Lectures 6 and 7: 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

Spring 2021
February 23 & March 1,2021

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

- 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
  - Constructing decision trees, Choosing Tests, Splitting Rules, Pruning Rules, Handling Missing Values, Extensions to Complex Tests
  - Bagging Decision Trees
  - Random Forest
  - Boosting: AdaBoost; Gradient Boosting

Nonlinear Discriminants - 1

- Why should we use only linear $y(\underline{x}, \underline{w}_i)$?

  $\exists$ numerous possibilities and we will revisit this issue later.

  1. **Additive models and smoothers** (Hastie and Tibshirani)

$$y(\underline{x}, \underline{w}_k) = y_k = \sum_{i=1}^{p} \sum_{j=1}^{M} w_{kij} \left(x_i\right)^j - w_{k0}$$

Polynomial of degree M

  - o Generalized additive models allow nonlinear transformations of data
  - o Do not allow interactions among $\left\{x_i^{\ s}\right\}$
  - o Other possibilities: splines, interaction functions,….

$$y(\underline{x}, \underline{w}_k) = y_k = \sum_{i=1}^{p} \sum_{j=1}^{p} w_{kij} x_i x_j + \sum_{i=1}^{p} w_{ki} x_i - w_{k0}$$

Nonlinear Discriminants - 2

2. **Projection Pursuit Regression (PPR)**

$$y(\underline{x}, \underline{w}_k) = \sum_{j=1}^{M} w_{kj} \varphi_{kj} (\underline{u}_j^T \underline{x} - u_{j0}) - w_{k0} \qquad \cdots\cdots \quad \text{MLP}$$

The parameters $\{\underline{u}_j \text{ and } u_{j0}\}$ define the projection of $\underline{x}$ onto a set of planes: $\underline{u}_j^T \underline{x} - u_{j0} = \alpha$ as in MLP. These projections are transformed via nonlinear actuation functions and these in turn are linearly combined to form the $y(.)$ variables. Note that $\phi(.)$ can be a function of *k*.

3. **Radial Basis Functions**

$$y(\underline{x}, \underline{w}_k) = -w_{k0} + \sum_{j=1}^{M} w_{kj} G\left(\frac{\left\| \underline{x} - \underline{\mu}_{kj} \right\|}{\sigma_{kj}} \right)$$

where $G(r) = e^{-r^2/2}$; Gaussian kernel $\Rightarrow$ RBF

$$G(r) = \frac{1}{\sqrt{c^2 + r^2}}; \text{ Multiquadric}$$

$$G(r) = U(r_0 - r); \text{ similar to a kernel estimator}$$

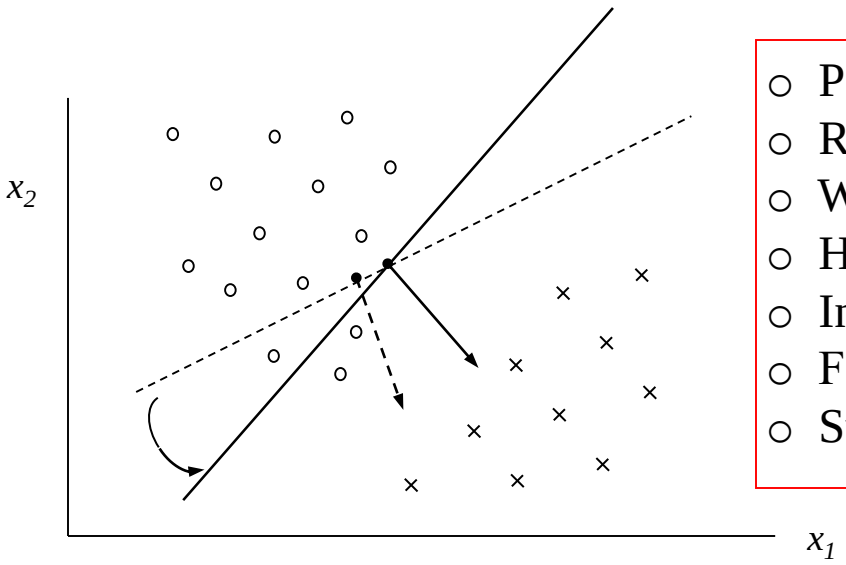
RBF is similar to clustered version of PNN.

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

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

$$\underline{e}^T$$

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

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

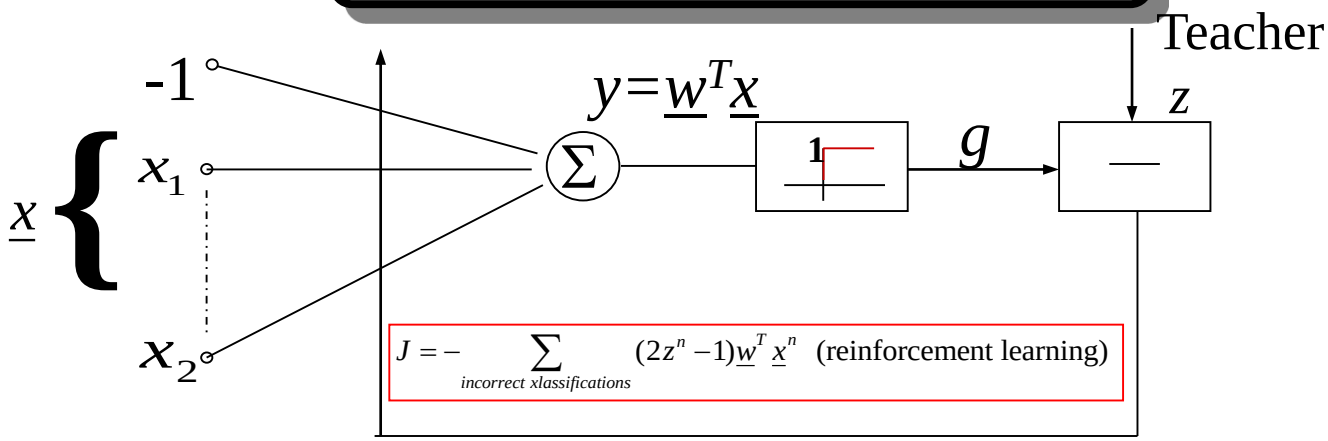
  $\underline{w} \leftarrow \underline{w} + \eta \underline{x}^{n}$

  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:

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

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

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

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

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

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}

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

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

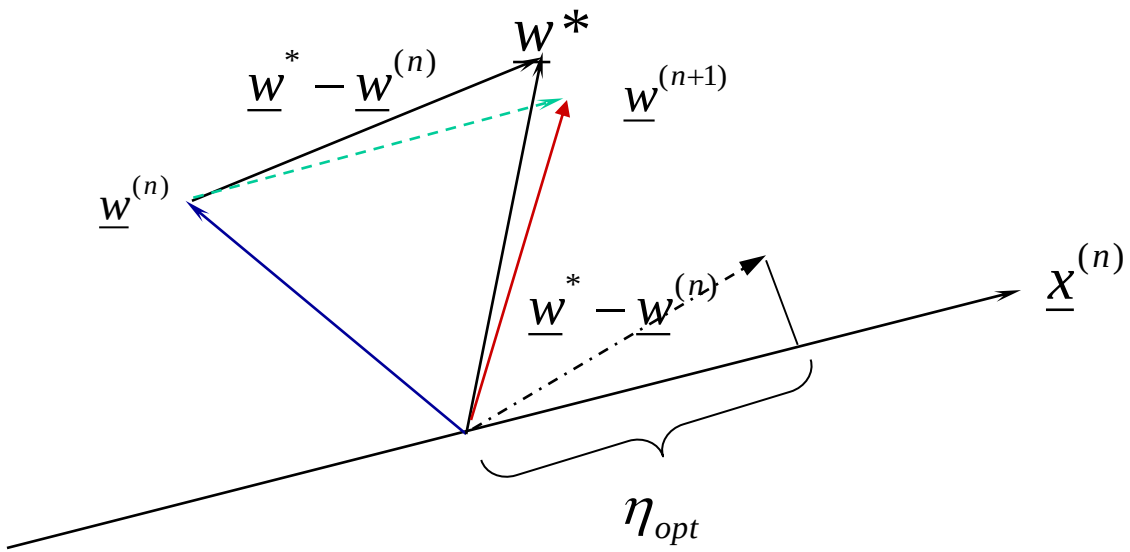
valid when don't scale also

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

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

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

Relaxation Method



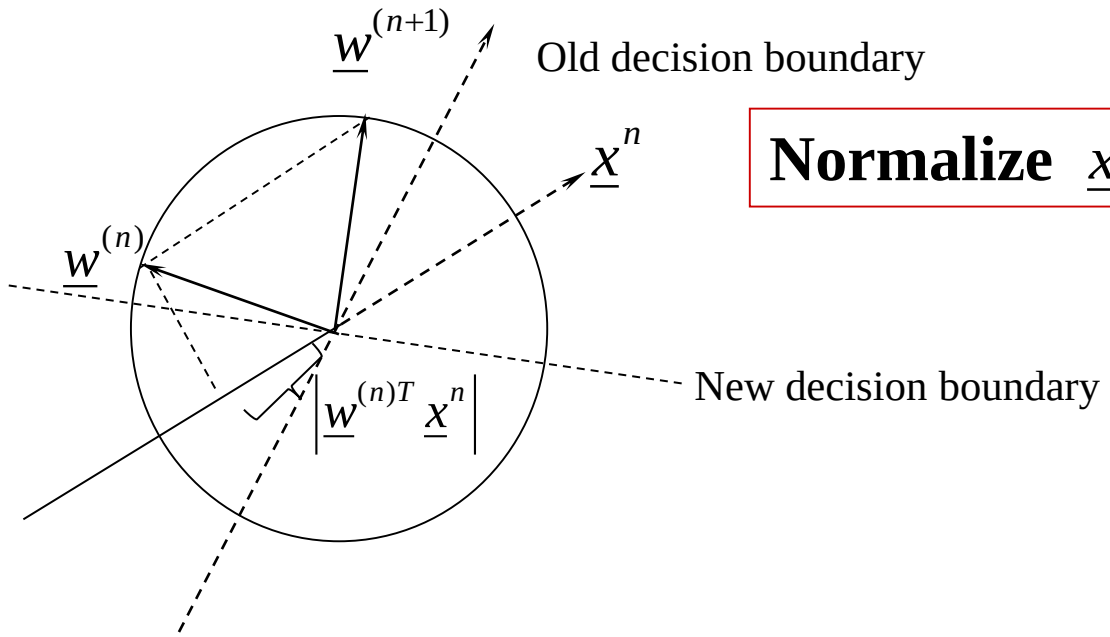
But do not know w^{*}. In *Relaxation method*, $w^{*T} x^{n}$ is replaced by a small positive number b .

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

Normalized Perceptron - 1

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

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



Normalize $\underline{x}^n$

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

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

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

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

$(p+1)$ 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

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

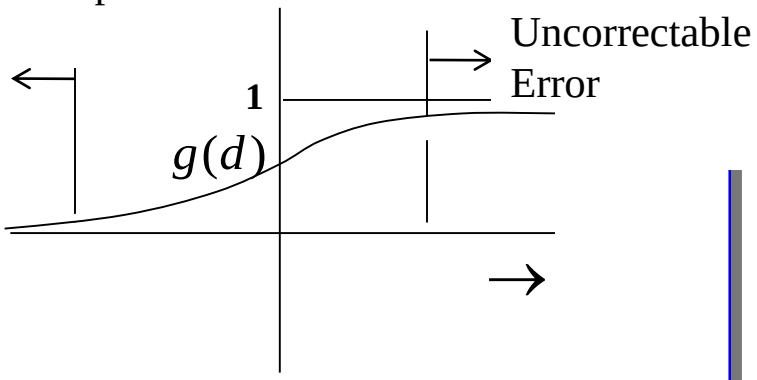
$\xi \downarrow 0 \Rightarrow$ step

Correct, but improve

Incorrect, but can be corrected

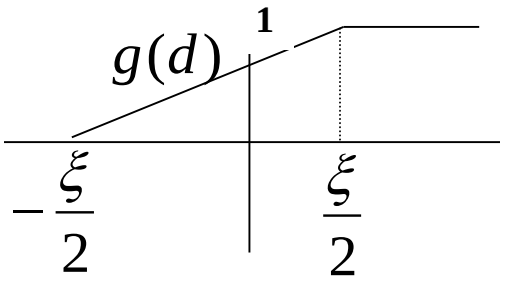
Correct, No action needed

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.*

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

- **Decision Trees : What They are and History**
  - Constructing decision trees
  - Choosing Tests
  - Splitting Rules
  - Pruning Rules
  - Handling Missing Values
  - Extensions to Complex Tests
- **Ensemble Algorithms**
  - Bagging Decision Trees
  - Random Forest
  - Boosting
    - Gradient Boosting
    - AdaBoost

What are Decision Trees?

- Common learning paradigms in AI and Statistics
- A non-parametric supervised learning method used for *classification* and *regression*
- **Idea**: Exploit regularities or features in observations to develop IF-THEN-ELSE rules and use them to predict the labels of new observations or estimate (infer) an unknown
  - Each instance I^k is represented by a collection of attributes or features $\left(x_1, x_2, \ldots, x_p\right)^k = \underline{x}^k$ (continuous /discrete) and class z^k
  - Partition the features via sequential queries

```
If rainy
    If not windy "take an umbrella"
    Else "wear a rain coat"
    End If
End If
```

History

- Developed since 1960[s] and popularized by Brieman and Quinlan

  ⇨ Tree Algorithms: ID3, C4.5, C5.0 and CART

- **Three uses**: Data description (compression, rules), classification and generalization
- Popular in statistics, pattern recognition, decision theory, signal processing and machine learning
- Have been used in industrial applications, particularly in diagnosis and quality control. For example, look at my papers on sequential fault diagnosis since 1990. Primarily in *IEEE T-SMC*.
- Why? Invariant to scaling; Can handle large datasets; Easily ignore redundant variables; Handle missing variables through surrogate splits; **Easy to interpret and explain**
- Nearly half of the data mining competitions are won by using some variants of tree ensemble methods: **Boosting, Random Forests, Bagging**
- Surveys: 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)

  S. B. Kotsiantis, "Decision trees: a Recent Overview," *Artificial Intelligence Review*, 39:261–283, 2013.

Illustrative Example

Take umbrella or Not

Case #	Outlook	Temp (°F)	Humidity (%)	Windy	Class
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

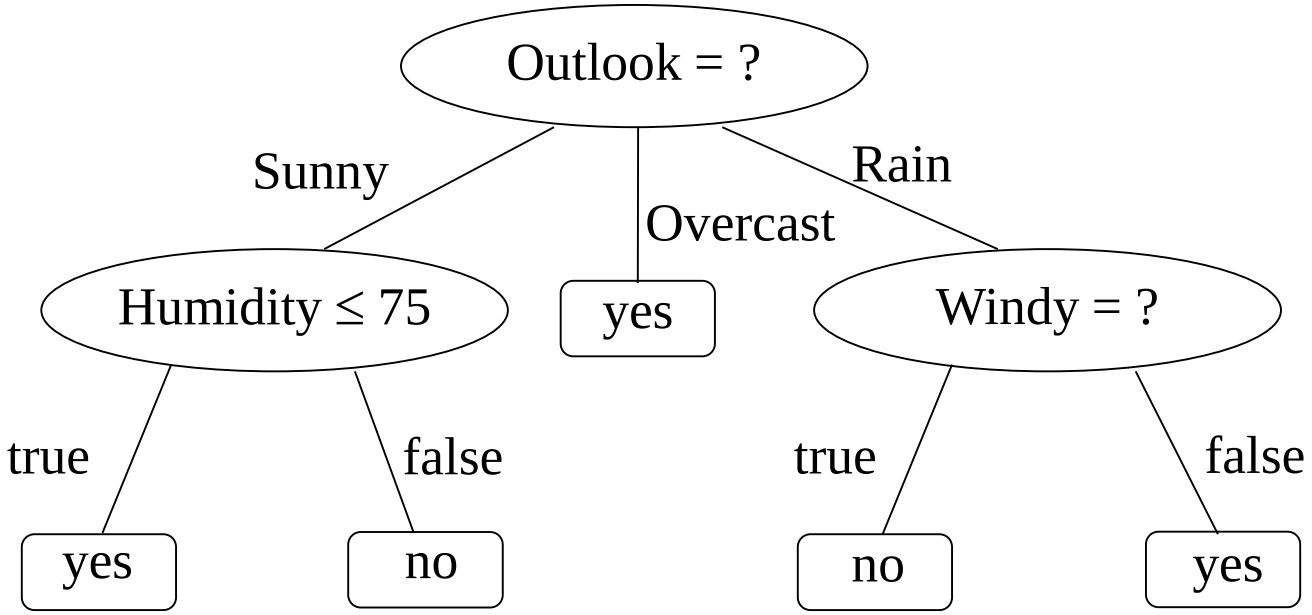
$$x_1 \in \{\text{Sunny, Overcast, Rain}\}$$
$$x_2 \in R$$
$$x_3 \in R$$
$$x_4 \in \{\text{true, false}\}$$

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

(Take umbrella or not)

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

Decision 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 Regions in a Decision Tree

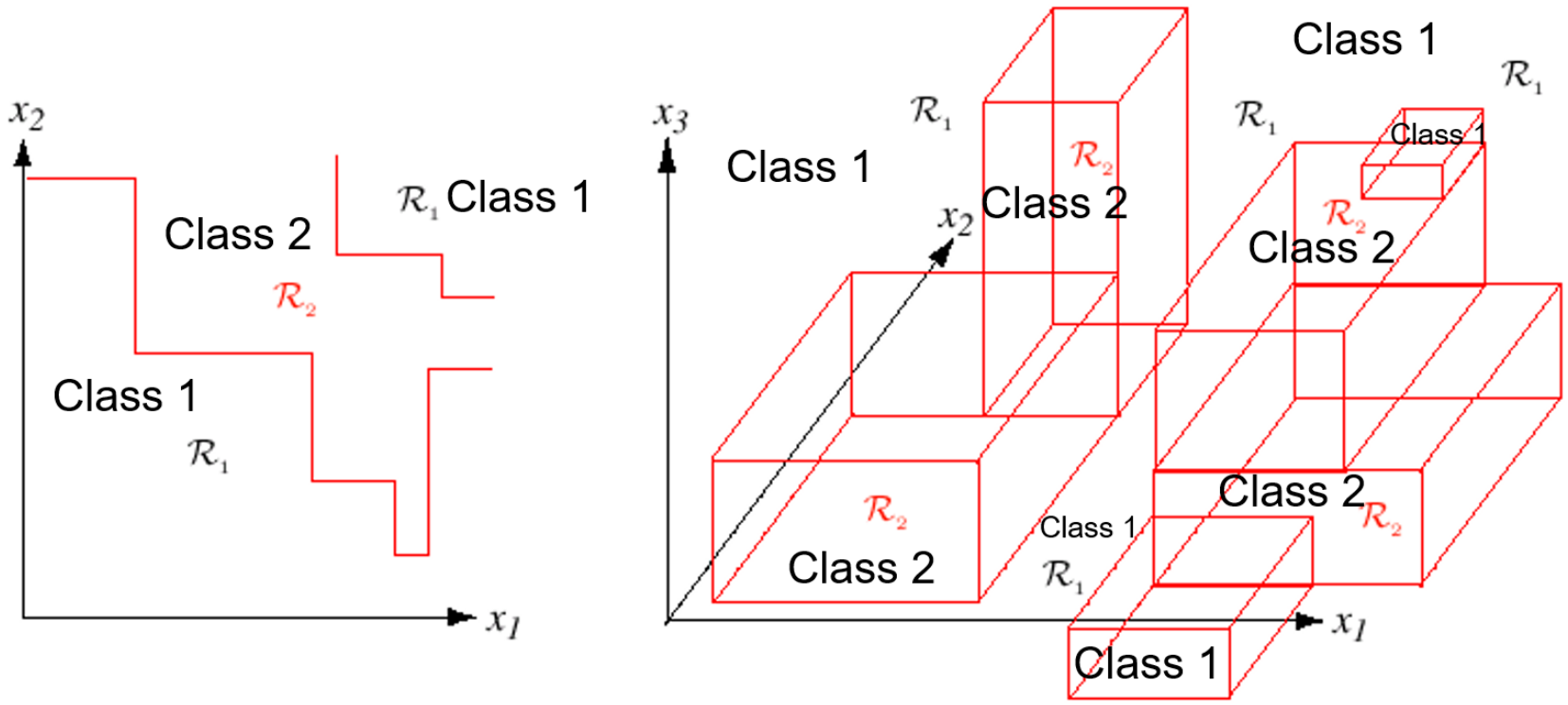


Figure from Duda, Hart and Sorkin

Attribute Tests - 1

- **Method for constructing trees -- generate compact trees**
  - All employ recursive divide-and-conquer algorithms
  - How to select tests? ⇨ also called **splitting rules**
- **Most 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 we have observations, $n_1 < n_2 < \cdots\cdots < n_N$

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

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 combinations**

$\underline{w}^T \underline{x} - w_0 \geq 0$ true or false …. Hyperplane tests

- **Each test could be a classifier** (See error correcting codes on page 77)

Information Gain for Classification with a Continuous Feature

$$z \in \{1,2,..,C\}; \underline{x} \in R^p; p(\underline{x}) = \sum_{i=1}^{C} P[\underline{x}, z=i] = \sum_{i=1}^{C} P(z=i)\, p(\underline{x} \mid z=i) = \sum_{i=1}^{C} \pi_i N(x; \mu_i, \sigma_i^2)$$

Recall for a binary test on x with a theshold t : $IG(z \mid x,t) = H(z) - H(z \mid x,t)$

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

$$H(z \mid x,t) = P(x<t)H(z \mid x<t) + P(x \geq t)H(z \mid x \geq t) = \left(\sum_{i=1}^{C} \pi_i \Phi(\frac{t-\mu_i}{\sigma_i}) \right) H(z \mid x<t) + \left(\sum_{i=1}^{C} \pi_i (1-\Phi(\frac{t-\mu_i}{\sigma_i})) \right) H(z \mid x \geq t)$$

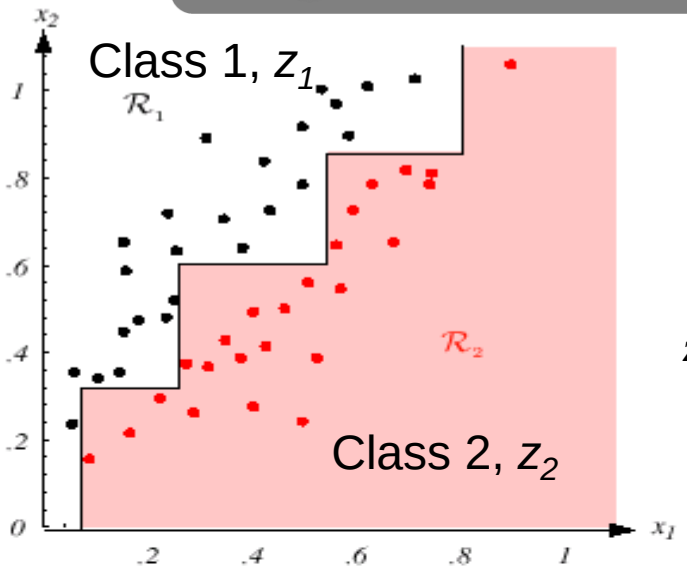
where $\Phi(c) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{c} e^{-u^2/2} du = CDF$ of a standard Normal (Gaussian) distribution.

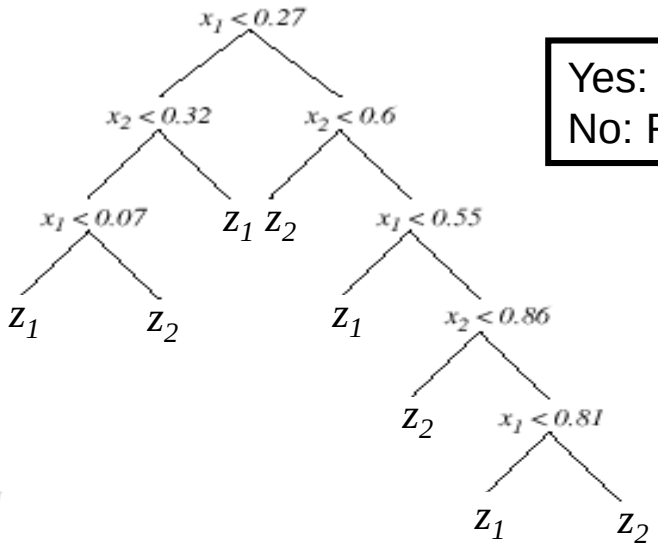
Note that by Bayes rule, $P(z=j \mid x<t) = \frac{P(x<t \mid z=j)P(z=j)}{P(x<t)} = \frac{\Phi(\frac{t-\mu_j}{\sigma_j})\pi_j}{\sum_{i=1}^{C} \pi_i \Phi(\frac{t-\mu_i}{\sigma_i})}$

So, $H(z \mid x<t) = -\sum_{j=1}^{C} \left(\frac{\Phi(\frac{t-\mu_j}{\sigma_j})\pi_j}{\sum_{i=1}^{C} \pi_i \Phi(\frac{t-\mu_i}{\sigma_i})} \right) \log_2 \left(\frac{\Phi(\frac{t-\mu_j}{\sigma_j})\pi_j}{\sum_{i=1}^{C} \pi_i \Phi(\frac{t-\mu_i}{\sigma_i})} \right)$; $H(z \mid x \geq t) = -\sum_{j=1}^{C} \left(\frac{[1-\Phi(\frac{t-\mu_j}{\sigma_j})]\pi_j}{\sum_{i=1}^{C} \pi_i [1-\Phi(\frac{t-\mu_i}{\sigma_i})]} \right) \log_2 \left(\frac{[1-\Phi(\frac{t-\mu_j}{\sigma_j})]\pi_j}{\sum_{i=1}^{C} \pi_i [1-\Phi(\frac{t-\mu_i}{\sigma_i})]} \right)$

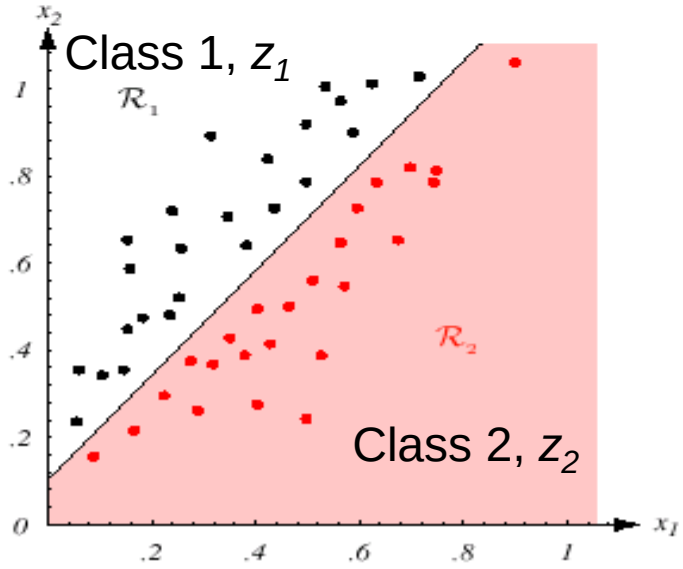
So, $IG(z \mid x,t) = -\sum_{j=1}^{C} [\pi_j \log_2 \pi_j - \Phi(\frac{t-\mu_j}{\sigma_j})\pi_j \log_2 \left(\frac{\Phi(\frac{t-\mu_j}{\sigma_j})\pi_j}{\sum_{i=1}^{C} \pi_i \Phi(\frac{t-\mu_i}{\sigma_i})} \right) - [1-\Phi(\frac{t-\mu_j}{\sigma_j})]\pi_j \log_2 \left(\frac{[1-\Phi(\frac{t-\mu_j}{\sigma_j})]\pi_j}{\sum_{i=1}^{C} \pi_i [1-\Phi(\frac{t-\mu_i}{\sigma_i})]} \right)]$

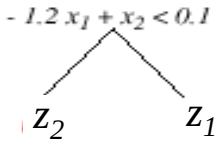
Single Attribute Tests versus Complex Tests





Yes: left
No: Right





From Duda, Hart and Sorkin

If the class of node decisions does 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.

Basic Idea of Selecting Tests

- **Idea**
  - Suppose applying a test T partitions D into $D_1, D_2, \ldots, D_m$
  - Want to select a test T such that the child nodes $D_1, D_2, \ldots, D_m$ are "*purer*" than their parent D
  - Measure of "*impurity*" is such that it is zero ("pure") if P_j is concentrated on one class and is maximal if $P_j = 1/C$ is uniform (Recall the definition of entropy), C is the number of classes
  - Stop building the tree (label the node as leaf) if all cases belong to a single class

Splitting Criteria: Entropy, Gini Index

- **Criteria**

1. **Entropy**: A measure of impurity (uncertainty)
   - $H(D) = -\sum_{j=1}^{C} P_j \log_2 P_j \qquad H(D) = \begin{cases} 0 & \text{if } P_j = 1 \\ \log_2 C & \text{if } P_j = \frac{1}{C} \end{cases}$
2. **Gini Index**
   - $G(D) = 1 - \sum_{j=1}^{C} P_j^2 = \sum_{j=1}^{C} P_j - \sum_{j=1}^{C} P_j^2 = \sum_{j=1}^{C} P_j (1 - P_j)$
   - $G(D)$ *and* $H(D)$ are concave
   - $G(D) = \begin{cases} 0 & \text{if } P_j = 1 \\ 1 - \frac{1}{C} & \text{if } P_j = \frac{1}{C} \end{cases}$
   - Expected error rate if the class label is chosen randomly from the class distribution at the node

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

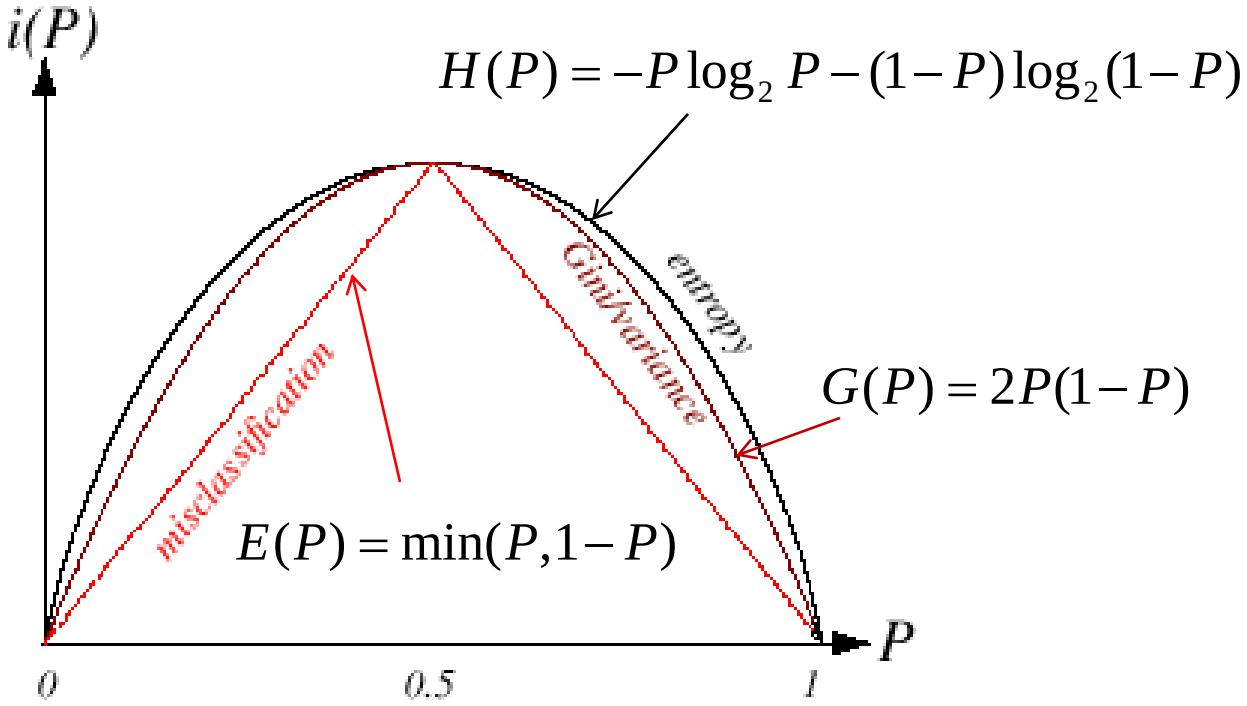
Splitting Criteria: Variance and Misclassification Impurities

3. **Variance Impurity** (in two class case): $V(D) = P_1(1 - P_1)$
   - Maximum when $P_1 = \frac{1}{2}$
   - Gini index is a generalization of variance impurity to more than two classes
4. **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 (with 2-4)

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

Impurity, Gini Index and Entropy



From Duda, Hart and Sorkin

Example of Computing *GP*

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

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

$$G(D) = \sum_{i=1}^{C} P_i(1-P_i); m = 2; C = 2$$

Variance Impurity = Gini Index here

$G(D) = (1/2).(1/2)+(1/2).(1/2)= 1/2$

$G(D|t_1) = (|D_{11}|/|D|)\, G(D_{11}) + (|D_{12}|/|D|)\, G(D_{12})$

$= (4/10)\,(0) + (6/10)[(5/6)(1/6) + (1/6)(5/6)]$

$= 0.167=0.167$

D_{jk}: j = test; k = outcome

Purity gain of t_1: $GP(D|t_1) = 0.5 - 0.167 = 0.333$

$G(D|t_2) = (|D_{21}|/|D|)\, G(D_{21}) + (|D_{22}|/|D|)\, G(D_{22})$

$= (7/10)\,[(2/7)(5/7) + (5/7)(2/7)]+(3/10)(0)$

$= (2/7) = 0.285$

Purity gain of t_2: $GP(D|t_2) = 0.5 - 0.285 = 0.215$

$\Rightarrow t_1$ **is a better test**

Example of Computing Classification Purity Gain

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

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

$$GCP(D|T) = M(D) - \sum_{i=1}^{m} \frac{|D_i|}{|D|} M(D_i) = M(D) - M(D|T)$$

$M(D) = 1/2$

$M(D|t_1) = (|D_{11}|/|D|)\, M(D_{11}) + (|D_{12}|/|D|)\, M(D_{12})$

$= (4/10)\,(0) + (6/10)(1/6)$

$= 0.10$

D_{jk}: j = test; k = outcome

Classification purity gain of t_1: $GCP(D|t_1) = 0.5 - 0.1 = 0.4$

$M(D|t_2) = (|D_{21}|/|D|)\, M(D_{21}) + (|D_{22}|/|D|)\, M(D_{22})$

$= (7/10)\,(2/5) + (3/10)(0)$

$= 0.28 = 0.28$

Classification purity gain of t_2: $GCP(D|t_2) = 0.5 - 0.28 = 0.22$

$\Rightarrow t_1$ **is a better test**

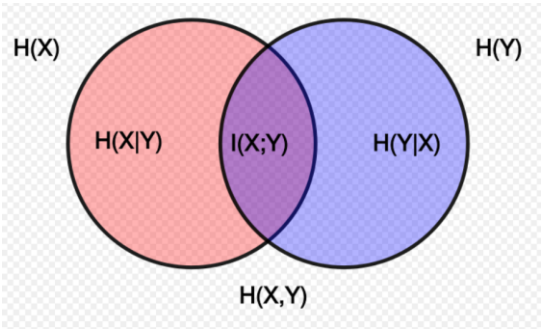
Splitting Criteria: Information Gain

5. 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: Joint Entropy

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

Recall the Venn Diagram



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

Example of Computing *IG*

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
- $P(t_1 = T) = 0.4;\ P(t_2 = T) = 0.7$
- $P(t_1 = T | z = +) = 0.8$
- $P(t_1 = T | z = -) = 0$

$IG(t_1, z) = H(z) - H(z \mid t_1) = H(t_1) - H(t_1 \mid z)$

$H(z) = -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1$

$H(z|t_1) = P(t_1 = T)\, H(z \mid t_1 = T) + P(t_1 = F)\, H(z \mid t_1 = F)$

$= -(4/10)[1\log_2 1 + 0\log_2 0] - (6/10)[(5/6)\log_2(5/6) + (1/6)\log_2(1/6)]$

$= 0.39$

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

$H(t_1) = -(0.4)\log_2(0.4) - (0.6)\log_2(0.6) = 0.971$

$H(t_1|z) = P(z = +)H(t_1 \mid z = +) + P(z = -)H(t_1 \mid z = -)$

$= -0.5\,(0.8\log_2 0.8 + 0.2\log_2 0.2) = 0.361$

Information gain: $IG(t_1, z) = 0.971 - 0.361 = 0.61$; Similarly, $IG(t_2, z) = 0.12$

$\Rightarrow t_1$ **is a better test**

Non-negativity of Information and Purity Gains

- Properties:
  - $GP(\mathbf{D}\,|\,\mathrm{T}) \geq 0$
    $IG(\mathbf{D},T) \geq 0$
    equal to zero if and only if T provides no information
  - Some authors consider the gain ratio

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

$$0 \leq \frac{GP(\mathbf{D}\,|\,T)}{G(\mathbf{D})} = 1 - \frac{G(D\,|\,T)}{G(D)} \leq 1$$

$$0 \leq \frac{GCP(\mathbf{D}\,|\,T)}{M(\mathbf{D})} = 1 - \frac{M(D\,|\,T)}{M(D)} \leq 1$$

H: Entropy
G: Gini Index
IG: Information Gain
GP: Gain in Purity
GCP: Gain in Classification Purity

In the example, the ratios for test t_1 are:
0.61 for *IG*; 0.266 for *GP*; 0.8 for *GCP*

Joint Mutual Information

6. "Deeper" Information-theoretic splits

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

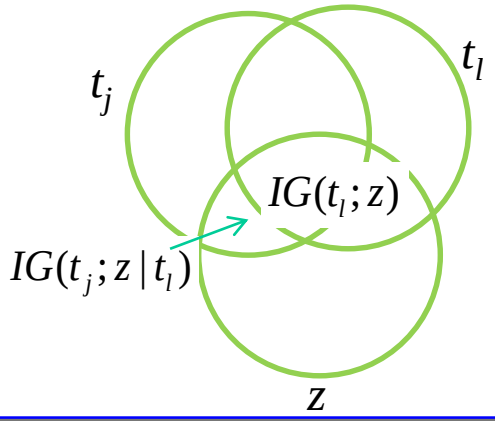
  Joint Mutual Information (JMI) Criterion: (works well)

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

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

- Conditional Mutual Information Maximization (CMIM) Criterion:

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



How to Compute JMI?

Binary Test Case:

Example: Diagnosis of single faults or normal (no-fault) state

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

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

$$so, P(t_j = 1) = q_j = \sum_{i=0}^{C} P(z=i)P(t_j \mid z=i) = \sum_{i=0}^{C} P_i \mu_{ij}; P(t_j = 0) = 1 - q_j$$

$$IG(t_j; z) = H(t_j) - H(t_j \mid z) = -[q_j \log_2 q_j + (1-q_j)\log_2(1-q_j)]$$
$$+ \sum_{i=0}^{C} P_i[\mu_{ij} \log_2 \mu_{ij} + (1-\mu_{ij})\log_2(1-\mu_{ij})]$$

JMI Computation in the Binary Case

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

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

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

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

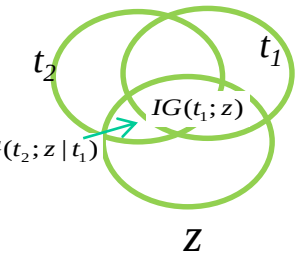
Example of Computing *JMI*

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
- $P(t_1 = T) = 0.4;\ P(t_2 = T) = 0.7$
- $P(t_1 = T | z = +) = 0.8$
- $P(t_1 = T | z = -) = 0$

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



$$H(z,t_1,t_2) = -\{(1/2)(2/5)(4/5)\log_2[(1/2)(2/5)(4/5)] + (1/2)(3/5)(4/5)\log_2[(1/2)(3/5)(4/5)]$$
$$+ (1/2)(2/5)(1/5)\log_2[(1/2)(2/5)(1/5)] + (1/2)(3/5)(1/5)\log_2[(1/2)(3/5)(1/5)]$$
$$+ (1/2)\log_2[1/2]\} = 1.846439$$

$$H(t_1,t_2) = -0.06\log_2[0.06] - 0.24\log_2[0.24] - 0.54\log_2[0.54] - 0.16\log_2[0.16] = 1.640728$$

$$H(z,t_1) = -\{(1/2)(4/5)\log_2[(1/2)(4/5)] + (1/2)(1/5)\log_2[(1/2)(1/5)] + (1/2)\log_2[1/2]\} = 1.360964$$

$$H(t_1) = -\{(2/5)\log_2[2/5] + (3/5)\log_2[3/5]\} = 0.9709506$$

$$\Rightarrow IG(t_2; z \mid t_1) = H(z,t_1) - H(t_1) - H(z,t_1,t_2) + H(t_1,t_2) = 0.1843023$$

$$H(z,t_2) = -\{(1/2)(2/5)\log_2[(1/2)(2/5)] + (1/2)(3/5)\log_2[(1/2)(3/5)] + (1/2)\log_2[1/2]\} = 1.485475$$

$$H(t_2) = -\{(7/10)\log_2[7/10] + (3/10)\log_2[3/10]\} = 0.8812909$$

$$\Rightarrow IG(t_1; z \mid t_2) = H(z,t_2) - H(t_2) - H(z,t_1,t_2) + H(t_1,t_2) = 0.3984732$$

$$Note: IG(t_2; z \mid t_1) = H(t_2 \mid t_1) - H(t_2 \mid t_1, z) = H(z \mid t_1) - H(z \mid t_1, t_2)$$

$\Rightarrow t_1$ is a better test

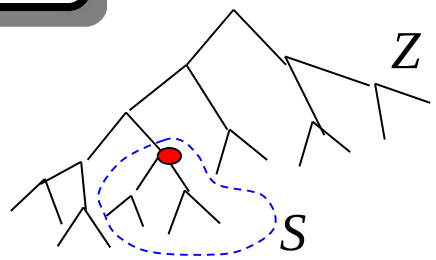
Other Criteria

- **Other Measures**:
  - G-statistic (related to mutual information)
  - Half-split partitioning (“two-ing” rule)
  - Linear discriminant splits

Decision Trees Exhibit Overfitting for Noisy Data

- **Reason**: Decision Trees Overfit (low bias and high variance)
  - Real-life data is noisy
    - Continuous attributes have measurement noise
    - Discrete attributes such as color depend on subjective interpretation
    - Instances are misclassified, mistakes are made in recording…
  - Noisy data leads to larger trees
    - Could 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



- **Cost-Complexity Pruning**
  - Suppose have a tree Z
  - Let $E(Z)$ be the number of classes misclassified out of N *(cost)*
  - Let $L(Z)$ be the number of leaves of Z *(complexity of tree)*
  - Cost complexity of Z is: $\frac{E(Z)}{N} + \alpha L(Z)$
- 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 (MAP decision)
  - We now have a tree $\hat{Z}$ with $L(S) - 1$ fewer leaves
  - That is, $L(\hat{Z}) = L(Z) - [L(S) - 1]$

Pruning Criterion

- Suppose the new **error rate** is: $\frac{E(\hat{Z})}{N}$
  - Prune S if

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

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

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

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

Pruning Algorithm

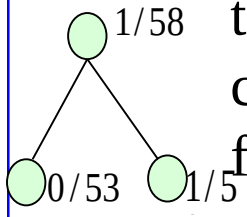
- **Algorithm**
  - Set $k = 0, \quad Z_0 = Z$
  - ♣ Set $\alpha = \infty$
  - Visit non terminal nodes t in bottom-up order and calculate $E(Z_t)$ and $L(Z_t)$ by summing over the descendants

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

  - 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

- **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 ⇒ 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.
  - 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.

CART: Classification and Regression Trees

- Classification and regression trees (**CART**) is a non-parametric decision tree learning technique that produces either classification or regression trees, depending on whether the dependent variable is categorical or numeric, respectively
- **Strengths**:
  - Trees for numeric data
  - Automatic feature selection
  - No model in advance
  - May work better than traditional regression
  - Does not require knowledge of statistics to interpret the results
- **Weaknesses**:
  - Requires large amount of training data
  - Effect of the predictors/features may not be as easy to interpret as a traditional regression model

Splitting Criteria for Regression

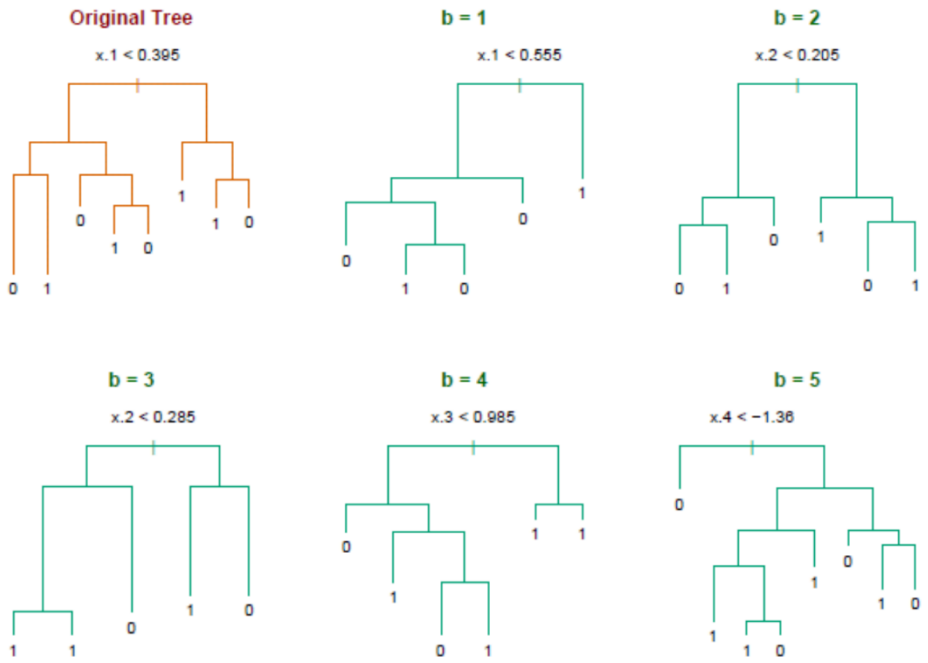
- **Splitting Criteria**: *Mean Square Error (MSE)*
  - Unlike linear models, decision tree regression is able to capture *non-linear interaction between the features and the target*
  - Good at handling tabular data with numerical features, or categorical features not designed to be very sparse
  - $MSE(X_m) = \frac{1}{N_m}\sum_{i=1}^{N_m}(Y_i - \bar{Y}_m)^2$, for *node m*

Standard Deviation Reduction Criterion

- Partitioning is done using a **divide-and-conquer** strategy according to the feature that will result in the greatest increase in homogeneity in the outcome after a split is preformed
- The algorithm can be used to construct a decision tree for regression by replacing Information Gain with *Standard Deviation Reduction*
  - Standard Deviation (S) : $S = \sqrt{\sum(x - \bar{x})^2 / n}$
  - Coefficient of Variation (CV) is used to decide when to stop branching: $CV = \frac{S}{\bar{x}} * 100\%$
  - Estimate: Average (Avg) is the value in the leaf nodes
  - Standard deviation reduction (**SDR**) is the decrease in standard deviation after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest standard deviation reduction (i.e., the most homogeneous child nodes)

Bagging

- Bagging: **B**ootstrap **Agg**regat**ing**
- Method: 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?

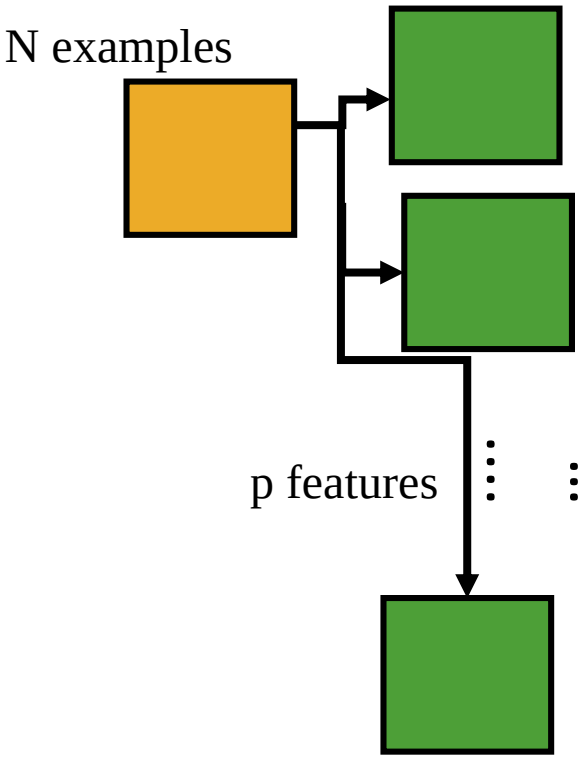
  ⇨ Random Forests

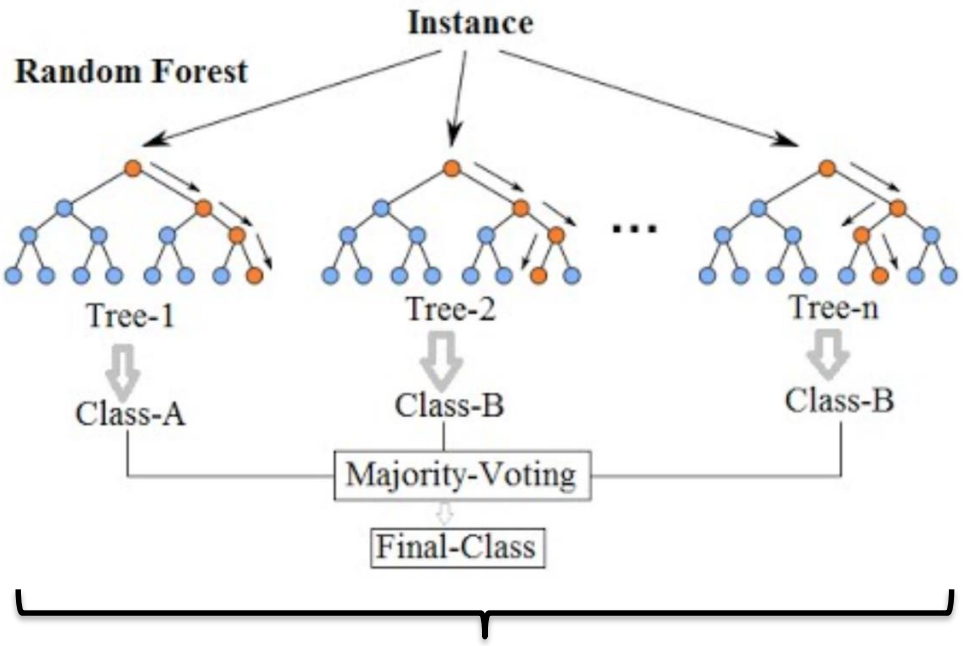
Hastie et al.,"The Elements of Statistical Learning: Data Mining, Inference, and Prediction", Springer (2009)

Random Forest

Create bootstrap samples from training data

For each tree select only a random subset of features





Take the majority vote from all trees

Random Forest Algorithm

- **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.

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

- **To make a prediction at a new point $\underline{x}$ we do:**
  - For regression: average the results
  - For classification: majority vote

Boosting: Weak Learners in Series

- **Goal:** Greedy algorithm that successively applies a weak learning algorithm on *weighted versions of the data* using bounds on the loss function
- **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

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

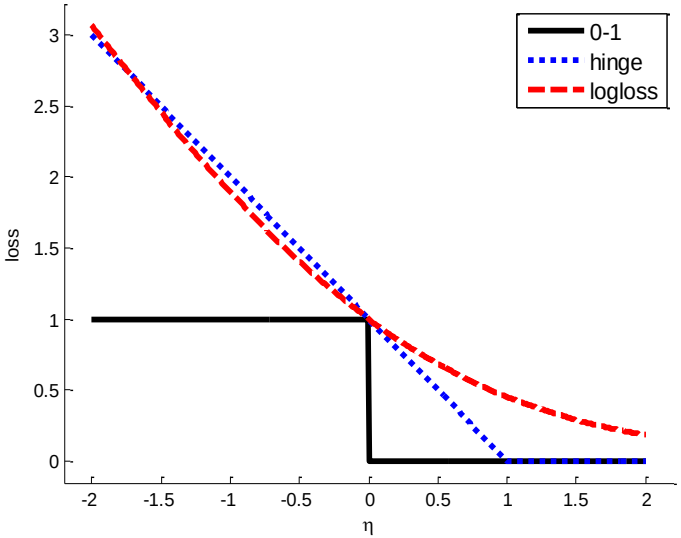
  - Log Loss (Logistic loss)

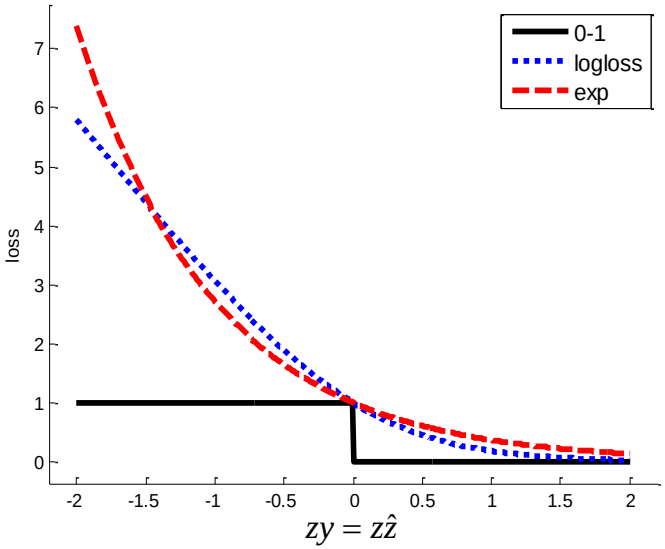
$$L(z,\hat{z}) \le L_l(z,y) = \ln(1+e^{-zy}) / \ln 2$$

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

Illustration of Loss Functions

- **Comparison of the three loss functions:**





AdaBoost Algorithm

- **Principle**
  - Fit a sequence of weak learners (i.e., decision trees) on repeatedly modified versions of the data
  - Combine the predictions from all through a weighted majority vote/sum and produce the final prediction
- **Data Modification**
  - Set all weights to $w_i = 1/N$ initially
  - Sample weights are individually modified during iteration and the learning algorithm is reapplied to the reweighted data
  - Those training examples that were incorrectly predicted by the boosted model have their weights increased, whereas the weights are decreased for those that were predicted correctly
  - Examples that are difficult to predict receive ever-increasing influence. **Each subsequent weak learner is thereby forced to concentrate on the examples that are missed by the previous ones in the sequence**

AdaBoost Weight Update

- 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 $J_m = \sum_{n=1}^{N} w_n^{(m)} (1 - \delta_{z^n f(\underline{x}^n)})$

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

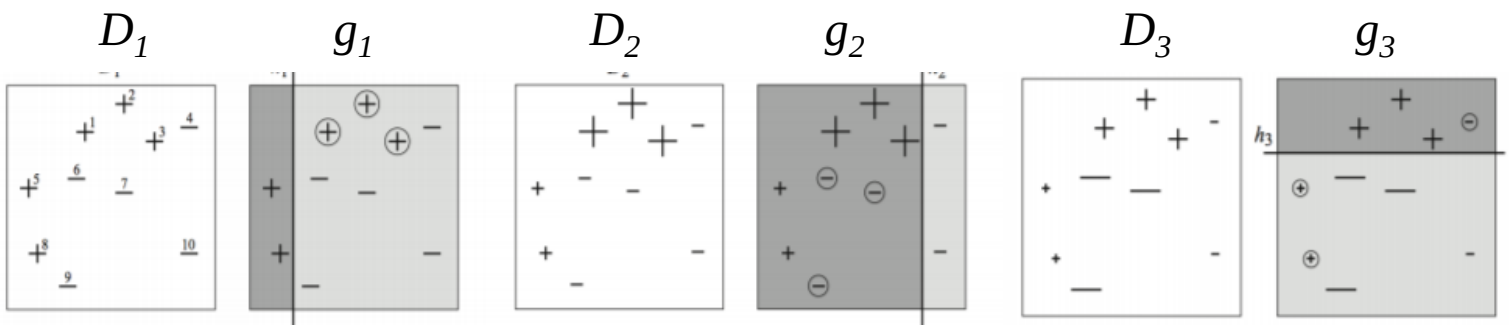
$$= \frac{\sum_{n=1}^{N} w_n^{(m)} \delta_{z^n f(\underline{x}^n)} / \sum_{n=1}^{N} w_n^{(m)}}{\sum_{n=1}^{N} w_n^{(m)} (1 - \delta_{z^n f(\underline{x}^n)}) / \sum_{n=1}^{N} w_n^{(m)}} = \frac{1 - e_m}{e_m} \Rightarrow \alpha_m = \frac{1}{2} \ln\left(\frac{1 - e_m}{e_m}\right)$$

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

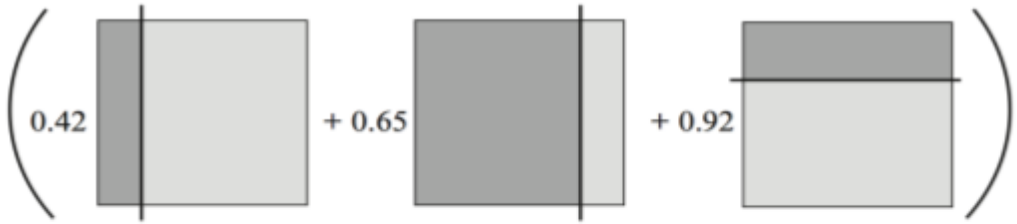
$$\hat{z} = \text{sgn}[\sum_{m=1}^{M} \alpha_m f_m(\underline{x}^n)]; M \approx 10 - 20$$

AdaBoost in Action

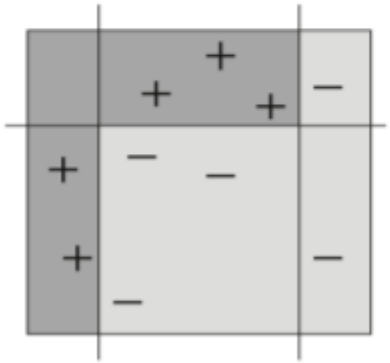
- **AdaBoost in Action**



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



$=$



Gradient Boosting

- **Intuition**
  - Repetitively leverage the patterns in residuals and strengthen a model with weak predictions and make it better
- **Procedure**
  1) Fit a simple linear regressor or decision tree on data
  2) Calculate error residuals: $e_1 = y - y_{predicted1}$
  3) Fit a new model on error residuals e_1 as target variable with the same input variables and obtain $e_{1_{predicted}}$
  4) Add the predicted residuals to the previous prediction $y_{predicted2} = y_{predicted1} + e_{1_{predicted}}$
  5) Fit another model on residuals that are still left. i.e., $e_2 = y - y_{predicted2}$ and repeat steps 2 to 5 until it starts overfitting or the sum of residuals become constant
  6) Overfitting can be controlled by consistently checking accuracy on validation data

Gradient Boosting Theory

- L_2 Boosting for regression (Residual $\hat{z}_m$ = Negative Gradient), so **gradient boosting is similar to iterative improvement of Least Squares**

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

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

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)

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

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

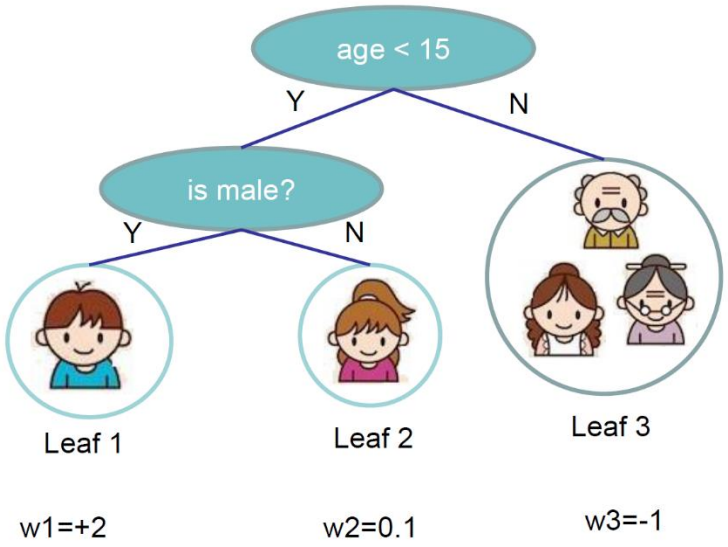
Regularization in Gradient Boosting

- **Complexity of a tree and regularization**

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

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 Details

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

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

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

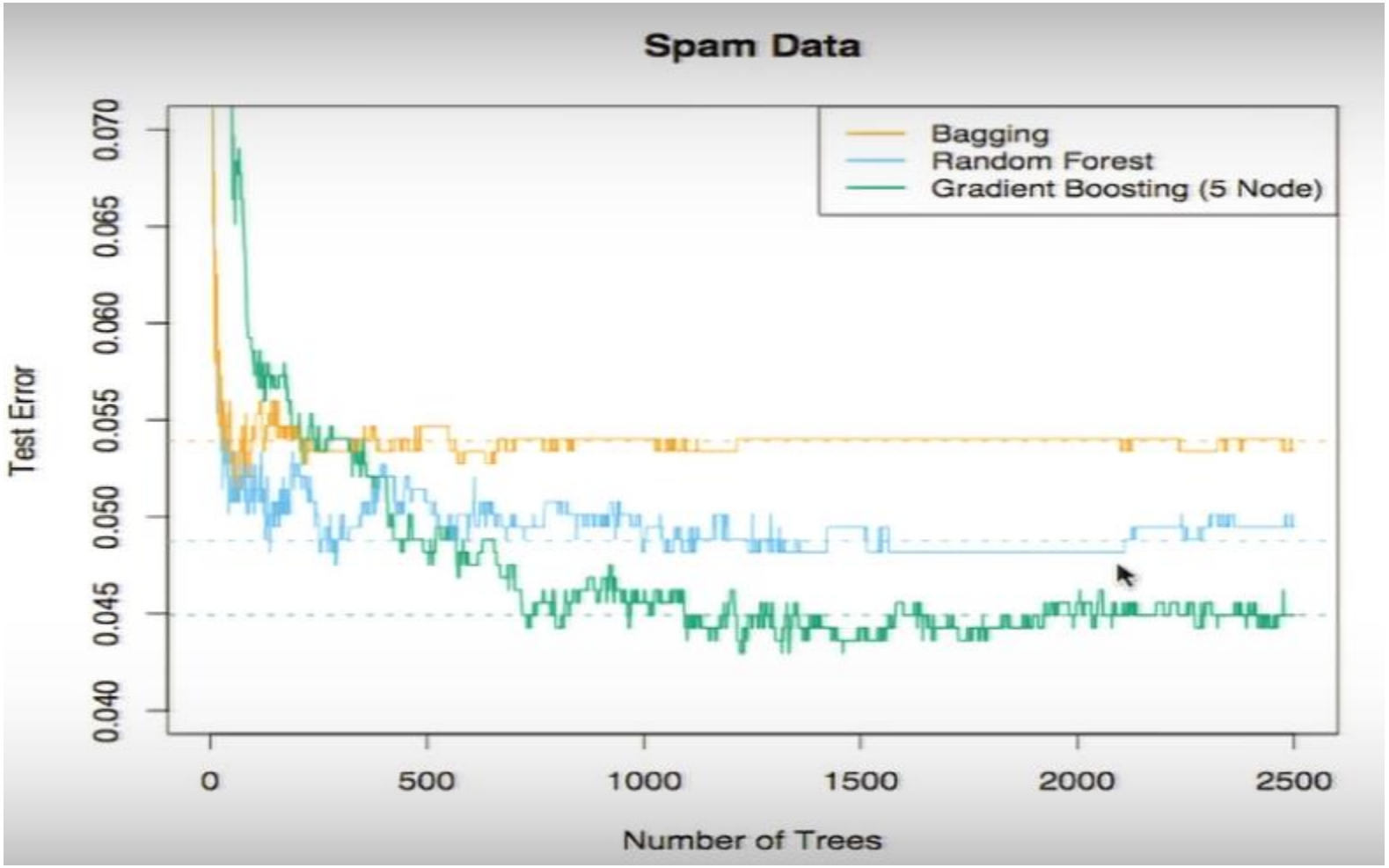
$$\text{For known } \{I_j\}, \textit{optimal } w_j = -\frac{G_j}{H_j + \lambda}; L_m(f_m) = \frac{1}{2} \sum_{j=1}^{T} (\frac{-G_j^2}{H_j + \lambda} + 2\gamma)$$

- 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/

Missing Value Problem

- **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 for which outcomes for some cases are not known
  - When classifying an unseen instance whose outcome for a test is undetermined

Handling Missing Attribute Values

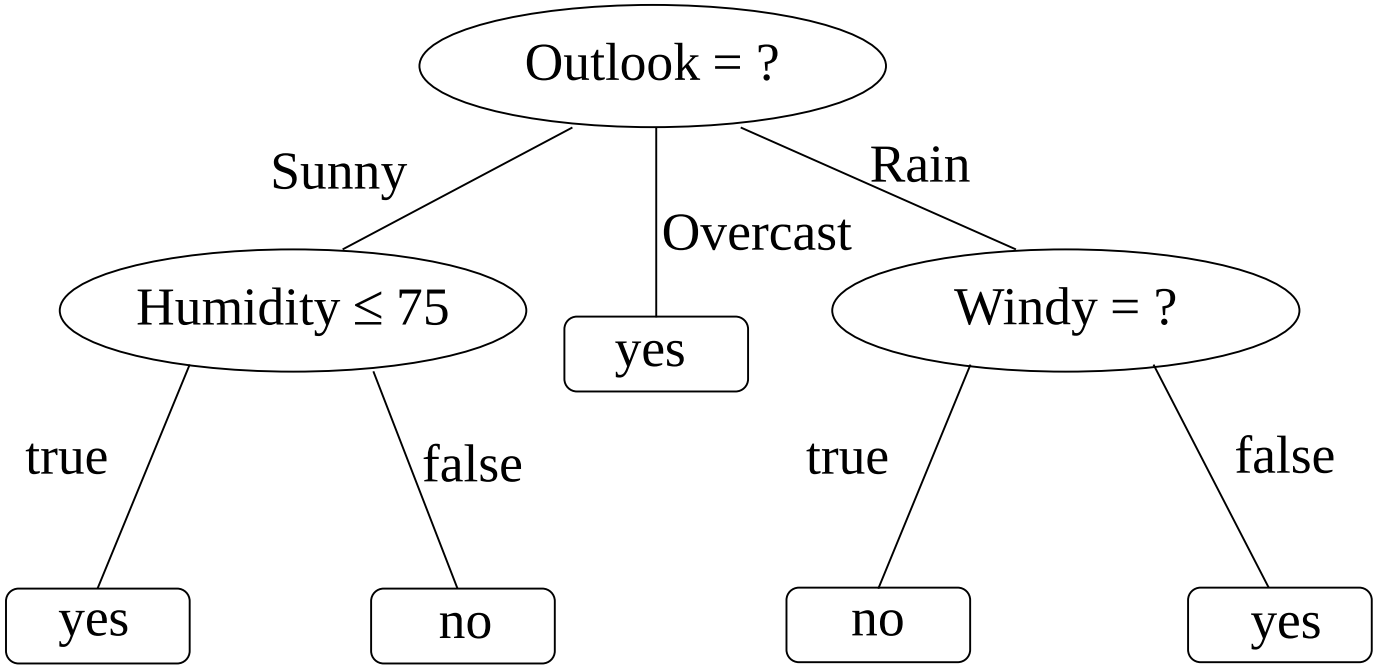
- **How to handle missing values?**
  - Filling in missing values
    - Discrete: pick the most frequent value
    - Continuous: use mean, median,…
  - Estimating test outcomes by some other means
    - Backup (surrogate) tests

    ⇨ 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.

Treat Missing Test Outcomes Probabilistically

- **Treat test outcomes probabilistically**
  - Case: Outlook = ?, Humidity = 85, Windy

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



Total Probability Theorem Comes to the Rescue

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

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

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

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

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

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

$\Rightarrow$ Prediction: no

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 the least Hamming distance
- L_K **- Boosted Tree** (Friedman, 1999)
  - Gradient and diagonal Hessian are easy to compute
  - Gradient boosting algorithm
  - Implementation: https://github.com/tqchen/xgboost

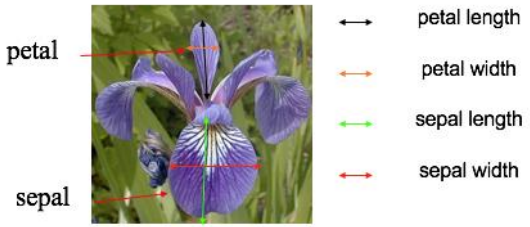
Decision Trees on Iris Dataset

- Dataset: Iris Dataset
- Data Characteristics
  - Dimension: (150, 5)
  - Variable name: sepal_length, sepal_width, petal_length, petal_width, Class
  - Class: Iris-virginica, Iris-setosa, Iris-versicolor
  - Statistics

iris setosa

iris versicolor

iris virginica



Name	sepal_length	sepal_width	petal_length	petal_width
Mean	5.843333	3.054000	3.758667	1.198667
Std	0.828066	0.433594	1.764420	0.763161
Min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
Max	7.900000	4.400000	6.900000	2.500000

Classification Results

- Classification Results (use *10-fold* CV)
  - Decision Tree:
    - Accuracy: 0.95;

    Confusion Matrix: $\begin{bmatrix} 50 & 0 & 0 \\ 0 & 47 & 3 \\ 0 & 4 & 46 \end{bmatrix}$
  - Bagging:
    - Accuracy: 0.95;

    Confusion Matrix: $\begin{bmatrix} 50 & 0 & 0 \\ 0 & 48 & 2 \\ 0 & 5 & 45 \end{bmatrix}$
  - Random Forest:
    - Accuracy: 0.95;

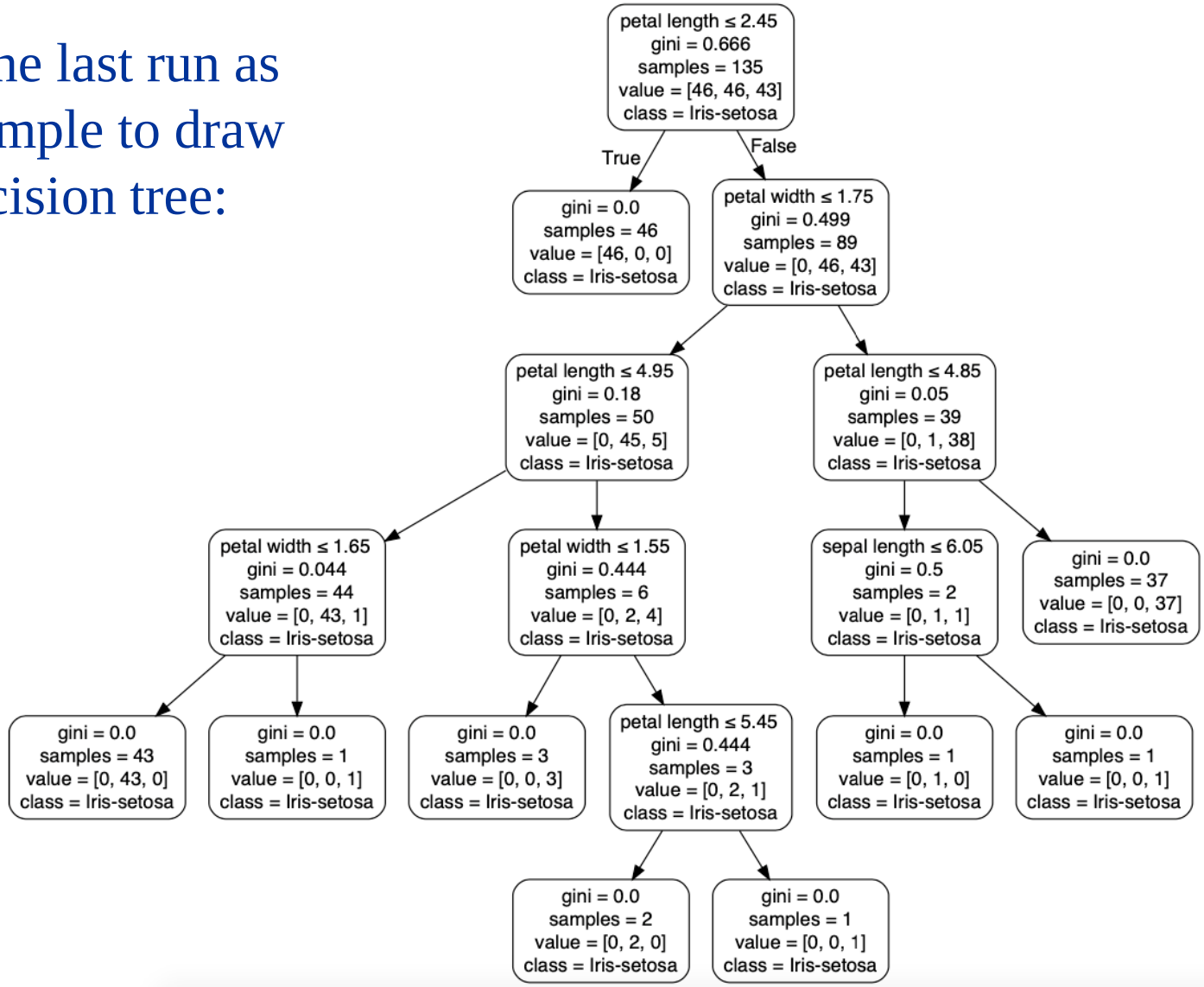
    Confusion Matrix: $\begin{bmatrix} 50 & 0 & 0 \\ 0 & 47 & 3 \\ 0 & 4 & 46 \end{bmatrix}$
  - Adaboost:
    - Accuracy: 0.94;

    Confusion Matrix: $\begin{bmatrix} 50 & 0 & 0 \\ 0 & 48 & 2 \\ 0 & 7 & 43 \end{bmatrix}$
  - Gradient Boosting:
    - Accuracy: 0.95;

    Confusion Matrix: $\begin{bmatrix} 50 & 0 & 0 \\ 0 & 46 & 4 \\ 0 & 3 & 47 \end{bmatrix}$

Illustration of the Decision Tree

- Take the last run as an example to draw the decision tree:



Daily Demand Forecasting

- Data: Daily Demand Forecasting of Orders Data Set
- Data Characteristics
  - Dimension: (60, 13)
  - Variable name:
    - Week of the month, Day of the week, Non-urgent order, Urgent order,
    - Order type A, Order type B, Order type C, Fiscal sector orders,
    - Orders from the traffic controller sector, Banking orders (1),
    - Banking orders (2), Banking orders (3), Target (Total orders)
  - Statistics

	Week	Day	Non-urgent	Urgent	A	B	C	Fiscal	Traffic	(1)	(2)	(3)	Target
Count	60	60	60	60	60	60	60	60	60	60	60	60	60
Mean	3.0167	4.03	172.555	118.920	52.11	109.2	139.5	77.4	44504	46641	79401	23115	300.87
Std	1.2821	1.40	69.5058	27.170	18.83	50.74	41.44	186.5	12198	45220	40504	13148	89.602
Min	1	2	43.651	77.371	21.83	25.12	74.37	0	11992	3452	16411	7679	129.41
25%	2	3	125.348	100.888	39.46	74.91	113.6	1.243	43994	20130	50680	12609	238.19
50%	3	4	151.062	113.114	47.17	99.48	127.9	7.831	44312	32527	67181	18011	288.03
75%	4	5	194.606	132.108	58.46	132.2	160.1	20.36	52111	45118	94787	31047	334.24
Max	5	6	435.304	223.270	118.2	267.3	302.4	865	71772	210508	118411	73839	616.45

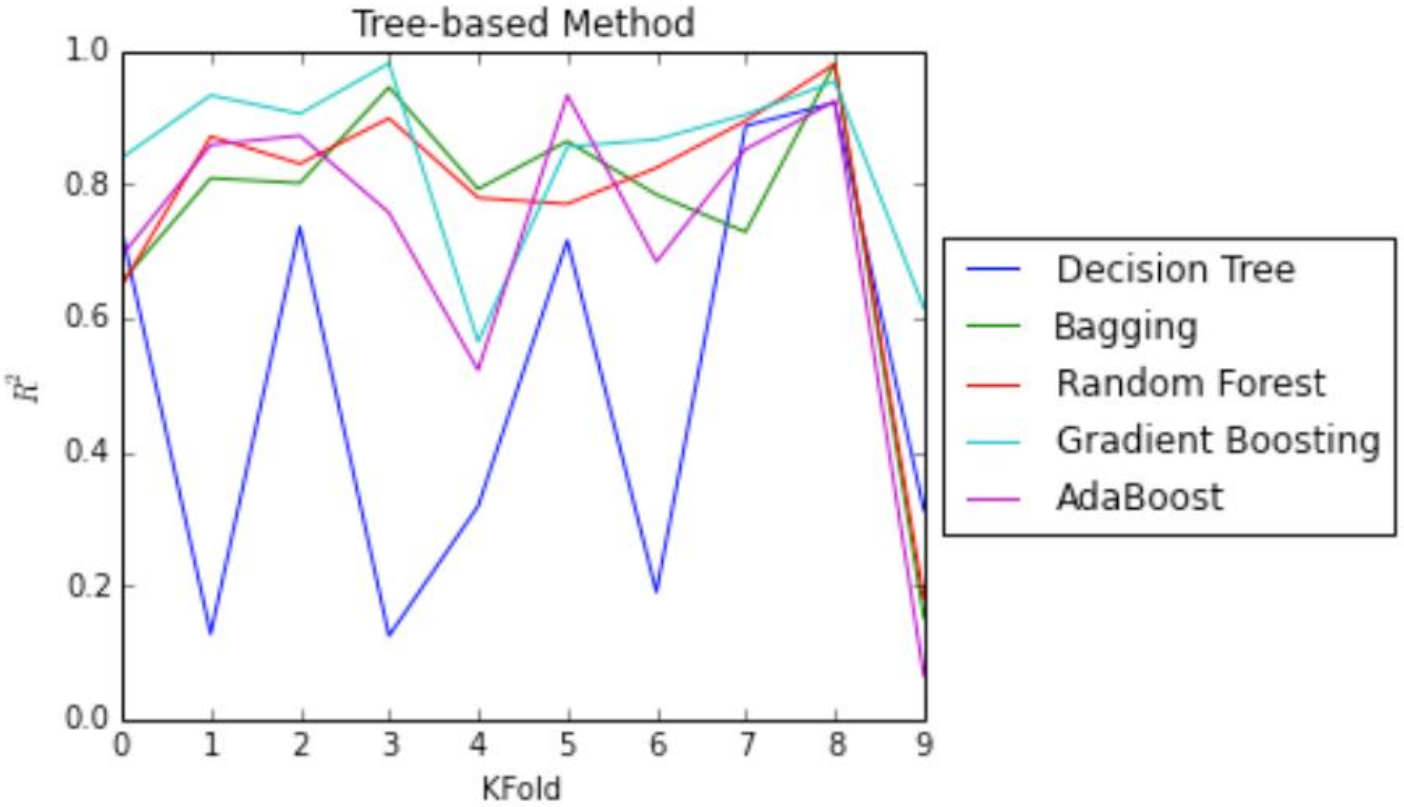
Regression Results

❑ Regression Results (use *10-fold* CV)

	Decision Tree	Bagging	Random Forest	Gradient Boosting	AdaBoost
R^2 For Each of the 10 runs	0.73219263	0.65367567	0.64712644	0.83904701	0.69243168
	0.12742654	0.8089253	0.87127467	0.93275196	0.85845528
	0.73583454	0.80144741	0.83037705	0.90512359	0.872457
	0.1245631	0.94491186	0.89833425	0.98049061	0.75727369
	0.31860439	0.79270759	0.77942496	0.56446882	0.52201362
	0.71630002	0.86412714	0.77022867	0.85565265	0.93315235
	0.19024517	0.784296	0.82433512	0.86657848	0.68397311
	0.88698967	0.72839256	0.8943683	0.90376259	0.85221271
	0.92169198	0.98009775	0.9795034	0.95350312	0.92407035
	0.31264182	0.15138468	0.17993565	0.61506416	0.0651054
MSE	2212.28704	1728.99723	1680.97353	979.44188	1656.35413

R^2 versus Cross-validation Run

- Plot correlation of determination (R^2)



Gradient Boosting (red) is more stable than the rest.
Decision tree (blue) is least stable.

Simple Example where IG is Suboptimal

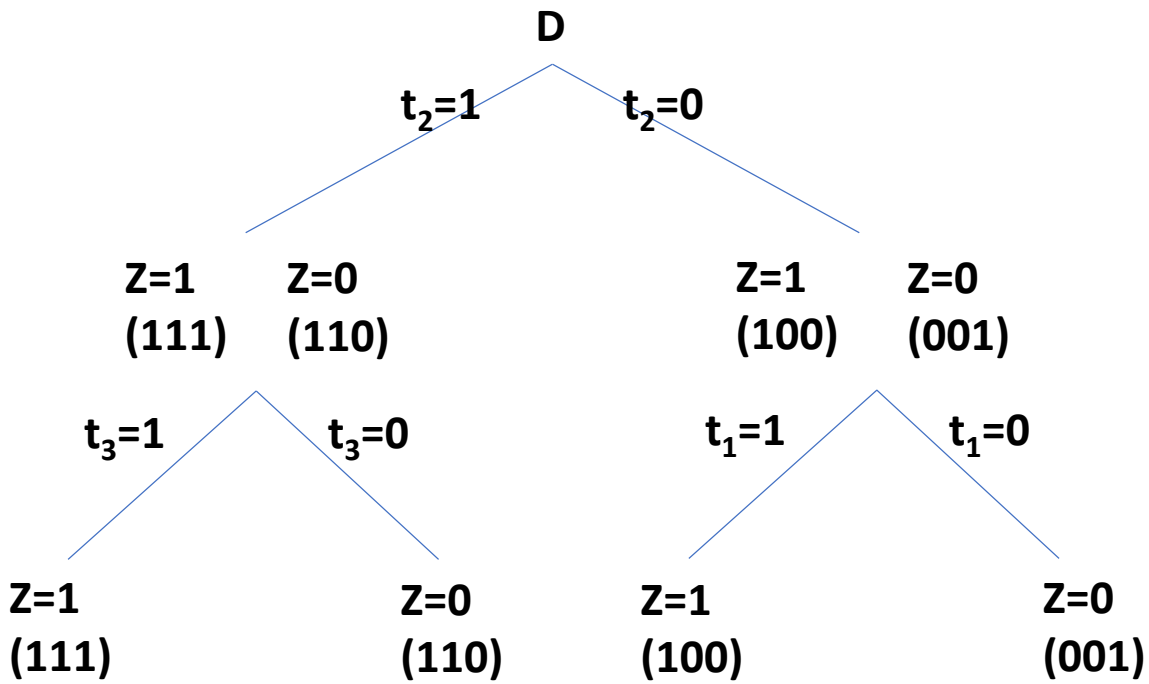
- Consider the training set

z=1	(111)	(100)
z=0	(110)	(001)
	$t_1 t_2 t_3$	$t_1 t_2 t_3$

- Derive a decision tree of depth 2 that attains zero training error
  - Can the information gain algorithm attain zero error with a decision tree of depth 2?
  - What about one with JMI criterion?

Optimal Classification Tree

(1) Decision Tree with a depth of 2



Information Gain Algorithm - 1

(2) Information Gain Algorithm

t_1	z	Count
1	1	2
	0	0
0	1	1
	0	1

Table 1: Attribute of t_1

t_2	z	Count
1	1	1
	0	1
0	1	1
	0	1

Table 2: Attribute of t_2

t_3	z	Count
1	1	1
	0	1
0	1	1
	0	1

Table 3: Attribute of t_3

$$H(z) = 1$$

$$\Rightarrow IG(t_1, z) = 1 - \left[-\frac{3}{4}\left(\frac{2}{3}\log_2 \frac{2}{3} + \frac{1}{3}\log_2 \frac{1}{3} \right) - \frac{1}{4}\left(1\log_2 1\right) \right] = 0.3113$$

$$IG(t_2, z) = 1 - \left[-\frac{1}{2}\left(\frac{1}{2}\log_2 \frac{1}{2} + \frac{1}{2}\log_2 \frac{1}{2} \right) - \frac{1}{2}\left(\frac{1}{2}\log_2 \frac{1}{2} + \frac{1}{2}\log_2 \frac{1}{2} \right) \right] = 0$$

$$IG(t_3, z) = 1 - \left[-\frac{1}{2}\left(\frac{1}{2}\log_2 \frac{1}{2} + \frac{1}{2}\log_2 \frac{1}{2} \right) - \frac{1}{2}\left(\frac{1}{2}\log_2 \frac{1}{2} + \frac{1}{2}\log_2 \frac{1}{2} \right) \right] = 0$$

Thus, we choose t_1

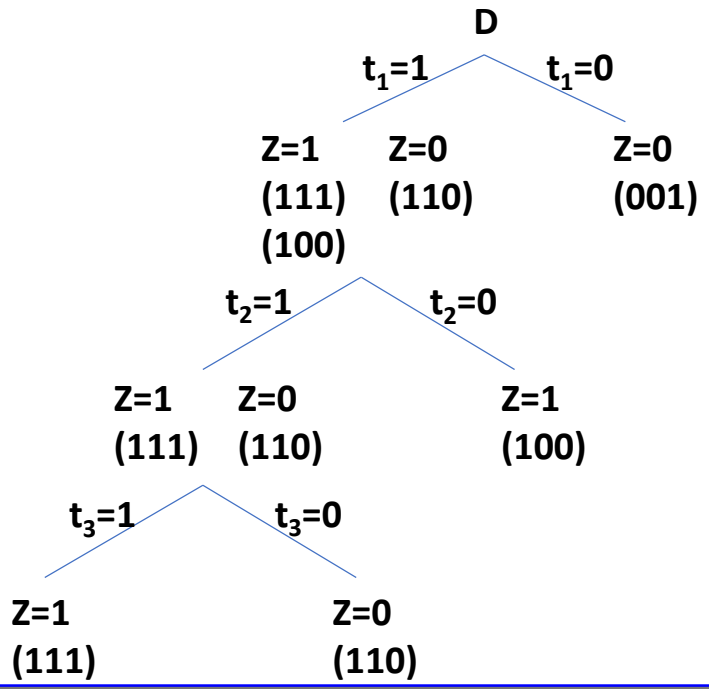
Information Gain Algorithm - 2

$$H(z) = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.9183$$

$$\Rightarrow IG(t_2, z) = 0.9183 - \left[-\frac{2}{3}\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) - \frac{1}{3}\left(1\log_2 1\right)\right] = 0.2516$$

$$IG(t_3, z) = 0.9183 - \left[-\frac{1}{3}\left(1\log_2 1\right) - \frac{2}{3}\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right)\right] = 0.2516$$

- Hard to decided between t_2 or t_3
- Then we can construct a tree as in the figure, which will not lead to a tree of depth 2



JMI Algorithm -1

(3) *JMI* Criterion

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

$$I(t_1; z \mid t_2) = H(z \mid t_2) - H(z \mid t_1, t_2) = 1 - \left(\frac{1}{2} * 1 + \frac{1}{4} * 0 + \frac{1}{4} * 0 \right) = \frac{1}{2}$$

$$I(t_1; z \mid t_3) = H(z \mid t_3) - H(z \mid t_1, t_3) = 1 - \left(\frac{1}{2} * 1 \right) = \frac{1}{2}$$

$$\Rightarrow j_1 = \frac{1}{2} + \frac{1}{2} = 1$$

$$I(t_2; z \mid t_1) = H(z \mid t_1) - H(z \mid t_1, t_2) = 0.6887 - \frac{1}{2} = 0.1887$$

$$I(t_2; z \mid t_3) = H(z \mid t_3) - H(z \mid t_2, t_3) = 1 - 0 = 1$$

$$\Rightarrow j_2 = 0.1887 + 1 = 1.887$$

$$I(t_3; z \mid t_1) = H(z \mid t_1) - H(z \mid t_1, t_3) = 0.6887 - \frac{1}{2} = 0.1887$$

$$I(t_3; z \mid t_2) = H(z \mid t_2) - H(z \mid t_2, t_3) = 1 - 0 = 1$$

$$\Rightarrow j_3 = 0.1887 + 1 = 1.887$$

t_2 and t_3 have same gain.
We choose t_2.

JMI Algorithm -2

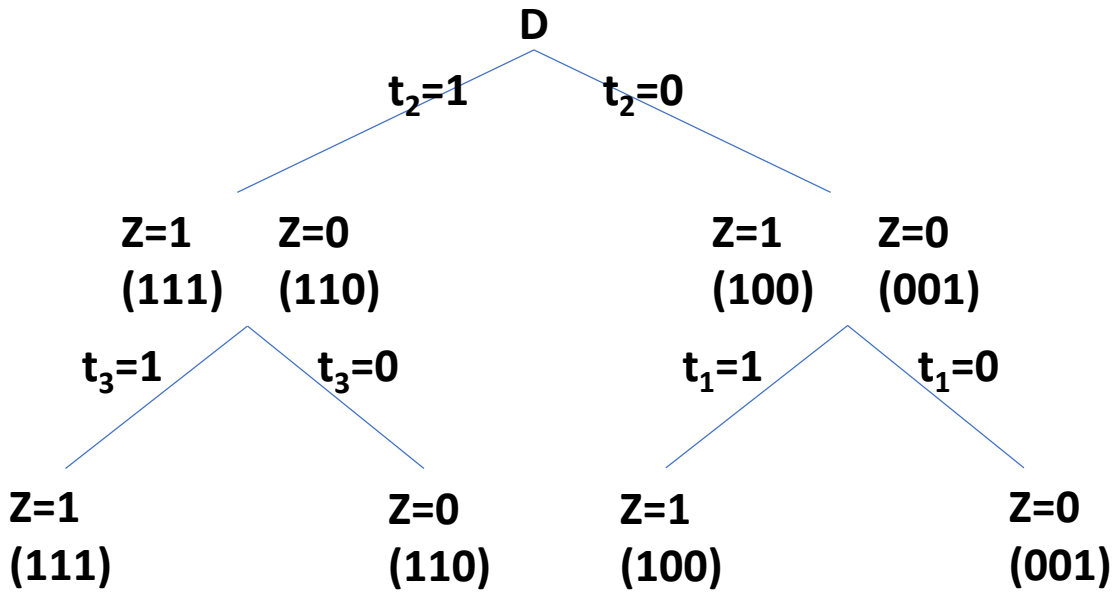
- **When $t_2 = 1$, we have**

$$I(t_1;z|t_2) = H(z|t_2) - H(z|t_1,t_2) = 0$$
$$I(t_1;z|t_3) = H(z|t_3) - H(z|t_1,t_3) = 0$$
$$I(t_3;z|t_1) = H(z|t_1) - H(z|t_1,t_3) = 1$$
$$I(t_3;z|t_2) = H(z|t_2) - H(z|t_2,t_3) = 1$$

- **When $t_2 = 0$, we have**

$$I(t_1;z|t_2) = H(z|t_2) - H(z|t_1,t_2) = 1$$
$$I(t_1;z|t_3) = H(z|t_3) - H(z|t_1,t_3) = 0$$
$$I(t_3;z|t_1) = H(z|t_1) - H(z|t_1,t_3) = 0$$
$$I(t_3;z|t_2) = H(z|t_2) - H(z|t_2,t_3) = 1$$

- **Then, the tree is optimal as shown in the Figure**



Example 2

- Training set

State	t_1	t_2	t_3	t_4	t_5	z	Count
s_0	0	0	0	0	0	0	70
s_1	0	1	0	0	1	1	1
s_2	0	0	1	1	0	2	2
s_3	1	0	0	1	1	3	10
s_4	1	1	0	0	0	1	5
s_5	1	1	1	1	0	3	12

- Build Decision Tree using
  - JMI
  - Information Gain
  - Purity Gain
  - Classification Purity Gain

JMI Algorithm -1

(1) *JMI*

$$j_k = \arg\max_j \sum_{l \in T_k} IG\left(t_j; z \mid t_l\right)$$

Because

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

where

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

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

$$H\left(z, t_l, t_j\right) = -\left\{ P_i \mu_{il} \mu_{ij} \log_2 \left[P_i \mu_{il} \mu_{ij} \right] + P_i \left(1 - \mu_{il}\right) \mu_{ij} \log_2 \left[P_i \left(1 - \mu_{il}\right) \mu_{ij} \right] + \mathrm{P}_i \mu_{il} \left(1 - \mu_{ij}\right) \log_2 \left[P_i \mu_{il} \left(1 - \mu_{ij}\right) \right] \right.$$
$$\left. + P_i \left(1 - \mu_{il}\right) \left(1 - \mu_{ij}\right) \log_2 \left[P_i \left(1 - \mu_{il}\right) \left(1 - \mu_{ij}\right) \right] \right\}$$

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

$$\text{where } q_{lj}^{(00)} = \sum_{i=0}^{C} P_i \left(1 - \mu_{il}\right) \left(1 - \mu_{ij}\right); \quad q_{lj}^{(10)} = \sum_{i=0}^{C} \mathrm{P}_i \mu_{il} \left(1 - \mu_{ij}\right)$$

$$q_{lj}^{(01)} = \sum_{i=0}^{C} P_i \left(1 - \mu_{il}\right) \mu_{ij}; \qquad q_{lj}^{(11)} = \sum_{i=0}^{c} P_i \mu_{il} \mu_{ij}$$

JMI Algorithm -2

❑ Root

Prior: $P(z=0)=0.7;\ \ P(z=1)=0.06;\ \ P(z=2)=0.02;\ \ P(z=3)=0.22$

From $P(t_j \mid z=i)=\begin{cases}1-\mu_{ij} & \text{for } t_j=0\\ \mu_{ij} & \text{for } t_j=1\end{cases};\quad j=1,2,3,4,5;\quad i=0,1,2,3$

$$\Rightarrow \begin{cases} P(t_1=1\mid z=0)=\mu_{01}=\dfrac{P(t_1=1,z=0)}{P(z=0)}=0; & P(t_1=1\mid z=1)=\mu_{11}=\dfrac{P(t_1=1,z=1)}{P(z=1)}=\dfrac{5}{6} \\ P(t_1=1\mid z=2)=\mu_{21}=\dfrac{P(t_1=1,z=2)}{P(z=2)}=0; & P(t_1=1\mid z=3)=\mu_{31}=\dfrac{P(t_1=1,z=3)}{P(z=3)}=1 \end{cases}$$

$$\begin{cases} P(t_2=1\mid z=0)=\mu_{02}=0 \\ P(t_2=1\mid z=1)=\mu_{12}=1 \\ P(t_2=1\mid z=2)=\mu_{22}=0 \\ P(t_2=1\mid z=3)=\mu_{32}=\dfrac{6}{11} \end{cases} \quad \begin{cases} P(t_3=1\mid z=0)=\mu_{03}=0 \\ P(t_3=1\mid z=1)=\mu_{13}=0 \\ P(t_3=1\mid z=2)=\mu_{23}=1 \\ P(t_3=1\mid z=3)=\mu_{33}=\dfrac{6}{11} \end{cases} \quad \begin{cases} P(t_4=1\mid z=0)=\mu_{04}=0 \\ P(t_4=1\mid z=1)=\mu_{14}=0 \\ P(t_4=1\mid z=2)=\mu_{24}=1 \\ P(t_4=1\mid z=3)=\mu_{34}=1 \end{cases} \quad \begin{cases} P(t_5=1\mid z=0)=\mu_{05}=0 \\ P(t_5=1\mid z=1)=\mu_{15}=\dfrac{1}{6} \\ P(t_5=1\mid z=2)=\mu_{25}=0 \\ P(t_5=1\mid z=3)=\mu_{35}=\dfrac{5}{11} \end{cases}$$

Then

$H(t_1)=0.8414646;\quad H(t_2)=0.6800770;\quad H(t_3)=0.5842388;\quad H(t_4)=0.7950403;\quad H(t_5)=0.4999160$

$H(z,t_1)=1.236187;\quad H(z,t_2)=1.415872;\quad H(z,t_3)=1.415872;\quad H(z,t_4)=1.197185;\quad H(z,t_5)=1.454873$

JMI Algorithm -3

$H(z,t_1,t_2)=1.454873;\quad H(z,t_1,t_3)=1.454873;\quad H(z,t_1,t_4)=1.236187;\quad H(z,t_1,t_5)=1.493875$
$H(z,t_2,t_3)=1.634559;\quad H(z,t_2,t_4)=1.415872;\quad H(z,t_2,t_5)=1.673560$
$H(z,t_3,t_4)=1.415872;\quad H(z,t_3,t_5)=1.673560$
$H(z,t_4,t_5)=1.673560$

$H(t_1,t_2)=1.174449;\quad H(t_1,t_3)=1.241307;\quad H(t_1,t_4)=1.160364;\quad H(t_1,t_5)=1.120841$
$H(t_2,t_3)=1.210683;\quad H(t_2,t_4)=1.337869;\quad H(t_2,t_5)=1.102960$
$H(t_3,t_4)=1.030209;\quad H(t_3,t_5)=1.015963$
$H(t_4,t_5)=1.10702$

Thus, according to the formula of information gain, we have

$$k_1 = IG(t_1;z\,|\,t_2)+IG(t_1;z\,|\,t_3)+IG(t_1;z\,|\,t_4)+IG(t_1;z\,|\,t_5)=1.9816837$$
$$k_2 = IG(t_2;z\,|\,t_1)+IG(t_2;z\,|\,t_3)+IG(t_2;z\,|\,t_4)+IG(t_2;z\,|\,t_5)=1.2305553$$
$$k_3 = IG(t_3;z\,|\,t_1)+IG(t_3;z\,|\,t_2)+IG(t_3;z\,|\,t_4)+IG(t_3;z\,|\,t_5)=0.8069176$$
$$k_4 = IG(t_4;z\,|\,t_1)+IG(t_4;z\,|\,t_2)+IG(t_4;z\,|\,t_3)+IG(t_4;z\,|\,t_5)=2.0297657$$
$$k_5 = IG(t_5;z\,|\,t_1)+IG(t_5;z\,|\,t_2)+IG(t_5;z\,|\,t_3)+IG(t_5;z\,|\,t_4)=0.4152118$$

From the result, t_4 is the maximum. So first step, we choose t_4.

JMI Algorithm - 4

❑ Left → When $t_4 = 0$

State	t_1	t_2	t_3	t_5	z	Count
s_0	0	0	0	0	0	70
s_1	0	1	0	1	1	1
s_4	1	1	0	0	1	5

Prior : $P(z=0) = 70/76 = 0.921;\quad P(z=1) = 6/76 = 0.079$ while all μ keep same

$H(t_1) = 0.3500106;\quad H(t_2) = 0.3984593;\quad H(t_3) = 0;\quad H(t_5) = 0.1010670$

$H(z,t_1) = 0.4497768;\quad H(z,t_2) = 0.3984593;\quad H(z,t_3) = 0.3984593;\quad H(z,t_5) = 0.4497768$

$H(z,t_1,t_2) = 0.4497768;\quad H(z,t_1,t_3) = 0.4497768;\quad H(z,t_1,t_5) = 0.5010944$

$H(z,t_2,t_3) = 0.3984593;\quad H(z,t_2,t_5) = 0.4497768;\quad H(z,t_3,t_5) = 0.4497768$

$H(t_1,t_2) = 0.4497768;\quad H(t_1,t_3) = 0.3500106;\quad H(t_1,t_5) = 0.4150904$

$H(t_2,t_3) = 0.3984593;\quad H(t_2,t_5) = 0.4497768\quad H(t_3,t_5) = 0.101067$

Under t_4=0, t_2 is the maximum. Thus, we choose t_2, which separate class 0 and

$$k_1 = IG(t_1;z\,|\,t_2) + IG(t_1;z\,|\,t_3) + IG(t_1;z\,|\,t_5) = 0.56$$

$$k_2 = IG(t_2;z\,|\,t_1) + IG(t_2;z\,|\,t_3) + IG(t_2;z\,|\,t_5) = 0.84$$

$$k_3 = IG(t_3;z\,|\,t_1) + IG(t_3;z\,|\,t_2) + IG(t_3;z\,|\,t_5) = 1.39\times10^{-17}$$

$$k_5 = IG(t_5;z\,|\,t_1) + IG(t_3;z\,|\,t_2) + IG(t_5;z\,|\,t_3) = 0.064$$

$t_2 = 0 \rightarrow$

State	t_1	t_3	t_5	z	Count
s_0	0	0	0	0	70

$t_2 = 1 \rightarrow$

State	t_1	t_3	t_5	z	Count
s_1	0	0	1	1	1
s_4	1	0	0	1	5

JMI Algorithm -5

❑ Right → When $t_4 = 1$

State	t_1	t_2	t_3	t_5	z	Count
s_2	0	0	1	0	2	2
s_3	1	0	0	1	3	10
s_5	1	1	1	0	3	12

Prior : $P(z=2) = 2/14 = 0.083$; $P(z=3) = 22/24 = 0.917$ while all μ keep same

$H(t_1) = 0.4138169$; $H(t_2) = 1$; $H(t_3) = 0.9798688$; $H(t_5) = 0.9798688$

$H(z,t_1) = 0.4138169$; $H(z,t_2) = 1.3250112$; $H(z,t_3) = 1.3250112$; $H(z,t_5) = 1.3250112$

$H(z,t_1,t_2) = 1.325011$; $H(z,t_1,t_3) = 1.325011$; $H(z,t_1,t_5) = 1.3250114$

$H(z,t_2,t_3) = 2.236206$; $H(z,t_2,t_5) = 2.236206$; $H(z,t_3,t_5) = 2.236206$

$H(t_1,t_2) = 1.325011$; $H(t_1,t_3) = 1.325011$; $H(t_1,t_5) = 1.325011$

$H(t_2,t_3) = 1.975606$; $H(t_2,t_5) = 1.975606$; $H(t_3,t_5) = 1.956701$

Under t_4=1, t_1 is the maximum. Thus, we choose t_1, which separate class 2 and

$$k_1 = IG(t_1;z|t_2) + IG(t_1;z|t_3) + IG(t_1;z|t_5) = 1.0152961$$

$$k_2 = IG(t_2;z|t_1) + IG(t_2;z|t_3) + IG(t_2;z|t_5) = 0.1690854$$

$$k_3 = IG(t_3;z|t_1) + IG(t_3;z|t_2) + IG(t_3;z|t_5) = 0.1300492$$

$$k_5 = IG(t_5;z|t_1) + IG(t_3;z|t_2) + IG(t_5;z|t_3) = 0.1300492$$

$t_1 = 0 \rightarrow$

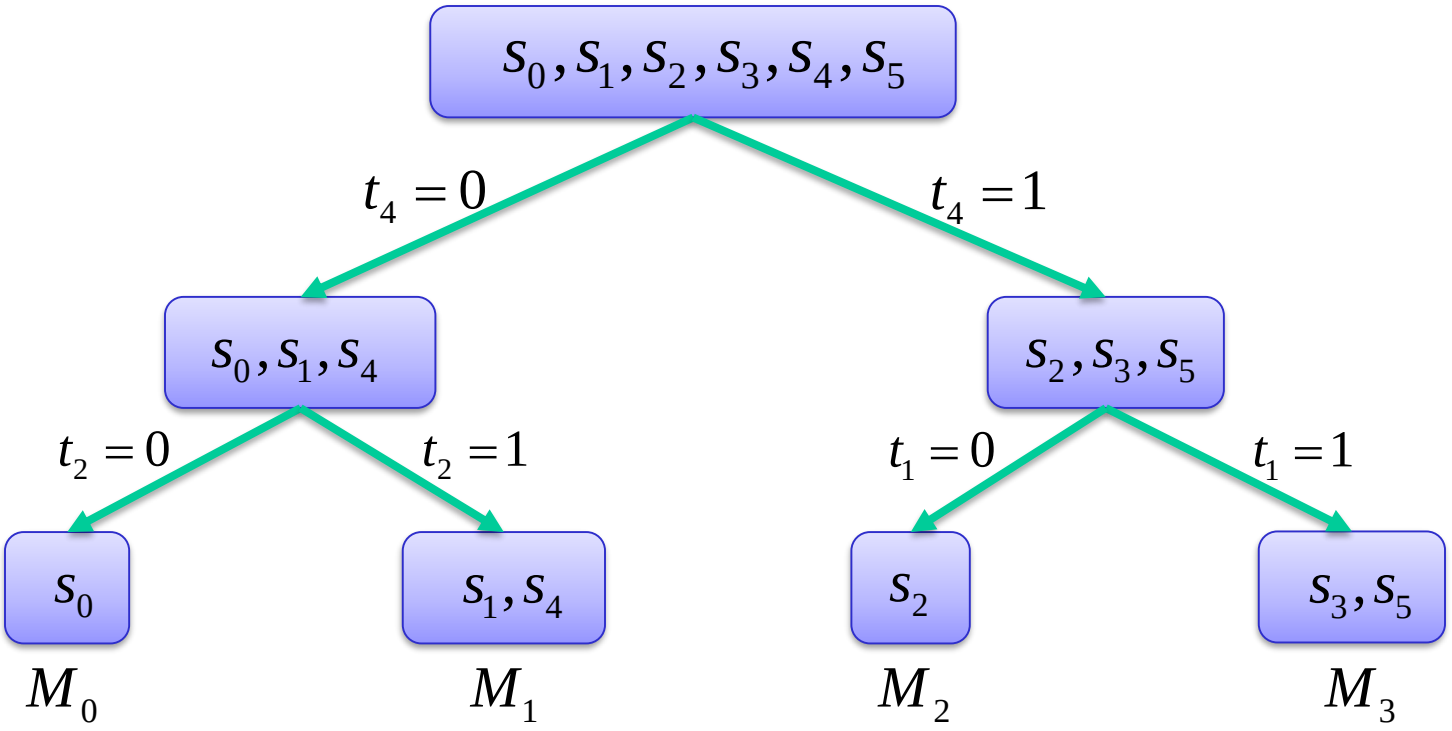
State	t_2	t_3	t_5	z	Count
s_2	1	1	0	1	2

$t_1 = 1 \rightarrow$

State	t_2	t_3	t_5	z	Count
s_3	0	0	1	3	10
s_5	1	1	0	3	12

Classification Tree with JMI

- Then the tree is as shown in the Figure (Optimal)



IG Algorithm - 1

(2) Information Gain

$$IG(t_j;z) = H(t_j) - H(t_j \mid z) = H(z) - H(z \mid t_j)$$

where

$$H(t_j) = -\left[q_j \log_2 q_j + (1-q_j)\log_2 (1-q_j) \right] \quad \text{where } q_j = \sum_{i=0}^{c} P_i \mu_{ij}$$

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

- Root

From calculation in JMI, we have

$H(t_1) = 0.8414646;\quad H(t_2) = 0.6800770;\quad H(t_3) = 0.5842388;\quad H(t_4) = 0.7950403;\quad H(t_5) = 0.4999160$

Also we can calculate

$H(t_1 \mid z) = 0.03900135;\quad H(t_2 \mid z) = 0.2186866;\quad H(t_3 \mid z) = 0.2186866$

$H(t_4 \mid z) = 0;\qquad H(t_5 \mid z) = 0.257688$

Thus

$IG(t_1,z) = H(t_1) - H(t_1 \mid z) = 0.8024633;\quad IG(t_2,z) = H(t_2) - H(t_2 \mid z) = 0.4613904$

$IG(t_3,z) = H(t_3) - H(t_3 \mid z) = 0.3655522;\quad IG(t_4,z) = H(t_4) - H(t_4 \mid z) = 0.7950403$

$IG(t_5,z) = H(t_5) - H(t_5 \mid z) = 0.2422280$

$IG(t_1,z)$ is the maximum. Choose t_1

IG Algorithm - 2

- Left → When $t_1 = 0$

State	t_2	t_3	t_4	t_5	z	Count
s_0	0	0	0	0	0	70
s_1	1	0	0	1	1	1
s_2	0	1	1	0	2	2

Prior : $P(z=0)=\frac{70}{73}; \quad P(z=1)=\frac{1}{73}; \quad P(z=2)=\frac{2}{73}$

$H(t_2)=0.10441908; \quad H(t_3)=0.18116640; \quad H(t_4)=0.18116640; \quad H(t_5)=0.02332382$

$H(t_2 \mid z)=0; \quad H(t_3 \mid z)=0; \quad H(t_4 \mid z)=0; \quad H(t_5 \mid z)=0.008904417$

Then the Information Gain when $t_1 = 0$ is

$IG(t_2,z)=0.10441908; \quad IG(t_3,z)=0.18116640; \quad IG(t_4,z)=0.18116640; \quad IG(t_5,z)=0.01441941$

$t_3 = t_4$ and they are the maximum. Choose t_3. Continue,

$P(z=0)=\frac{70}{71}; \quad P(z=1)=\frac{1}{71}$

$H(t_2)=0.10679203; \quad H(t_4)=0; \quad H(t_5)=0.02388664$

$H(t_2 \mid z)=0; \quad H(t_4 \mid z)=0; \quad H(t_5 \mid z)=0.009155245$

$IG(t_2,z)=0.1067920; \quad IG(t_4,z)=0; \quad IG(t_5,z)=0.0147314$

Choose t_2 that separate class 0 and class 1

$t_3 = 0$

State	t_2	t_4	t_5	z	Count
s_0	0	0	0	0	70
s_1	1	0	1	1	1

$t_3 = 1$

State	t_2	t_4	t_5	z	Count
s_2	0	1	0	2	2

IG Algorithm 3

- **Right → When $t_1 = 1$**

State	t_2	t_3	t_4	t_5	z	Count
s_3	0	0	1	1	2	10
s_4	1	0	0	0	1	5
s_5	1	1	1	0	3	12

$$P(z=1)=\frac{5}{27};\quad P(z=3)=\frac{22}{27}$$

$$H(t_2)=0.9509560;\quad H(t_2)=0.9910761;$$
$$H(t_4)=0.6912899;\quad H(t_5)=0.9716682$$

$$H(t_2\mid z)=0.8099505;\quad H(t_3\mid z)=0.8099505;$$
$$H(t_4\mid z)=0;\quad H(t_5\mid z)=0.9303251$$

$$IG(t_2,z)=0.14100551;\quad IG(t_3,z)=0.18112552;$$
$$IG(t_4,z)=0.69128987;\quad IG(t_5,z)=0.04134313$$

t_4 is the maximum, it separate class 1 and class 3. Choose t_4.

$t_4 = 1$

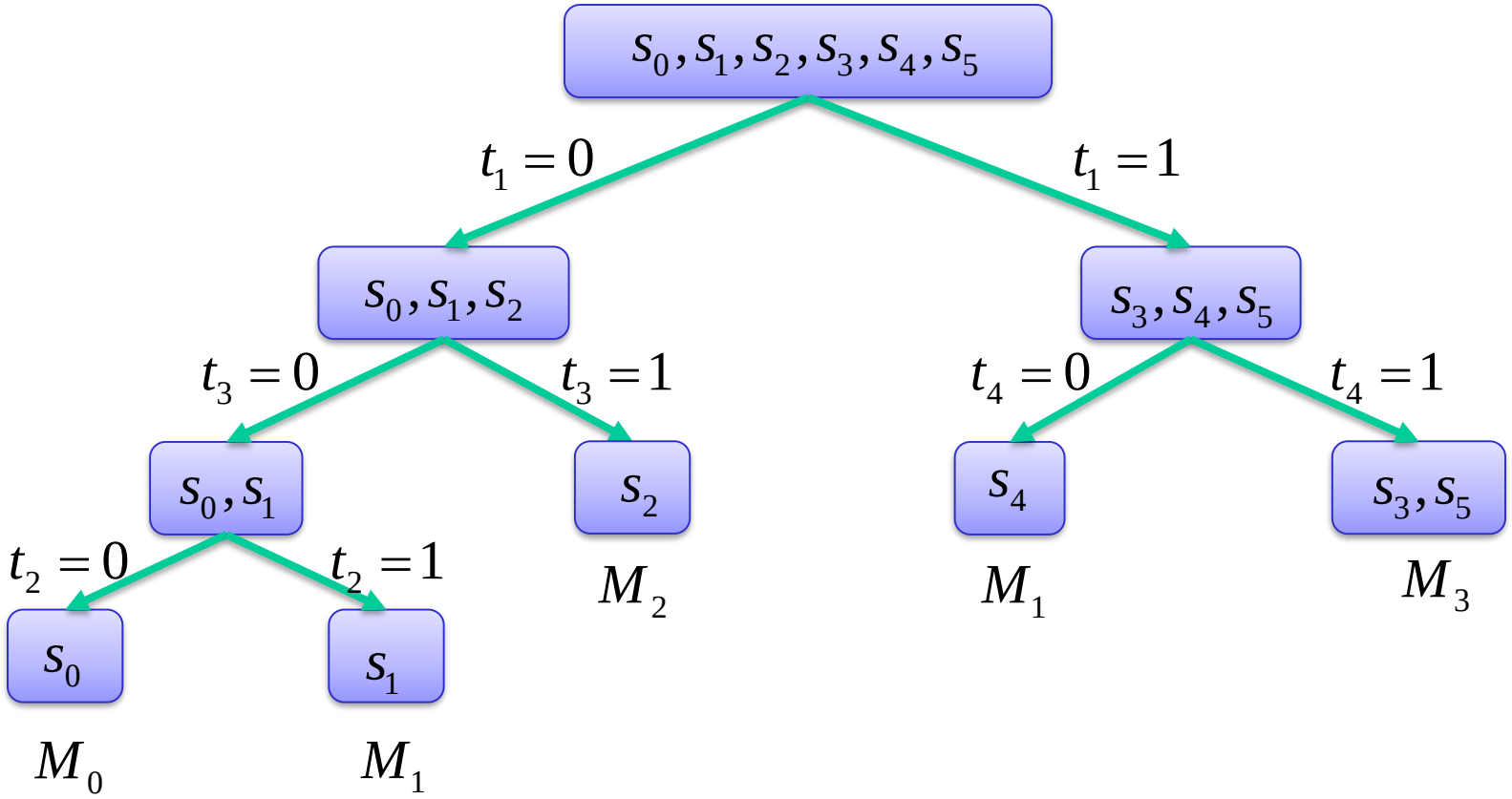
State	t_2	t_3	t_5	z	Count
s_3	0	0	1	3	10
s_5	1	1	0	3	12

$t_4 = 0$

State	t_2	t_3	t_5	z	Count
s_4	1	0	0	1	5

Classification Tree with IG

- Then, the IG tree is as shown in the Figure (Not optimal)



Purity Gain Algorithm - 1

(3) Purity Gain

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

$$\text{where} \quad G(D) = 1 - \sum_{j=1}^{C} P_j^2 = \sum_{j=1}^{C} P_j - \sum_{j=1}^{C} P_j^2 = \sum_{j=1}^{C} P_j (1 - P_j)$$

- Root

$P(z=0) = 0.7; \quad P(z=1) = 0.06; \quad P(z=2) = 0.02; \quad P(z=3) = 0.22$

$G(D) = 0.7(1-0.7) + 0.06(1-0.06) + 0.02(1-0.02) + 0.22(1-0.22) = 0.4576$

$$G(D\,|\,t_1) = \frac{73}{100} * \left[1 - \left(\frac{70}{73}\right)^2 - \left(\frac{1}{73}\right)^2 - \left(\frac{2}{73}\right)^2\right] + \frac{27}{100} * \left[1 - \left(\frac{10}{27}\right)^2 - \left(\frac{5}{27}\right)^2 - \left(\frac{12}{27}\right)^2\right] = 0.2284 \Rightarrow GP(D\,|\,t_1) = 0.2291$$

$$G(D\,|\,t_2) = \frac{82}{100} * \left[1 - \left(\frac{70}{82}\right)^2 - \left(\frac{2}{82}\right)^2 - \left(\frac{10}{82}\right)^2\right] + \frac{18}{100} * \left[1 - \left(\frac{1}{18}\right)^2 - \left(\frac{5}{18}\right)^2 - \left(\frac{12}{18}\right)^2\right] = 0.2953 \Rightarrow GP(D\,|\,t_1) = 0.1622$$

$$G(D\,|\,t_3) = \frac{86}{100} * \left[1 - \left(\frac{70}{86}\right)^2 - \left(\frac{1}{86}\right)^2 - \left(\frac{10}{86}\right)^2 - \left(\frac{5}{86}\right)^2\right] + \frac{14}{100} * \left[1 - \left(\frac{2}{14}\right)^2 - \left(\frac{12}{14}\right)^2\right] = 0.3098 \Rightarrow GP(D\,|\,t_1) = 0.1477$$

$$G(D\,|\,t_4) = \frac{76}{100} * \left[1 - \left(\frac{70}{76}\right)^2 - \left(\frac{1}{76}\right)^2 - \left(\frac{5}{76}\right)^2\right] + \frac{24}{100} * \left[1 - \left(\frac{2}{24}\right)^2 - \left(\frac{10}{24}\right)^2 - \left(\frac{12}{24}\right)^2\right] = 0.2485 \Rightarrow GP(D\,|\,t_1) = 0.2091$$

$$G(D\,|\,t_5) = \frac{89}{100} * \left[1 - \left(\frac{70}{89}\right)^2 - \left(\frac{2}{89}\right)^2 - \left(\frac{5}{89}\right)^2 - \left(\frac{12}{89}\right)^2\right] + \frac{11}{100} * \left[1 - \left(\frac{1}{11}\right)^2 - \left(\frac{10}{11}\right)^2\right] = 0.3381 \Rightarrow GP(D\,|\,t_1) = 0.1194$$

$GP(D\,|\,t_1)$ is the maximum. Choose t_1

Purity Gain Algorithm - 2

- Left → When $t_1 = 0$

$$\text{Prior}: P(z=0) = \frac{70}{73}; \quad P(z=1) = \frac{1}{73}; \quad P(z=2) = \frac{2}{73}$$

$$G(D) = 1 - \left(\frac{70}{73}\right)^2 - \left(\frac{1}{73}\right)^2 - \left(\frac{2}{73}\right)^2 = 0.07956465$$

$$G(D \mid t_2) = \left(\frac{72}{73}\right) * \left[1 - \left(\frac{70}{72}\right)^2 - \left(\frac{2}{73}\right)^2\right] + \left(\frac{1}{73}\right) * \left[1 - \left(\frac{1}{1}\right)^2\right] = 0.05327245$$

$$G(D \mid t_3) = \left(\frac{71}{73}\right) * \left[1 - \left(\frac{70}{71}\right)^2 - \left(\frac{1}{71}\right)^2\right] + \left(\frac{2}{73}\right) * \left[1 - \left(\frac{2}{2}\right)^2\right] = 0.02701138$$

$$G(D \mid t_4) = \left(\frac{71}{73}\right) * \left[1 - \left(\frac{70}{71}\right)^2 - \left(\frac{1}{71}\right)^2\right] + \left(\frac{2}{73}\right) * \left[1 - \left(\frac{2}{2}\right)^2\right] = 0.02701138$$

$$G(D \mid t_5) = \left(\frac{72}{73}\right) * \left[1 - \left(\frac{70}{72}\right)^2 - \left(\frac{2}{73}\right)^2\right] + \left(\frac{1}{73}\right) * \left[1 - \left(\frac{1}{1}\right)^2\right] = 0.05327245$$

$$GP(D \mid t_2) = 0.02629220; \quad GP(D \mid t_3) = 0.05255326$$

$$GP(D \mid t_4) = 0.05255326; \quad GP(D \mid t_5) = 0.02629220$$

$t_3 = t_4$ and they are the maximum. Choose t_3

Purity Gain Algorithm - 3

- **Right → When $t_1 = 1$**

$$P(z=1)=\frac{5}{27};\quad P(z=3)=\frac{22}{27}$$

$$G(D)=1-\left(\frac{5}{27}\right)^2-\left(\frac{22}{27}\right)^2=0.3017833$$

$$G(D\,|\,t_2)=0.2614379;\quad G(D\,|\,t_3)=0.2469136;\quad G(D\,|\,t_4)=0.4040404;\quad G(D\,|\,t_5)=0.2614379$$

$$GP(D\,|\,t_2)=0.04034536;\quad GP(D\,|\,t_3)=0.05486968$$

$$GP(D\,|\,t_4)=-0.10225714;\quad GP(D\,|\,t_5)=0.04034536$$

t_3 is the maximum Choose t_3.

Classification Purity Gain Algorithm - 1

(4) Classification Purity Gain

$$GCP(D\,|\,T) = M(D) - \sum_{i=1}^{m} \frac{|D_i|}{|D|} M\left(D_i\right) = M(D) - M(D\,|\,T)$$

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

- Root

$$P(z=0) = 0.7; \quad P(z=1) = 0.06; \quad P(z=2) = 0.02; \quad P(z=3) = 0.22$$

$$M(D) = 1 - \max(0.7, 0.06, 0.02, 0.22) = 0.3$$

$$M(D\,|\,t_1) = \frac{73}{100} * \left[1 - \max\left(\frac{70}{73}, \frac{1}{73}, \frac{2}{73}\right)\right] + \frac{27}{100} * \left[1 - \max\left(\frac{22}{27}, \frac{5}{27}\right)\right] = 0.08 \Rightarrow GCP(D\,|\,t_1) = 0.22$$

$$M(D\,|\,t_2) = \frac{82}{100} * \left[1 - \max\left(\frac{70}{82}, \frac{2}{82}, \frac{10}{82}\right)\right] + \frac{18}{100} * \left[1 - \max\left(\frac{6}{18}, \frac{12}{18}\right)\right] = 0.18 \Rightarrow GCP(D\,|\,t_2) = 0.12$$

$$M(D\,|\,t_3) = \frac{86}{100} * \left[1 - \max\left(\frac{70}{86}, \frac{6}{86}, \frac{10}{86}\right)\right] + \frac{14}{100} * \left[1 - \max\left(\frac{2}{14}, \frac{12}{14}\right)\right] = 0.18 \Rightarrow GCP(D\,|\,t_3) = 0.12$$

$$M(D\,|\,t_4) = \frac{76}{100} * \left[1 - \max\left(\frac{70}{76}, \frac{6}{76}\right)\right] + \frac{24}{100} * \left[1 - \max\left(\frac{2}{24}, \frac{22}{24}\right)\right] = 0.08 \Rightarrow GCP(D\,|\,t_4) = 0.22$$

$$M(D\,|\,t_5) = \frac{89}{100} * \left[1 - \max\left(\frac{70}{89}, \frac{2}{89}, \frac{5}{89}, \frac{12}{89}\right)\right] + \frac{11}{100} * \left[1 - \max\left(\frac{1}{11}, \frac{10}{11}\right)\right] = 0.2 \Rightarrow GCP(D\,|\,t_5) = 0.1$$

$t_1 = t_4$ is the maximum. Choose t_1

Classification Purity Gain Algorithm - 2

❑ Left → When $t_1 = 0$

$P(z=0) = \frac{70}{73}$; $P(z=0) = \frac{1}{73}$; $P(z=0) = \frac{2}{73}$

$M(D) = 0.04109589$

$M(D|t_2) = 0.02739726$; $M(D|t_3) = 0.01369863$; $M(D|t_4) = 0.01369863$; $M(D|t_5) = 0.02739726$

$GCP(D|t_2) = 0.01369863$; $GCP(D|t_3) = 0.02739726$; $GCP(D|t_4) = 0.02739726$; $GCP(D|t_5) = 0.01369863$

$t_3 = t_4$ and are the maximum. Choose t_3

- **Right → When $t_1 = 1$**

$P(z=1) = \frac{5}{27}$; $P(z=3) = \frac{22}{27}$

$M(D) = 0.1851852$

$M(D|t_2) = 0.1851852$; $M(D|t_3) = 0.1851852$; $M(D|t_4) = 0$; $M(D|t_5) = 0.3376906$

$GCP(D|t_2) = 0$; $GCP(D|t_3) = 0$; $GCP(D|t_4) = 0.1851852$; $GCP(D|t_5) = -0.1525054$

t_4 is the maximum. Choose t_4

This gives the same tree as the information gain criterion.

Regression Example

Outlook	Temperature	Humidity	Windy	Hours Played
Rainy	Hot	High	FALSE	26
Rainy	Hot	High	TRUE	30
Overcast	Hot	High	FALSE	48
Sunny	Mild	High	FALSE	46
Sunny	Cool	Normal	FALSE	62
Sunny	Cool	Normal	TRUE	23
Overcast	Cool	Normal	TRUE	43
Rainy	Mild	High	FALSE	36
Rainy	Cool	Normal	FALSE	38
Sunny	Mild	Normal	FALSE	48
Rainy	Mild	Normal	TRUE	48
Overcast	Mild	High	TRUE	62
Overcast	Hot	Normal	FALSE	44
Sunny	Mild	High	TRUE	30

Predictors: Outlook, Temperature, Humidity, Windy
Target: Hours Played

Build a Regression Tree

Weighted Regression

- **Standard Deviation calculation**

(a) Some features for *target* variable:

$$n = 14 \qquad \bar{x} = \frac{\sum x}{n} = 39.8$$

$$S = \sqrt{\frac{\sum (x - \bar{x})^2}{n}} = 9.32 \qquad CV = \frac{S}{\bar{x}} * 100\% = 23\%$$

(b) For two attributes (*Target & Predictor*)

$$S(T, X) = \sum_{c \in X} P(c) S(c)$$

		Hours Played (StDev)	Count
Outlook	Overcast	3.49	4
	Rainy	7.78	5
	Sunny	10.87	5
			14

$$S(Hours, Outlook) = P(Sunny) * S(Sunny) + P(Overcast) * S(Overcast) + P(Rainy) S(Rainy)$$

$$= \frac{4}{14} * 3.49 + \frac{5}{14} * 7.78 + \frac{5}{14} * 10.87 = 7.66$$

SD Reduction-based Regression - 1

Standard Deviation Reduction calculation

Step 1: The standard deviation of the target is calculated.

$$S(H) = 9.32$$

Step 2: The dataset is then split on the different attributes. The resulting standard deviation is subtracted from the standard deviation before the split. The result is the standard deviation reduction.

$$SDR(T, X) = S(T) - S(T, X)$$

$$\Rightarrow SDR(Hours, Outlook) = S(Hours) - S(Hours, Outlook) = 9.32 - 7.66 = 1.66$$

		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
	SDR = 1.66	

		Hours Played (StDev)
Temp	Cool	10.54
	Hot	8.95
	Mild	7.65
	SDR = 0.17	

		Hours Played (StDev)
Humidity	High	9.36
	Normal	8.37
	SDR = 0.28	

		Hours Played (StDev)
Windy	False	7.87
	True	10.59
	SDR = 0.29	

The attribute with the largest standard deviation reduction is chosen for the decision node, which is Outlook.

SD Reduction-based Regression - 2

Step 3: The dataset is divided based on the values of the selected attribute. This process is run recursively on the non-leaf branches, until all data is processed.

Outlook	Temp	Humidity	Windy	Hours Played
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30
Overcast	Hot	High	FALSE	46
Overcast	Cool	Normal	TRUE	43
Overcast	Mild	High	TRUE	52
Overcast	Hot	Normal	FALSE	44
Rainy	Hot	High	FALSE	25
Rainy	Hot	High	TRUE	30
Rainy	Mild	High	FALSE	35
Rainy	Cool	Normal	FALSE	38
Rainy	Mild	Normal	TRUE	48

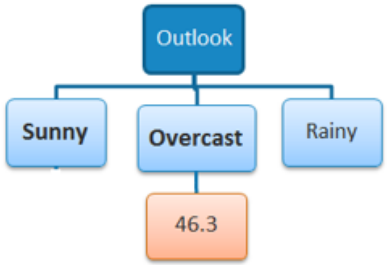
In practice, we need some termination criteria. For example, when coefficient of variation(CV) for a branch becomes smaller than a certain threshold (e.g., 10%) and/or when too few instances (*n*) remain in the branch (e.g., 3).

SD Reduction-based Regression - 3

Step 4: "Overcast" subset does not need any further splitting because its CV (8%) is less than the threshold (10%). The related leaf node gets the average of the "Overcast" subset.

Outlook - Overcast

		Hours Played (StDev)	Hours Played (AVG)	Hours Played (CV)	Count
Outlook	Overcast	3.49	46.3	8%	4
	Rainy	7.78	35.2	22%	5
	Sunny	10.87	39.2	28%	5



However, the "Sunny" branch has a CV (28%), which is more than the threshold (10%) and is further split. We select "Windy" as the best node after "Outlook" because it has the largest SDR.

Outlook - Sunny

Temp	Humidity	Windy	Hours Played
Mild	High	FALSE	45
Cool	Normal	FALSE	52
Cool	Normal	TRUE	23
Mild	Normal	FALSE	46
Mild	High	TRUE	30
			S = 10.87
			AVG = 39.2
			CV = 28%

		Hours Played (StDev)	Count
Temp	Cool	14.50	2
	Mild	7.32	3

SDR = 10.87-((2/5)*14.5 + (3/5)*7.32) = **0.678**

		Hours Played (StDev)	Count
Humidity	High	7.50	2
	Normal	12.50	3

SDR = 10.87-((2/5)*7.5 + (3/5)*12.5) = **0.370**

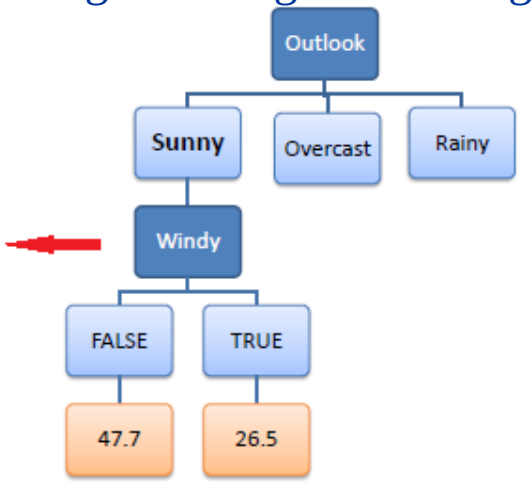
		Hours Played (StDev)	Count
Windy	False	3.09	3
	True	3.50	2

SDR = 10.87-((3/5)*3.09 + (2/5)*3.5) = **7.62**

SD Reduction-based Regression - 4

Because the number of data points for both branches (FALSE and TRUE) is equal to or less than 3, we stop further branching and assign the average of each branch to the related leaf node.

Temp	Humidity	Windy	Hours Played
Mild	High	FALSE	45
Cool	Normal	FALSE	52
Mild	Normal	FALSE	46
Cool	Normal	TRUE	23
Mild	High	TRUE	30



Outlook - Rainy

Temp	Humidity	Windy	Hours Played
Hot	High	FALSE	25
Hot	High	TRUE	30
Mild	High	FALSE	35
Cool	Normal	FALSE	38
Mild	Normal	TRUE	48
			S = 7.78
			AVG = 35.2
			CV = 22%

		Hours Played (StDev)	Count
Temp	Cool	0	1
	Hot	2.5	2
	Mild	6.5	2

SDR = 7.78 - ((1/5)*0+(2/5)*2.5 + (2/5)*6.5) = **4.18**

		Hours Played (StDev)	Count
Humidity	High	4.1	3
	Normal	5.0	2

SDR = 7.78 - ((3/5)*4.1 + (2/5)*5.0) = **3.32**

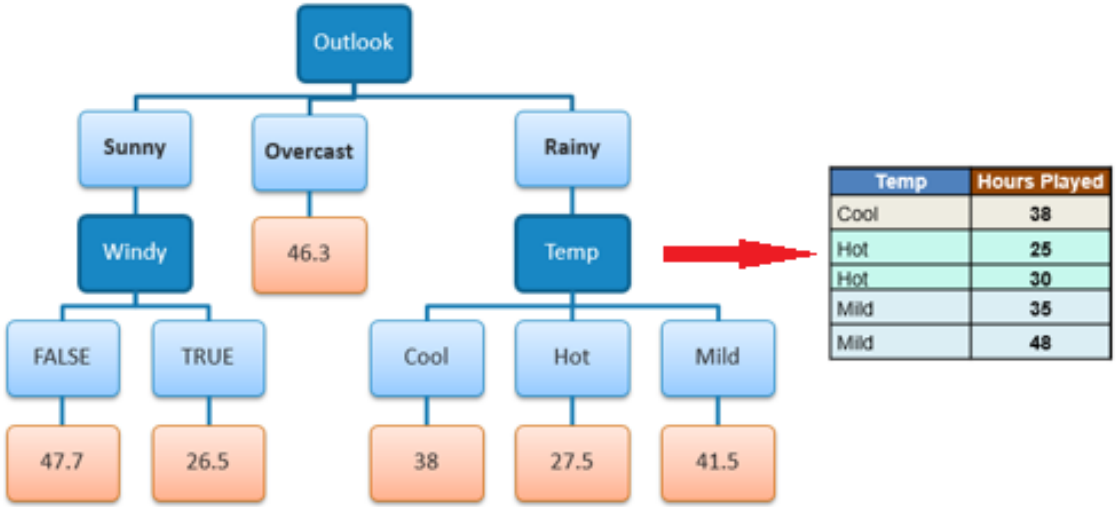
		Hours Played (StDev)	Count
Windy	False	5.6	3
	True	9.0	2

SDR = 7.78 - ((3/5)*5.6 + (2/5)*9.0) = **0.82**

The "Rainy" branch has a CV (22%), which is more than the threshold (10%). This branch needs further splitting. We select "Windy" as the best node because it has the largest SDR.

SD Reduction-based Regression - 4

Because the number of data points for all three branches (Cool, Hot and Mild) is equal to or less than 3, we stop further branching and assign the average of each branch to the related leaf node.



Temp	Hours Played
Cool	38
Hot	25
Hot	30
Mild	35
Mild	48

When the number of instances is more than one at a leaf node, we calculate the average as the final value for the target

Summary

- 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
  - Constructing decision trees, Choosing Tests, Splitting Rules, Pruning Rules, Handling Missing Values, Extensions to Complex Tests
  - Bagging Decision Trees
  - Random Forest
  - Boosting: AdaBoost; Gradient Boosting