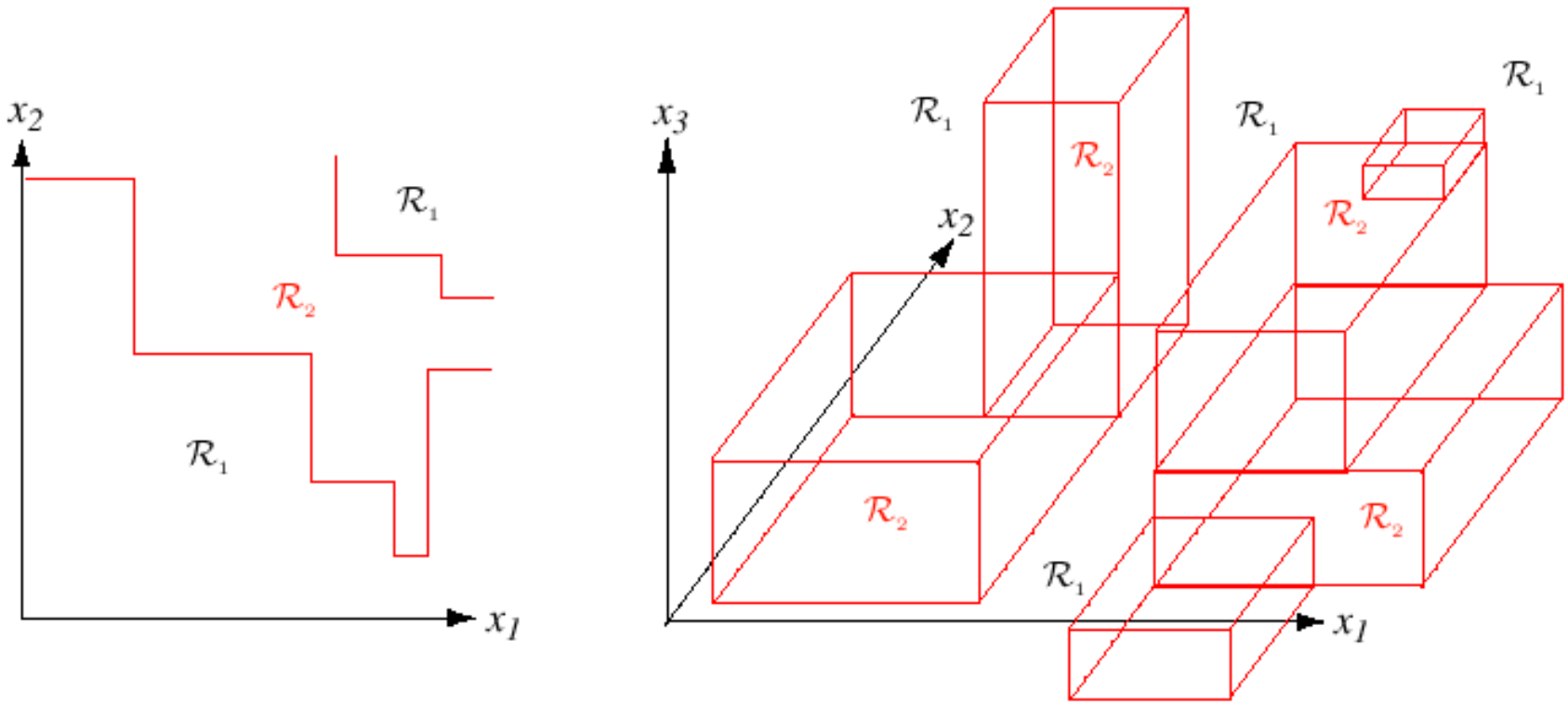


FIGURE Monothetic decision trees create decision boundaries with portions perpendicular to the feature axes. The decision regions are marked $\mathcal{R}_1$ and $\mathcal{R}_2$ in these two-dimensional and three-dimensional two-category examples. With a sufficiently large tree, any decision boundary can be approximated arbitrarily well in this way.

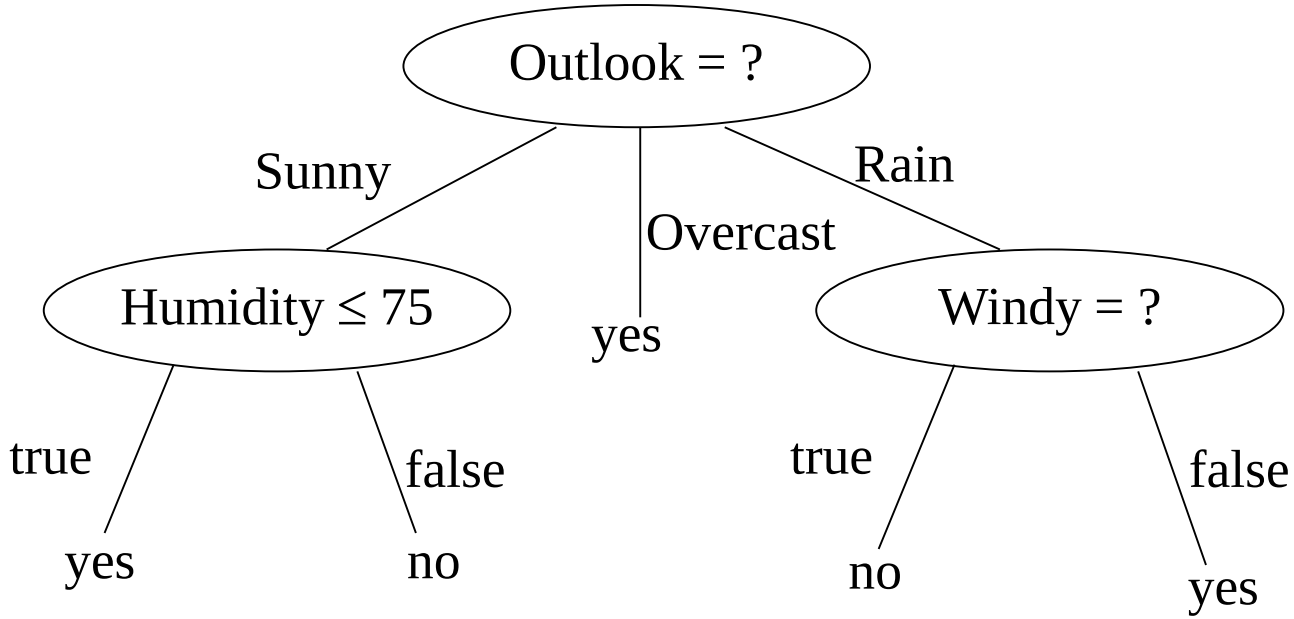
History of Decision Trees

- Decision Trees:
  - Developed since 1960s and popularized by Brieman and Quinlan

    CART C 4.5 Gradient Boosting Random Forests

    (Classification and Regression trees)
  - Have been used in industrial applications, particularly in diagnosis and control, quality control
  - Invariant to scaling; can handle large datasets; easily ignore redundant variables; handle missing variables easily through surrogate splits; Easy to interpret
  - Nearly half of the data mining competitions are won by using some variants of tree ensemble methods: Boosting, Random Forests, Bagging
  - Used in statistics, pattern recognition, decision theory, signal processing, machine learning and artificial neural networks
  - Three uses: Data description (compression, rules), classification, and generalization
- Survey: Srirama K. Murthy: "*Automatic Construction of Decision Trees From Data: A Multi-disciplinary Survey*," pp 1-49. Kulwer Academic Publishers. Data Mining and Knowledge Disocery 2 (4): 345-389 (1998)
- Also, look at my papers on sequential fault diagnosis since 1990. Primarily in *IEEE T-SMC*.

Decision Tree for the Weather Problem



- Case: Outlook=sunny, Temp=82, Humidity =85, Windy = True $\Rightarrow$ no
- This is a case of exact partitioning where we grow the tree until all the cases are classified correctly.
- In noisy situations, distributions of classes overlap. Here, we would like to stop growing the tree or prune it after constructing.

Decision Tree Example: Fruit Classification

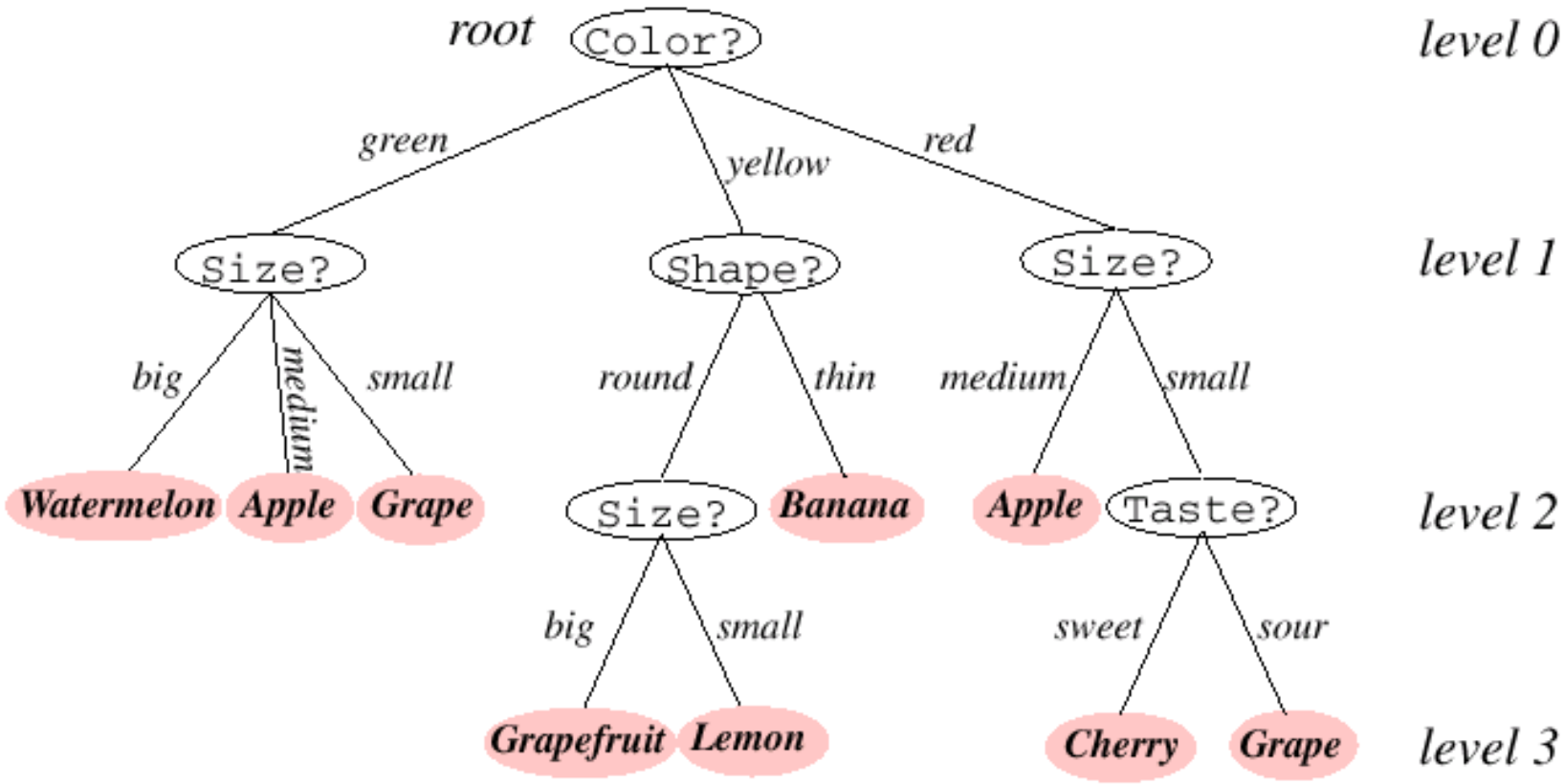


FIGURE Classification in a basic decision tree proceeds from top to bottom.

Defining "Attribute Tests" - 1

- Method for constructing trees ---- want to generate compact trees
  - All employ recursive divide-and-conquer algorithm
  - How to select tests? ……. Splitting rules
- Most tree construction techniques use single attribute (feature) tests
  - Discrete attribute $x_i : x_i \in \{\upsilon_1\ \upsilon_2 \cdots\cdots \upsilon_m\}$

    $x_i = ?$ with m outcomes

    $x_i = \upsilon_j$, m binary tests with outcomes true or false
  - Continuous attribute x_i

    $x_i \le t$ (with outcomes true or false) for some constant t

    Suppose have observations, $n_1 < n_2 < \cdots\cdots < n_N$

    Need to select thresholds $t \in [n_i, n_{i+1})$; *e.g.*, $t = (n_i + n_{i+1})/2$

Defining "Attribute Tests" - 2

- Select t ə mutual information $IG(z|x,t)$ is maximum

$$IG(z \mid x,t) = H(z) - H(z \mid x,t)$$

$$H(z) = -\sum_{j=1}^{C} P(z = j) \log_2 P(z = j)$$

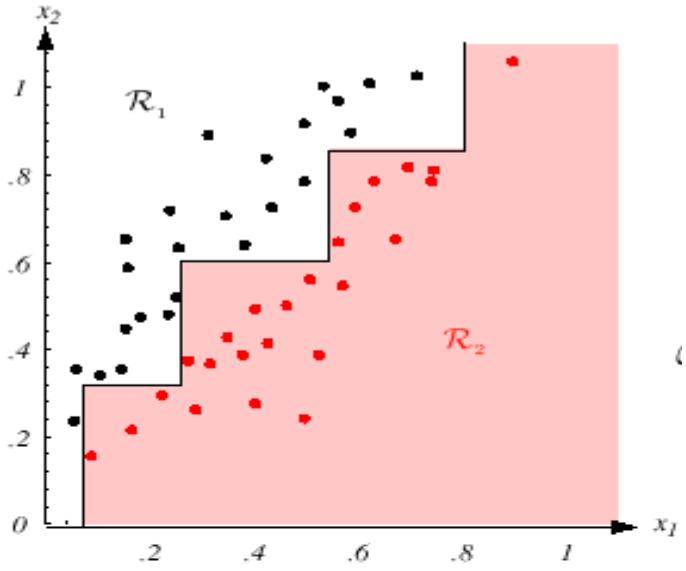
$$H(z \mid x,t) = P(x < t) H(z \mid x < t) + P(x \geq t) H(z \mid x \geq t)$$

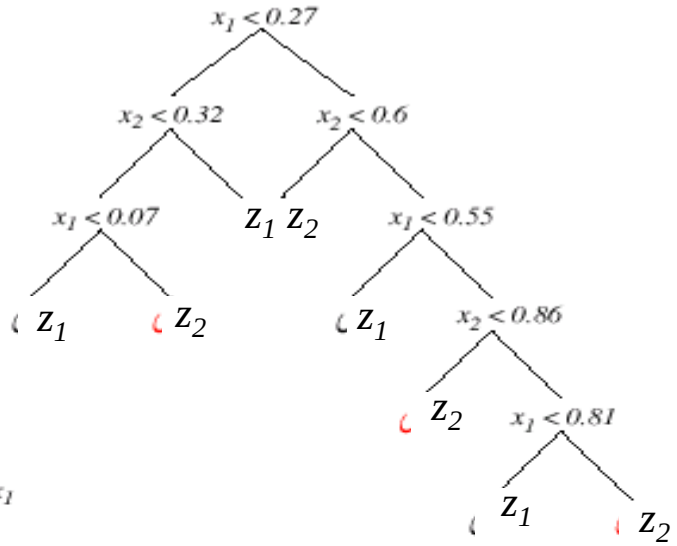
- Complex tests consider subsets or linear combination

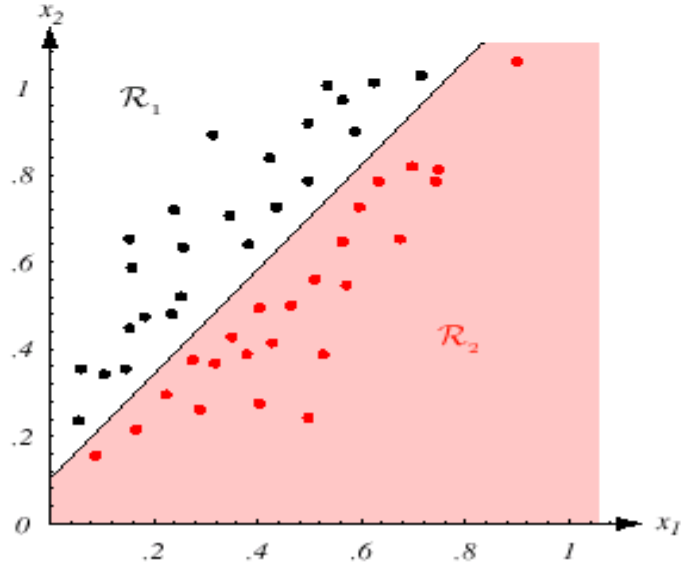
$$\underline{w}^T \underline{x} - w_0 \geq 0 \quad \text{true or false}$$

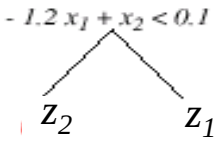
- Each test could be a classifier (see Error Correcting Codes later)

Method for Constructing Trees









If the class of node decisions do not match the form of the training data, a very complicated decision tree will result, as shown at the top. However, if proper decision forms ("Tests") are used, the tree can be quite simple.

Criteria for Splitting Tests - 1

- Idea
  - Suppose apply a test T that partitions D **into** $D_1\ D_2, \cdots\cdots, D_m$
  - Want to select a test T such that the child nodes $D_1 \cdots\cdots D_m$ are "*purer*" than their parent D.
  - Measure of impurity is such that it is zero if P_j is concentrated on one class and is maximal if $P_j = 1/C$ is uniform. Recall the definition of entropy.
  - So, stop building the tree if all cases belong to a single class

- Entropy: A measure of impurity (uncertainty)

$$H(D) = -\sum_{j=1}^{C} P_j \log_2 P_j \qquad H(D) = \begin{cases} 0 & \text{if } P_j = 1 \\ \log_2 C & \text{if } P_j = \dfrac{1}{C} \end{cases}$$

Criteria for Splitting Tests – 2

- Gini index

$$G(D) = 1 - \sum_{j=1}^{C} P_j^2 = 1 - \underline{P}^T \underline{P} = \sum_i P_i \sum_{j \neq i} P_j = \sum_i P_i (1 - P_i)$$

$G(D)$ *and* $H(D)$ are concave.

$$G(D) = \begin{cases} 0 & \text{if } P_j = 1 \\ 1 - \dfrac{1}{C} & \text{if } P_j = \dfrac{1}{C} \end{cases}$$

– Gini index = Expected error rate if the class label is chosen randomly from the class distribution at the node

$$G(D) = \sum_{i=1}^{C} P_i \sum_{j \neq i} P_j = 1 - \underline{P}^T \underline{P}$$

- Variance Impurity (in two class case): V(D)= P_1(1- P_1). Maximum when P_1=*1/2*.

Criteria for Splitting Tests – 3

- Gini index is a generalization of variance impurity to more than two classes

- Misclassification Impurity (MAP Classification Error)

$$M(D) = 1 - \max_j P_j$$

$$M(D) = \begin{cases} 0 & \text{if } P_j = 1 \\ 1 - \frac{1}{C} & \text{if } P_j = \frac{1}{C} \end{cases}$$

So, decrease in average impurity = Gain in purity

$$GP(D \mid T) = G(D) - \sum_{i=1}^{m} \frac{|D_i|}{|D|} G(D_i)$$

Impurity, Gini Index and Entropy

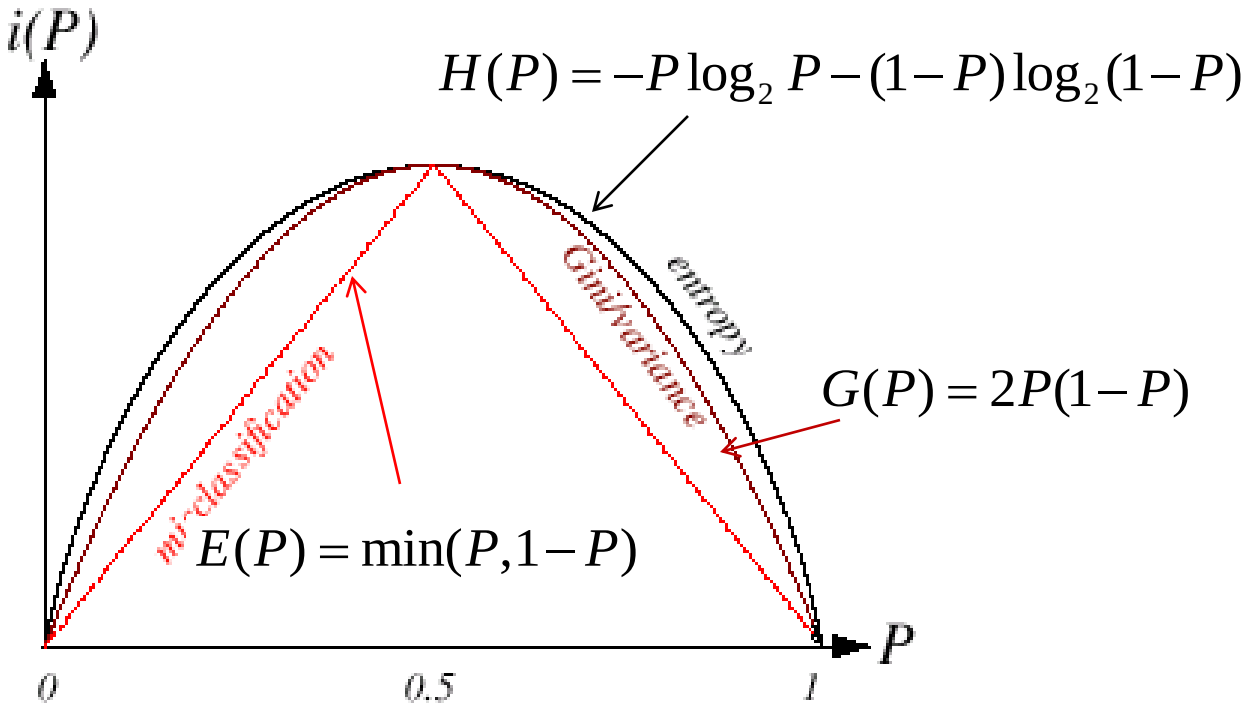


FIGURE For the two-category case, the impurity functions peak at equal class frequencies and the variance and the Gini impurity functions are identical. The entropy, variance, Gini, and misclassification impurities (given by Eqs. 1–4, respectively) have been adjusted in scale and offset to facilitate comparison here; such scale and offset do not directly affect learning or classification.

Criteria for Splitting Tests – 4

- Information gain:

$$IG(D,T) = H(D) - \underbrace{\sum_{i=1}^{m} \frac{|D_i|}{|D|} H(D_i)}_{H(D|T)}$$

Mutual Information

Stop building the tree if $IG(D,T)=0$ for all T

- Another way:

$$H(D,T) = H(D) + H(T|D)$$
$$= H(T) + H(D|T)$$
$$\Rightarrow H(D) - H(D|T) = H(T) - H(T|D)$$
$$\Rightarrow IG(D,T) = H(D) - H(D|T)$$
$$= H(T) - H(T|D)$$
$$= H(T) \quad \text{if } T \text{ is perfect.}$$

$$\text{Recall } H(X|Y) = -\sum_j P(y_j) H(X|Y=y_j) = -\sum_j P(y_j) \sum_i P(x_i|y_j) \log_2 P(x_i|y_j)$$

Example

Attributes Labels

t_1	t_2	z	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1

Which one do we choose t_1 or t_2?
Prior: z = + or – equally likely
$P(t_1=T) = 0.4$; $P(t_2=T)=0.7$
$P(t_1=T|z=+) = 0.8$; $P(t_1=T|z=-)=0$

$IG(t_1,z) = H(z) - H(z|t_1) = H(t_1)-H(t_1|z)$
$H(z) = -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1$
$H(z|t_1) = P(t_1=T)H(z|t_1=T) + P(t_1=F)H(z|t_1=F)$
$= -4/10\,(1\log_2 1 + 0\log_2 0) - 6/10\,(5/6\log_2 5/6 + 1/6\log_2 1/6)$
$= 0.39$

Information gain $IG(t_1,z)$ =1-0.39=0.61; Similarly $IG(t_2,z) = 0.12$

$H(t_1) = -(0.4)\log_2(0.4) - (0.6)\log_2(0.6) = 0.971$
$H(t_1|z) = P(z=+)H(t_1|z=+)+P(z=-)H(t_1|z=-) = -0.5\,(0.8\log_2 0.8 + 0.2\log_2 0.2)$
$= 0.361$
$IG(t_1,z)$=0.971-0.361=0.61; Similarly, $IG(t_2,z) = 0.12 \Rightarrow t_1$ is a better test

Criteria for Splitting Tests – 5

- Properties:

$$\left.\begin{array}{l} GP(\mathbf{D}\,|\,\mathrm{T}) \geq 0 \\ IG(\mathbf{D},T) \geq 0 \end{array}\right\}$$ equal to zero if and only if T provides no information

Some authors consider the gain ratio

$$0 \leq \frac{IG(\mathbf{D}\,|\,T)}{-\sum_{i=1}^{n} \frac{|D_i|}{\mathbf{D}} \log_2 \frac{|D_i|}{\mathbf{D}}} = \frac{IG(\mathbf{D}\,|\,T)}{H(\mathbf{D})} = 1 - \frac{H(D\,|\,T)}{H(D)} \leq 1$$

- Other Measures: G-statistic (related to mutual information), half-split partitioning ("two-ing" rule), linear discriminant splits

- "Deeper" Information- theoretic splits
  - Joint Mutual Information (JMI)
  - Conditional Mutual Information Maximization (CMIM)

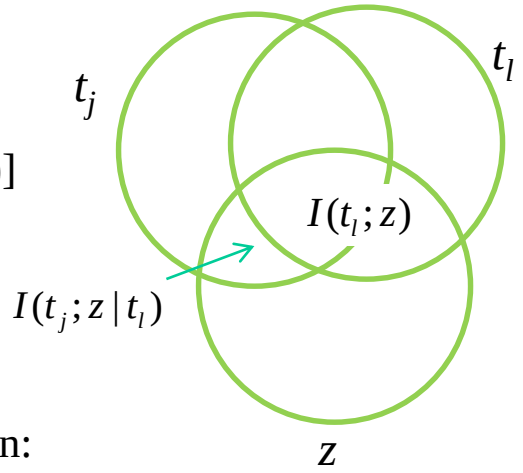
JMI & CMIM Criteria

T_k = Current Test set used on the path leading to node k

Joint Mutual Information (JMI) Criterion: (works well)

$$j_k = \arg\max_j \sum_{l \in T_k} I(\{t_l, t_j\}; z) = \arg\max_j \sum_{l \in T_k} [I(t_l; z) + I(t_j; z \mid t_l)]$$

$$= \arg\max_j \sum_{l \in T_k} I(t_j; z \mid t_l)$$



Conditional Mutual Information Maximization (CMIM) Criterion:

$$j_k = \arg\max_j \min_{l \in T_k} \left[I(t_j; z \mid t_l) \right]$$

Binary Test Case:

$$P(t_j \mid z = i) = \begin{cases} 1 - \mu_{ij} & \text{for } t_j = 0 \\ \mu_{ij} & \text{for } t_j = 1 \end{cases}; j = 1, 2, .., p; i = 0, 1, 2, .., C$$

$$P(z = i) = P_i; i = 0, 1, 2, .., C; 0 \Rightarrow \text{ none of the classes}$$

$$P(t_j = 1) = q_j = \sum_{i=0}^{C} P_i \mu_{ij}$$

Binary Test Case

$$JMI:\ j_k = \arg\max_j \sum_{l \in T_k} I(t_j; z \mid t_l)$$

Recall $I(t_j; z \mid t_l) = H(z \mid t_l) - H(z \mid t_l, t_j) = H(z, t_l) - H(t_l) - H(z, t_l, t_j) + H(t_l, t_j)$

$$where\ H(z, t_l) = -\sum_{i=0}^{C} \{P_i \mu_{il} \log_2 [P_i \mu_{il}] + P_i (1 - \mu_{il}) \log_2 [P_i (1 - \mu_{il})]\}$$

$$H(t_l) = -[q_l \log_2 q_l + (1 - q_l) \log_2 (1 - q_l)]; q_l = \sum_{i=0}^{C} P_i \mu_{il}$$

$$H(z, t_l, t_j) = -\sum_{i=0}^{C} \{P_i \mu_{il} \mu_{ij} \log_2 [P_i \mu_{il} \mu_{ij}] + P_i (1 - \mu_{il}) \mu_{ij} \log_2 [P_i (1 - \mu_{il}) \mu_{ij}]$$
$$+ P_i \mu_{il} (1 - \mu_{ij}) \log_2 [P_i \mu_{il} (1 - \mu_{ij})] + P_i (1 - \mu_{il})(1 - \mu_{ij}) \log_2 [P_i (1 - \mu_{il})(1 - \mu_{ij})]$$

$$H(t_l, t_j) = -q_{lj}^{(00)} \log_2 q_{lj}^{(00)} - q_{lj}^{(10)} \log_2 q_{lj}^{(10)} - q_{lj}^{(01)} \log_2 q_{lj}^{(01)} - q_{lj}^{(11)} \log_2 q_{lj}^{(11)}$$

$$where\ q_{lj}^{(00)} = \sum_{i=0}^{C} P_i (1 - \mu_{il})(1 - \mu_{ij}); q_{lj}^{(10)} = \sum_{i=0}^{C} P_i \mu_{il} (1 - \mu_{ij}); q_{lj}^{(01)} = \sum_{i=0}^{C} P_i (1 - \mu_{il}) \mu_{ij}; q_{lj}^{(11)} = \sum_{i=0}^{C} P_i \mu_{il} \mu_{ij}$$

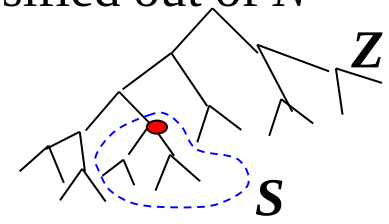
Simplifying

$$I(t_j; z \mid t_l) = -\sum_{i=0}^{C} \{\frac{p_i \mu_{il}}{q_l} \log_2 \left[\frac{p_i \mu_{il}}{q_l}\right] + \frac{p_i (1 - \mu_{il})}{(1 - q_l)} \log_2 \left[\frac{p_i (1 - \mu_{il})}{(1 - q_l)}\right]\}$$
$$- \sum_{i=0}^{m} \{\frac{p_i \mu_{il} \mu_{ij}}{q_{lj}^{(11)}} \log_2 \left[\frac{p_i \mu_{il} \mu_{ij}}{q_{lj}^{(11)}}\right] + \frac{p_i (1 - \mu_{il}) \mu_{ij}}{q_{lj}^{(01)}} \log_2 \left[\frac{p_i (1 - \mu_{il}) \mu_{ij}}{q_{lj}^{(01)}}\right]$$
$$+ \frac{p_i \mu_{il} (1 - \mu_{ij})}{q_{lj}^{(10)}} \log_2 \left[\frac{p_i \mu_{il} (1 - \mu_{ij})}{q_{lj}^{(10)}}\right] + \frac{p_i (1 - \mu_{il})(1 - \mu_{ij})}{q_{lj}^{(00)}} \log_2 \left[\frac{p_i (1 - \mu_{il})(1 - \mu_{ij})}{q_{lj}^{(00)}}\right]$$

Pruning the Tree - 1

- Overfitting: Decision Trees Overfit (low bias and high variance)
  - Real-life data is noisy, e.g., continuous attributes have measurement noise, discrete attributes such as color depend on subjective interpretation, instances are misclassified, and mistakes are made in recording.
  - Noisy data leads to larger trees . . fit the noise in addition to the structure in the data . . tend to be "*brittle*" or "*sensitive*". *"High Variability and Low Bias".*
  - Can prevent overfitting by stopping (e.g., fixed depth) or by pruning back the full tree to an appropriate size. *Pruning is preferred*. Also, *ensemble trees* reduce variance.
- Cost-Complexity Pruning:
  - Suppose have a tree Z.
  - Let $E(\mathbf{Z})$ be the number of classes misclassified out of N
  - Let $L(\mathbf{Z})$ be the number of leaves of Z

    Cost complexity of Z is: $\frac{E(\mathbf{Z})}{N} + \alpha L(\mathbf{Z})$



Pruning the Tree - 2

– Suppose we prune Z by replacing a sub-tree S by a leaf and identifying the most frequent class among the instances from which S was constructed $\Rightarrow$ we now have a tree $\hat{Z}$ with $L(S)-1$ fewer leaves. That is,

$$L\left(\hat{Z}\right) = L(Z) - \left(L(S) - 1\right)$$

➢ Suppose the new error rate is

$$\frac{E\left(\hat{Z}\right)}{N}$$

$\Rightarrow$ Prune S if

$$\frac{E(Z)}{N} + \alpha L(Z) > \frac{E\left(\hat{Z}\right)}{N} + \alpha L\left(\hat{Z}\right)$$

or

$$\alpha\left(L(Z) - L\left(\hat{Z}\right)\right) > \frac{E\left(\hat{Z}\right) - E(Z)}{N}$$

Pruning the Tree - 3

or, $\alpha > \underbrace{\frac{E(\hat{Z}) - E(Z)}{N} \frac{1}{(L(S)-1)}}_{g(Z,\hat{Z})}$

So, if $g(Z,\hat{Z}) = \frac{\Delta E}{N} \frac{1}{(L(S)-1)}$ $\begin{cases} < \alpha & \text{prune} \\ > \alpha & \text{do not prune} \end{cases}$

- Algorithm
  - Set $k = 0, \quad Z_0 = Z$
  - ♣ Set $\alpha = \infty$
  - Visit non terminal nodes t in bottom-up order and calculate $E(Z_t)$ and $L(Z_t)$ by summing over the descendants

$$g(t) = \frac{E(Z_t) - E(Z_k)}{N\left[L(Z_k) - L(Z_t)\right]} \quad \textbf{and} \quad \alpha = \min(\alpha, g(t))$$

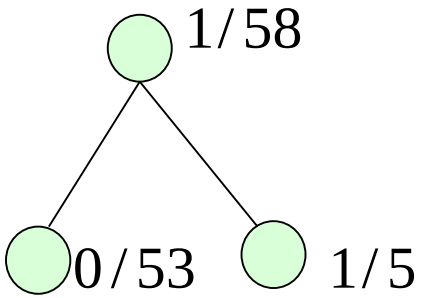
Pruning the Tree - 4

- Visit the nodes in the top-down order and prune whenever $g(t) = \alpha$
- Set $k = k+1, \quad \alpha_k = \alpha, \quad Z_k = Z$
- If z is non-trivial (>1 node), return to ♣

- Reduced error pruning :
  - Record the number of errors at each leaf node and, for each internal node, the number of errors that would be made if that node were a leaf node $\Rightarrow$ use MAP rule to make decision.
  - When all error counts are determined, each internal node is investigated starting from the bottom levels of the tree.

Pruning the Tree - 5

– Compare the number of errors made by the subtree rooted at that node with the number of errors that would result from changing the node to a leaf. If the latter is not greater than the former, prune the tree.



– Since the total number of errors made by a tree is the sum of errors at leaves, it is clear that the final sub-tree minimizes the number of errors on the pruning set.

Handling Missing Attribute Values - 1

- Missing Attribute Values:
  - Cases do not have all features
  - This can impact decision tree methods at three stages:
    - When comparing tests on attributes with different numbers of missing values
    - When partitioning set *D* on the outcomes of a chosen test, since the outcomes for some cases may not be known
    - When classifying an unseen instance whose outcome for a test is undetermined.

Handling Missing Attribute Values - 2

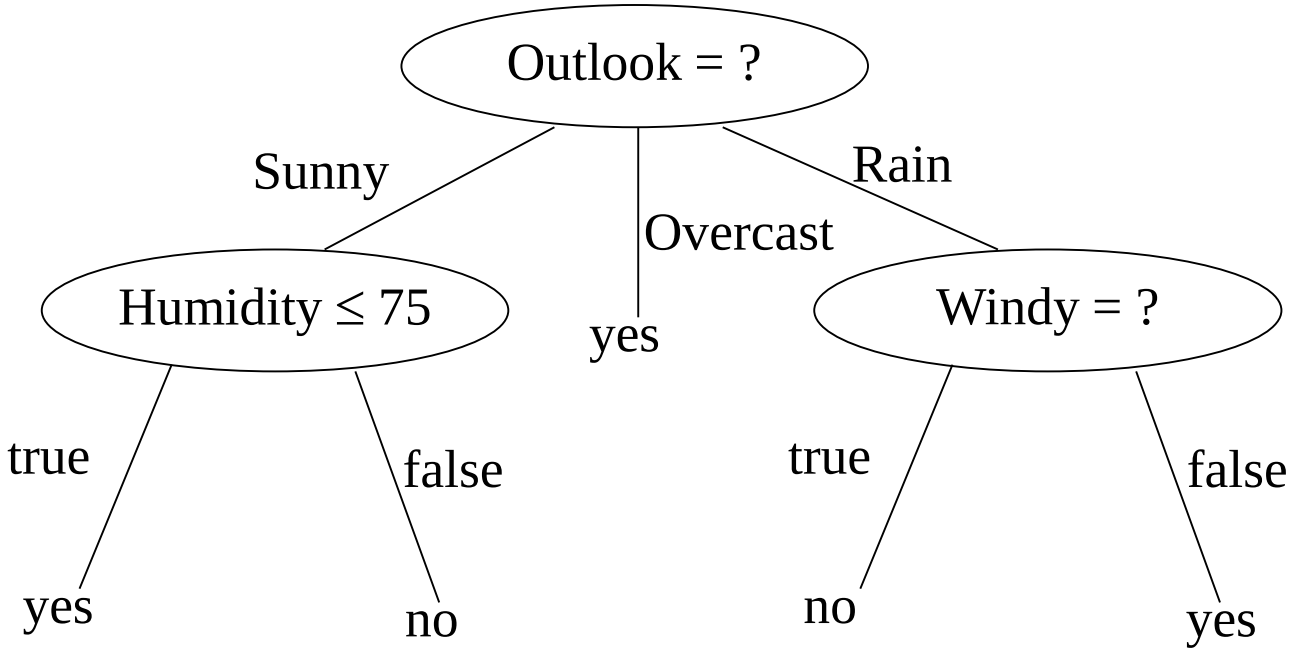
- How to handle missing values?
  - Filling in missing values
    - Discrete: pick the most frequent value
    - Continuous: use mean, median,…
  - Estimating test outcomes by some other means
    - Backup (surrogate) tests

      Test on a different attribute that produces a similar partition of *D*. So, when the value of a tested attribute is not known, the best surrogate split whose outcome is known is used to predict the outcome of the original test.

Handling Missing Attribute Values - 3

- Treating test outcome probabilistically
  - Case: Outlook = ?, Humidity = 85, Windy

Sunny	Overcast	Rain
5/14	4/14	5/14



Handling Missing Attribute Values - 4

$$P(yes \mid Humidity = 85, Windy) = \sum_{outlook} P(yes, outlook \mid Humidity = 85, windy)$$

$$= P(yes, outlook = sunny \mid Humidity = 85) + P(yes, outlook = overcast) + P(yes, outlook = rainy \mid windy)$$

$$= 0 + \frac{4}{14} + 0 = \frac{4}{14}$$

$$P(no \mid Humidity = 85, Windy) = \sum_{outlook} P(no, outlook \mid Humidity = 85, windy)$$

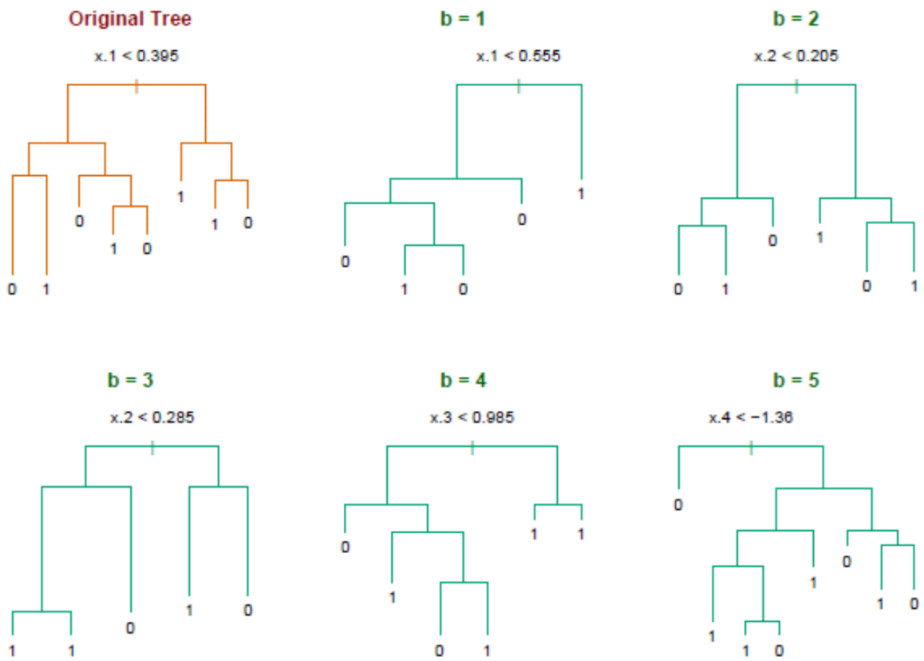
$$= P(no, outlook = sunny \mid Humidity = 85) + P(no, outlook = overcast) + P(no, outlook = rainy, windy)$$

$$= \frac{3}{3} \cdot \frac{5}{14} + \frac{2}{2} \cdot \frac{5}{14} = \frac{10}{14}$$

$\Rightarrow$ prediction: no

Ensemble Trees: Bagging

- Bagging: **B**ootstrap **Agg**regat**ing**
- Construct a decision tree from each bootstrap sample ("sample with replacement")
- Average out-of-bag sample error to evaluate performance
- Recall averaging reduces variance



- Bagging generates correlated trees
- What if we consider randomly selected features? **Random Forests**

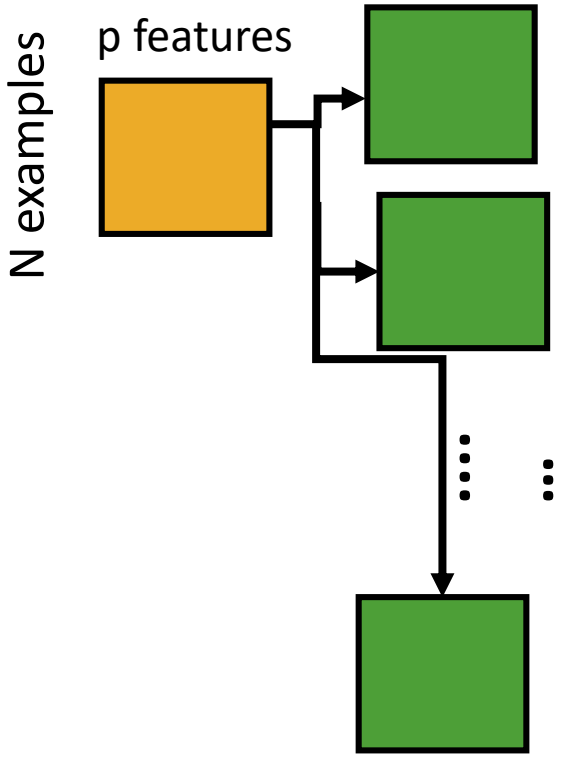
Hastie et al.,"The Elements of Statistical Learning: Data Mining, Inference, and Prediction", Springer (2009)

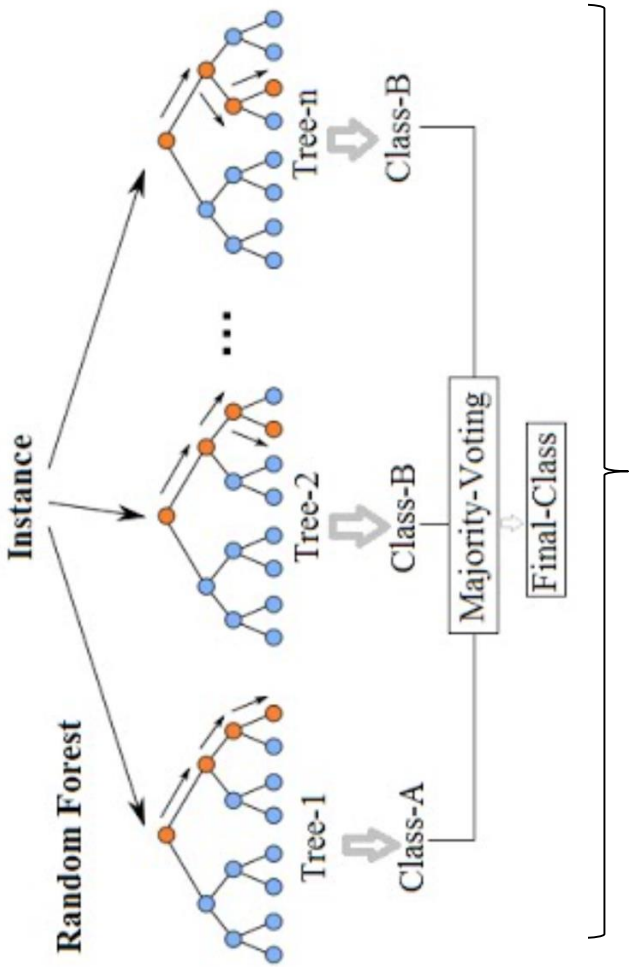
Ensemble Trees: Random Forests

Construct a decision tree from each bootstrap sample

At each node in choosing the split feature, choose only among $m<p$ features

Create bootstrap samples from the training data





Take the majority vote from all trees

Random Forests Algorithm

For b = 1 to B

(a) Draw a bootstrap sample D_b of size N from the training data.

(b) Grow a random-forest tree to the bootstrapped data, by recursively repeating the following steps at each terminal node of the tree, until the minimum node size n_{min} is reached.

i. Select m variables at random from the p variables.

ii. Pick the best variable/split-point among the m.

iii. Split the node into two child nodes.

end

Output the ensemble of trees.

To make a prediction at a new point $\underline{x}$ we do:

For regression: average the results

For classification: majority vote

- Classification: $m = p^{1/2}$ or $\log_2 p$; $n_{min} = 1$
- Regression: $m = p/3$; $n_{min} = 5$

Boosting

- Greedy algorithm that successively applies a weak learning algorithm on *weighted versions of the data* using bounds on the loss function
- Recall for binary classification ($z \in \{-1, 1\}$), loss function is

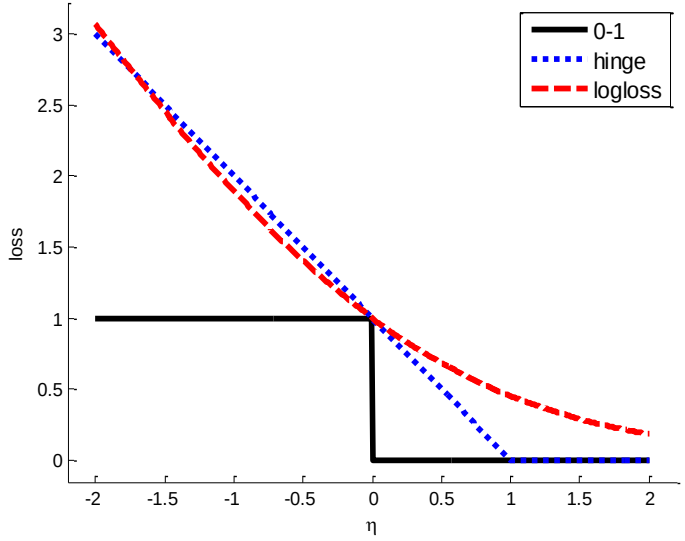
$$L(z,\hat{z}) = 1 - \text{sgn}(z\hat{z}) = 1 - \text{sgn}(\eta) = 1 - \delta_{z\hat{z}}; \eta = z\hat{z}$$

- Hinge loss used in Support Vector Machines (SVM)

$$L(z,\hat{z}) \le L_h(z,\hat{z}) = [1 - z\hat{z}]_+$$

- Exponential loss

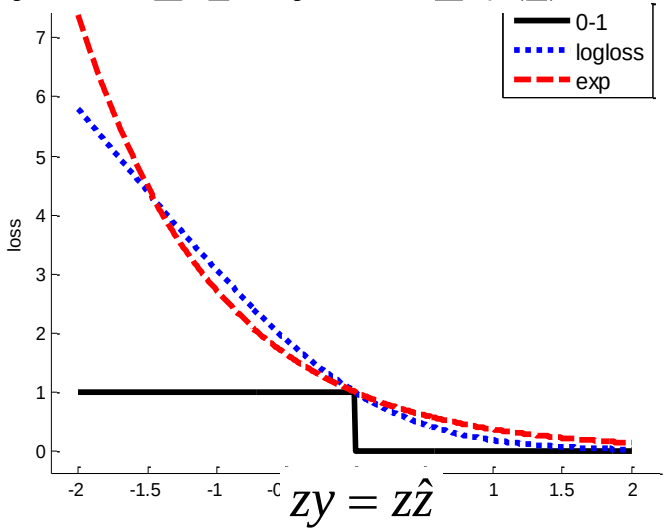
$$L(z,\hat{z}) \le L_e(z,\hat{z}) = e^{-z\hat{z}}$$



Log Loss (Logistic loss)

$$L(z,\hat{z}) \le L_l(z,y) = \ln(1+e^{-zy})/\ln 2$$

$$y = \hat{z} = \underline{w}^T \underline{x} \text{ or } y = \hat{z} = \underline{w}^T \phi(\underline{x})$$



AdaBoost

- At step m, minimize

$$L_m(f_m) = \sum_{n=1}^{N} \exp\{-z^n[\hat{z}_{m-1}(\underline{x}^n) + \alpha_m f_m(\underline{x}^n)]\} = \sum_{n=1}^{N} w_n^{(m)} e^{-\alpha_m z^n f_m(\underline{x}^n)}$$

$$w_n^{(m)} = \exp\{-z^n \hat{z}_{m-1}(\underline{x}^n)\} = \text{weight on data item } n$$

$$L_m(f_m) = e^{-\alpha_m} \sum_{n=1}^{N} w_n^{(m)} \delta_{z^n f(\underline{x}^n)} + e^{\alpha_m} \sum_{n=1}^{N} w_n^{(m)} (1 - \delta_{z^n f(\underline{x}^n)}) = \left(e^{\alpha_m} - e^{-\alpha_m}\right) \sum_{n=1}^{N} w_n^{(m)} (1 - \delta_{z^n f(\underline{x}^n)}) + e^{-\alpha_m} \sum_{n=1}^{N} w_n^{(m)}$$

$\Rightarrow$ Fit weak classifier f_m by minimizing error function $J_m = \sum_{n=1}^{N} w_n^{(m)} (1 - \delta_{z^n f(\underline{x}^n)})$

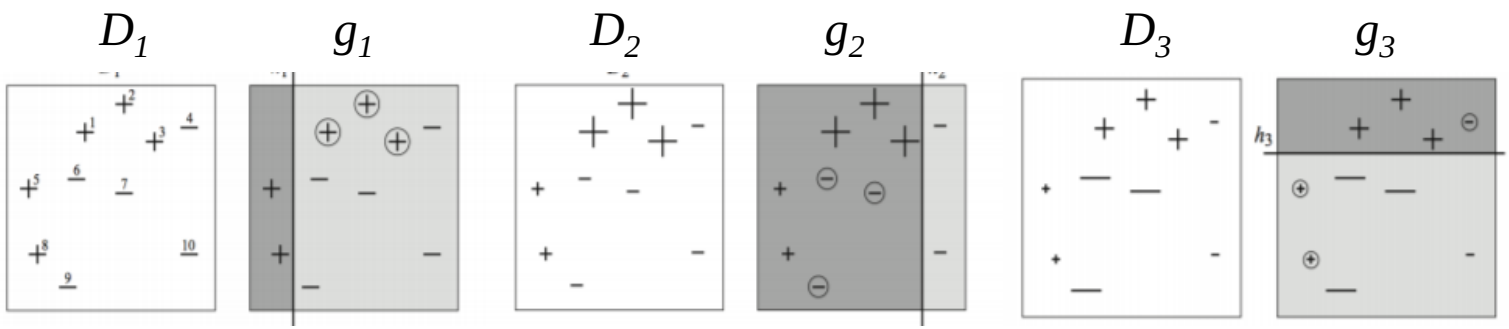
$$\text{Optimal } \alpha_m \text{ from } e^{2\alpha_m} = \frac{\sum_{n=1}^{N} w_n^{(m)} \delta_{z^n f(\underline{x}^n)}}{\sum_{n=1}^{N} w_n^{(m)} (1 - \delta_{z^n f(\underline{x}^n)})}$$

$$= \frac{\sum_{n=1}^{N} w_n^{(m)} \delta_{z^n f(\underline{x}^n)} / \sum_{n=1}^{N} w_n^{(m)}}{\sum_{n=1}^{N} w_n^{(m)} (1 - \delta_{z^n f(\underline{x}^n)}) / \sum_{n=1}^{N} w_n^{(m)}} = \frac{1 - e_m}{e_m} \Rightarrow \alpha_m = \frac{1}{2} \ln\left(\frac{1 - e_m}{e_m}\right)$$

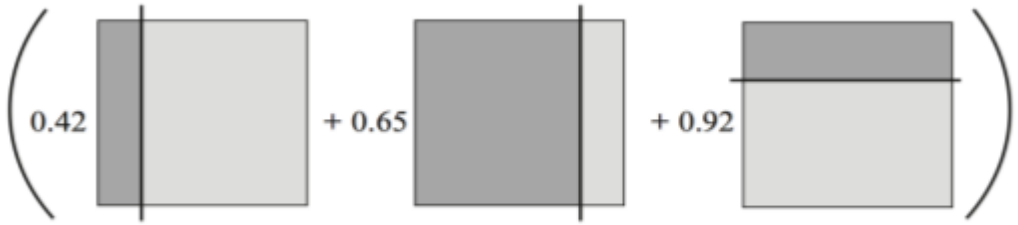
$$\Rightarrow w_n^{(m+1)} = w_n^{(m)} e^{-\alpha_m z^n f_m(\underline{x}^n)} = \begin{cases} w_n^{(m)} e^{-\alpha_m} & \text{if } f_m(\underline{x}^n) = z^n \\ w_n^{(m)} e^{\alpha_m} & \text{if } f_m(\underline{x}^n) \neq z^n \end{cases}$$

$$\hat{z} = \text{sgn}[\sum_{m=1}^{M} \alpha_m f_m(\underline{x}^n)]; M \approx 10 - 20$$

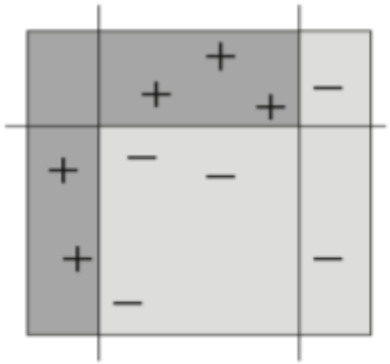
AdaBoost in Action



$$\hat{z} = \text{sgn}[\sum_{m=1}^{3} \alpha_m f_m(\underline{x}^n)] = \text{sign}$$



$=$



Gradient Boosting Algorithm

- L_2 Boosting for regression (recall residual = negative gradient wrt $\hat{z}_m$ … Gradient boosting … similar to iterative improvement of LS)

$$L(z^n, \hat{z}_m^n(\underline{x}^n)) = \left(r_m^n - f_m(\underline{x}^n)\right)^2; r_m^n = z^n - \hat{z}_{m-1}^n(\underline{x}^n); \hat{z}_m^n(\underline{x}^n) = \hat{z}_{m-1}^n(\underline{x}^n) + f_m(\underline{x}^n)$$

- Exponential loss is sensitive to outliers. Instead use **logloss** … Logitboost for binary …. Gradient boosting

Preselected network Structure (e.g., tree)

$$L_m(f_m) = \sum_{n=1}^{N} \ln\{1+\exp\{-z^n[\hat{z}_{m-1}(\underline{x}^n) + f_m(\underline{x}^n)]\}\} + \Omega(f_m) \ldots \Omega(.) \text{ regularization term}$$

$$\approx \sum_{n=1}^{N} [\ln\{1+\exp[-z^n \hat{z}_{m-1}(\underline{x}^n)]\} + f_m(\underline{x}^n)\left(\delta_{z^n,1}[g(\hat{z}_{m-1}(\underline{x}^n)) - 1] + \delta_{z^n,-1} g(\hat{z}_{m-1}(\underline{x}^n))\right)$$

$$+ \frac{1}{2}[f_m(\underline{x}^n)]^2 g(\hat{z}_{m-1}(\underline{x}^n))(1 - g(\hat{z}_{m-1}(\underline{x}^n)))] + \Omega(f_m)$$

- Gradient boosting is valid for general loss functions (see Friedman, '99)

$$L_m(f_m) = \sum_{n=1}^{N} L(z^n, \hat{z}_{m-1}(\underline{x}^n) + f_m(\underline{x}^n)) + \Omega(f_m) \ldots \Omega(.) \text{ regularization term}$$

$$\approx \sum_{n=1}^{N} \{L(z^n, \hat{z}_{m-1}(\underline{x}^n)] + g_n f_m(\underline{x}^n) + \frac{1}{2} h_n [f_m(\underline{x}^n)]^2\} + \Omega(f_m); g_n = \frac{\partial L(z^n, \hat{z}_{m-1}(\underline{x}^n))}{\partial \hat{z}_{m-1}(\underline{x}^n)}; h_n = \frac{\partial^2 L(z^n, \hat{z}_{m-1}(\underline{x}^n))}{\partial [\hat{z}_{m-1}(\underline{x}^n)]^2}$$

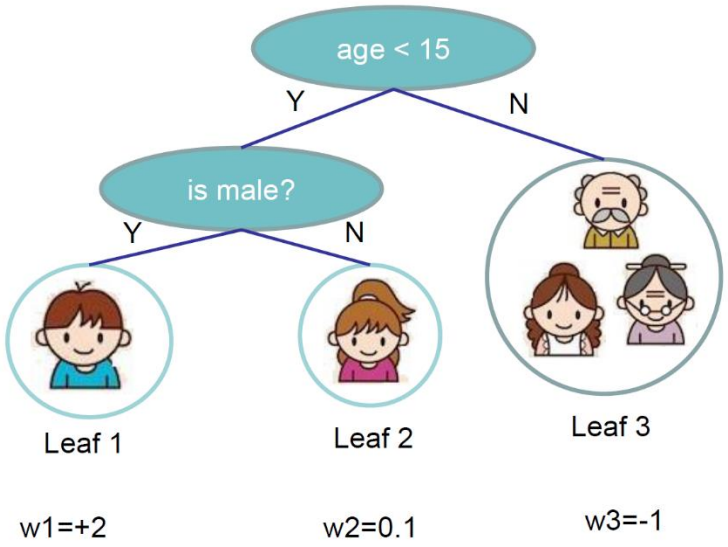
Gradient Boosting and Regularization

- Complexity of a tree and regularization

$$\Omega(f_m) = \gamma T + \frac{1}{2}\sum_{j=1}^{T} w_j^2; T = \#\text{ of leaves}$$

w_j = weight of leaf j (e.g., expected cost of path in the tree leading to leaf j)

$f_m(\underline{x}^n) = w_j$ if $\underline{x}^n$ is mapped to leaf j, i.e., $q(\underline{x}^n) = j; q$ is the mapping fuunction



$$T = 3$$

$$\Omega(f_m) = 3\gamma + \frac{1}{2}\lambda(4 + 0.01 + 1)$$

$$= 3\gamma + 2.505\lambda$$

Figure taken from Tianqi Chen "Introduction to Boosted Trees," October 22, 2014.

Gradient Boosting Implementation

- Define the data instance set in leaf j as $I_j = \{n : q(\underline{x}^n)=j\}$

$$L_m(f_m) \approx \sum_{n=1}^{N}\{L(z^n, \hat{z}_{m-1}(\underline{x}^n)] + g_n f_m(\underline{x}^n) + \frac{1}{2} h_n[f_m(\underline{x}^n)]^2\} + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$$

$$= \sum_{j=1}^{T}\{\underbrace{\left(\sum_{n\in I_j} g_n\right)}_{G_j} w_j + \frac{1}{2}(\sum_{n\in I_j} h_n + \lambda) w_j^2 + \gamma\} + \text{constant..... separable in } w_j \text{.... but huge possibilities for } \{I_j\}!$$

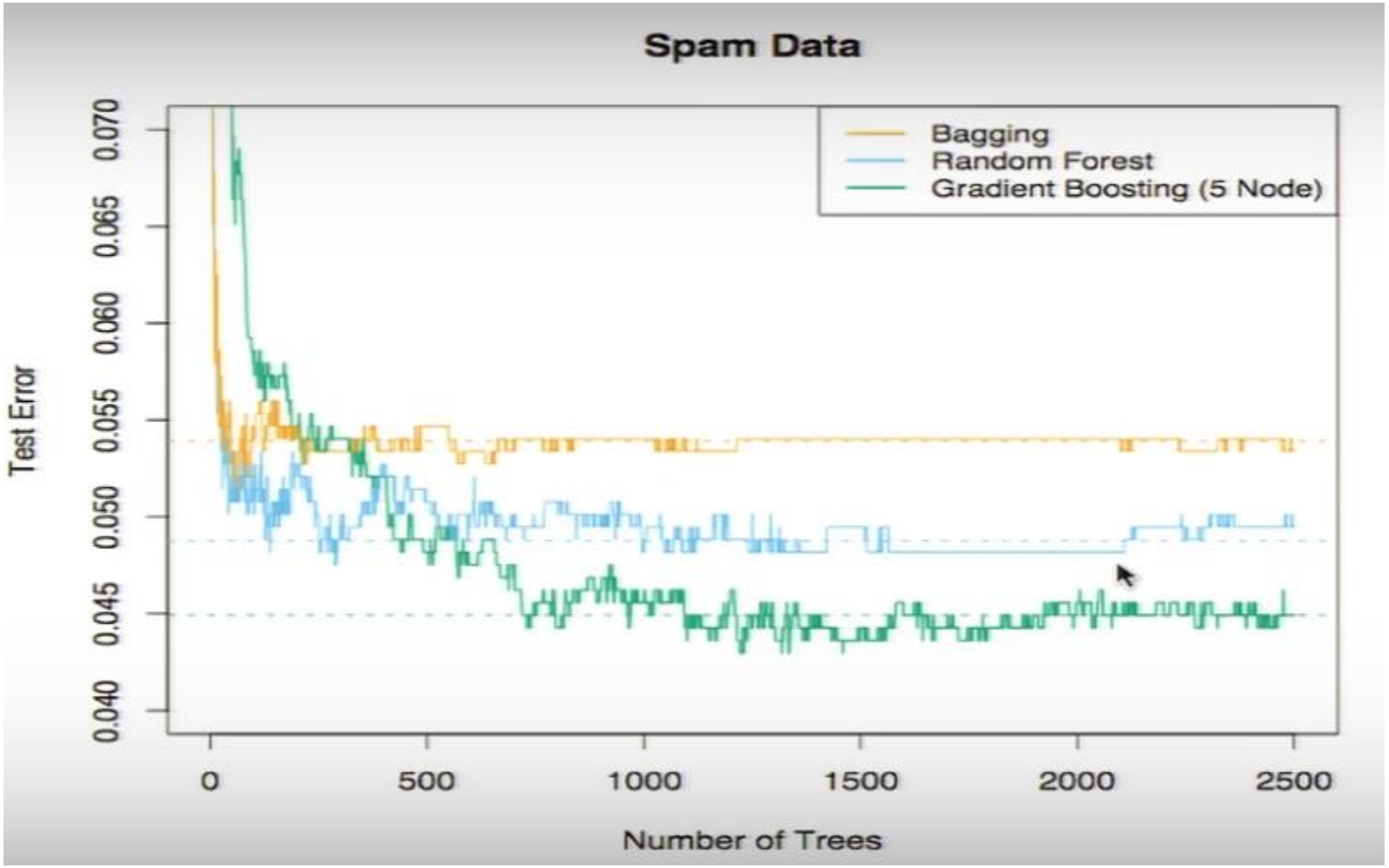
$$\text{For known } \{I_j\}, \textit{optimal } w_j = -\frac{G_j}{H_j + \lambda}; L_m(f_m) = \frac{1}{2}\sum_{j=1}^{T}(\frac{-G_j^2}{H_j + \lambda} + 2\gamma)$$

- Greedy construction of tree : Binary split at node j with $G_{Lj}, H_{Lj}, G_{Rj}, H_{Rj}$

$$Gain = \frac{1}{2}[\frac{G_{Lj}^2}{H_{Lj} + \lambda} + \frac{G_{Rj}^2}{H_{Rj} + \lambda} - \frac{G_j^2}{H_j + \lambda}] - \gamma; G_j = G_{Lj} + G_{Rj}; H_j = H_{Lj} + H_{Rj}$$

  - Continuous: Left to right linear scan over sorted instance is enough to decide the best split along the feature
  - Categorical: one-hot coding and ask the question: Is it category *k*?
  - Stop if the best gain is negative or recursively prune all the leaf splits with negative gain after the tree is built
  - To prevent overfitting, use a shrinkage factor: $\hat{z}_m = \hat{z}_{m-1} + \varepsilon f_m; \varepsilon \approx 0.1$

Bagging vs. Random Forests vs. Gradient Boosting



Hastie's Talk: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

Error Correcting Output Codes

- Multiclass problems
  - *M* class problem: *M* two-class tasks or *M(M-1)/2* two-class tasks
  - Obtain a separate decision tree for each class
  - What if more than one tree classifies? Majority rule

- Error Correcting Output Codes:
  - Each class: a *p* bit pattern chosen so that Hamming distance is as large as possible among classes
  - Learn each of the p bits via trees $\Rightarrow$ p trees
  - Select nearest class in terms of least Hamming distance
- L_K-Boosted Tree (Friedman, 1999)
  - Gradient and diagonal Hessian are easy to compute (see slide 13)
  - Gradient boosting algorithm
  - Implementation: https://github.com/tqchen/xgboost

Summary

- MMSE Estimate = Conditional Mean
- Approximating Posterior Probabilities (Discriminative Learning)
  - Linear Discriminants in Equal Covariance Case
  - Leads to Logistic Sigmoidal Function
  - Learn Weights via NLP
- Single Layer Perceptrons
  - Perceptron Learning Theorem
  - Relaxation and Normalized Updates
  - Extension to Multiple Classes
  - Fuzzy Updates
  - LVQ as a Perceptron
- Capacity of a Perceptron
- Decision Trees, Bagging, Random Forests, Gradient Boosting