10701 Recitation: Decision Trees & Model Selection (AIC & BIC)

Min Chi

Oct 7^{th}, 2010

NSH 1507

More on Decision Trees

Building More General Decision Trees

- Build a decision tree ($\geq$ 2 level) Step by Step.

- Building a decision tree with continuous input feature.

- Building a quad decision tree.

Building More General Decision Trees

- Build a decision tree (≥ 2 level) Step by Step.

Next

- Building a decision tree with continuous input feature.
- Building a quad decision tree.

Information Gain

- Advantage of attribute – decrease in uncertainty
  - Entropy of Y before split

$$H(Y) = -\sum_y P(Y = y) \log_2 P(Y = y)$$

  - Entropy of Y after splitting based on X_i
    - Weight by probability of following each branch

$$\begin{aligned} H(Y \mid X_i) &= \sum_x P(X_i = x) H(Y \mid X_i = x) \\ &= -\sum_x P(X_i = x) \sum_y P(Y = y \mid X_i = x) \log_2 P(Y = y \mid X_i = x) \end{aligned}$$

- Information gain is difference

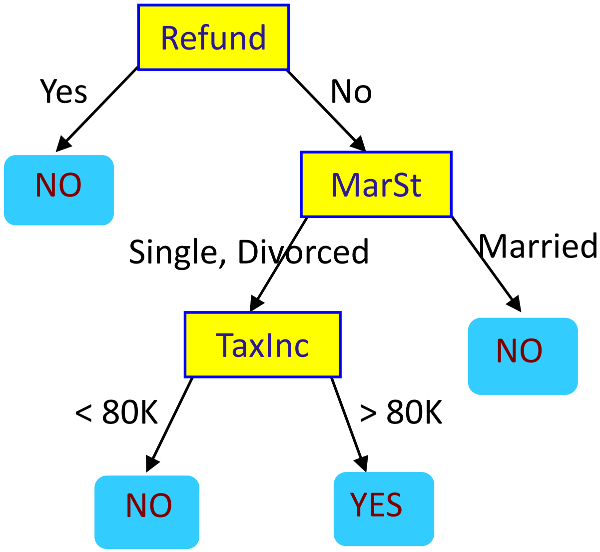
$$I(Y, X_i) = H(Y) - H(Y \mid X_i)$$

How to learn a decision tree

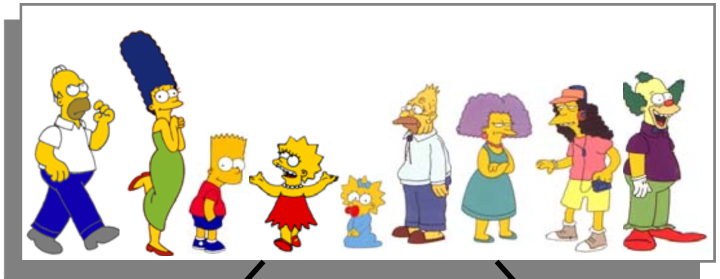
- Top-down induction [ID3, C4.5, CART, …]

Main loop:

1. $X \leftarrow$ the "best" decision attribute for next *node*
2. Assign X as decision attribute for *node*
3. For each value of X, create new descendant of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes



Person	Hair Length	Weight <161	Age <40	Class
 Homer	Short	No	Yes	M
 Marge	Long	Yes	Yes	F
 Bart	Short	Yes	Yes	M
 Lisa	Long	Yes	Yes	F
 Maggie	Long	Yes	Yes	F
 Abe	Short	No	No	M
 Selma	Long	Yes	No	F
 Otto	Long	No	Yes	M
 Krusty	Long	No	No	M
 Comic	Long	No	Yes	?



$$Entropy(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

$$Entropy(4\mathbf{F},5\mathbf{M}) = -(4/9)\log_2(4/9) - (5/9)\log_2(5/9)$$
$$= \mathbf{0.9911}$$

Short

Long

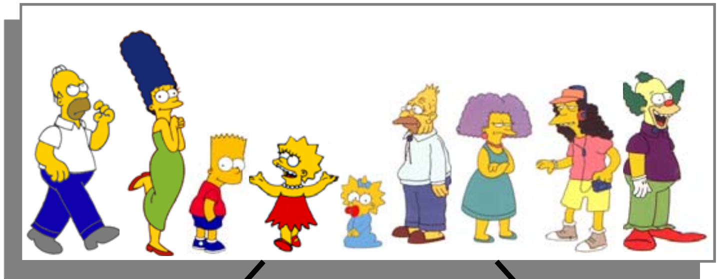
Let us try splitting on *Hair length*

$$Entropy(0\mathbf{F},3\mathbf{M}) = -(0/3)\log_2(0/3) - (1)\log_2(1)$$
$$= \mathbf{0}$$

$$Entropy(4\mathbf{F},2\mathbf{M}) = -(4/6)\log_2(4/6) - (2/6)\log_2(2/6)$$
$$= \mathbf{0.9183}$$

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

$$Gain(\text{Hair Length}) = \mathbf{0.9911} - (3/9 * \mathbf{0} + 6/9 * \mathbf{0.9183}) = \mathbf{0.3789}$$



$$Entropy(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

$$Entropy(4\mathbf{F},5\mathbf{M}) = -(4/9)\log_2(4/9) - (5/9)\log_2(5/9)$$
$$= \mathbf{0.9911}$$

yes

no

Weight <161?

Let us try splitting on *Weight*

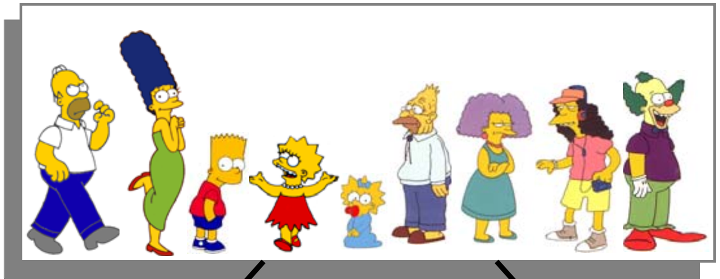
$$Entropy(4\mathbf{F},1\mathbf{M}) = -(4/5)\log_2(4/5) - (1/5)\log_2(1/5)$$
$$= \mathbf{0.7219}$$

$$Entropy(0\mathbf{F},4\mathbf{M}) = -(0/4)\log_2(0/4) - (4/4)\log_2(4/4)$$
$$= \mathbf{0}$$

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

$$Gain(\text{Weight} < 161) = \mathbf{0.9911} - (5/9 * \mathbf{0.7219} + 4/9 * \mathbf{0}\ ) = \mathbf{0.5900}$$

$$Entropy(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

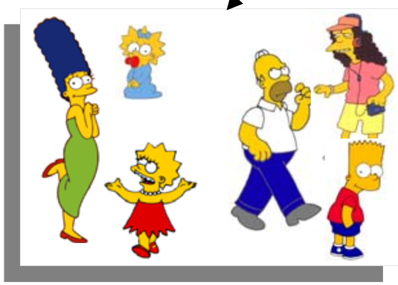


$$Entropy(4F,5M) = -(4/9)\log_2(4/9) - (5/9)\log_2(5/9) = 0.9911$$

yes

no

age <= 40?



Let us try splitting on *Age*

$$Entropy(3F,3M) = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$$

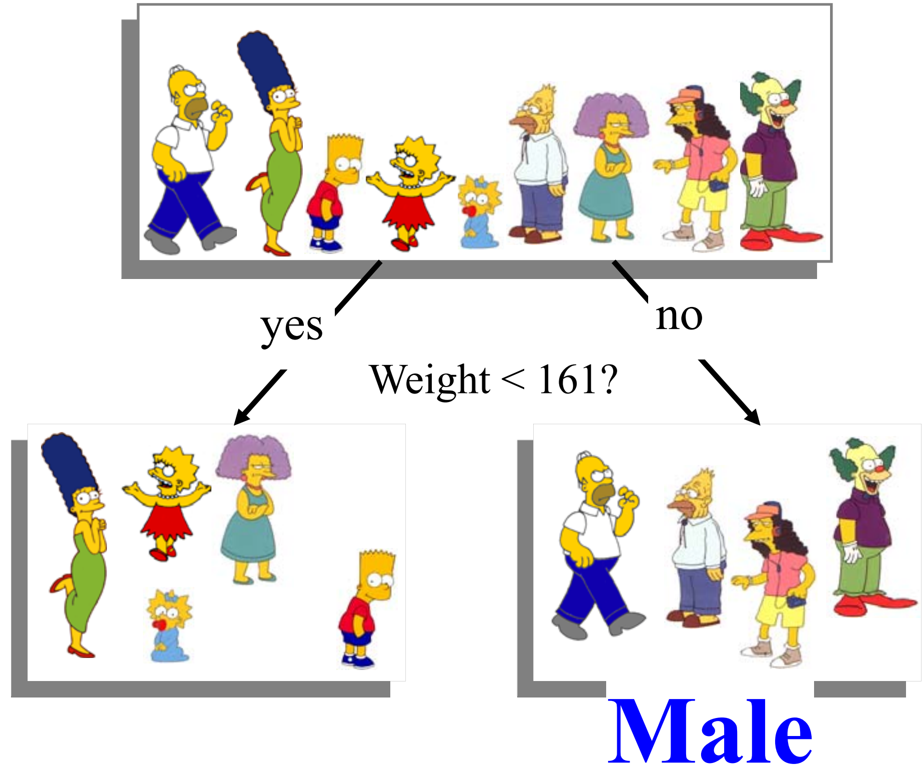
$$Entropy(1F,2M) = -(1/3)\log_2(1/3) - (2/3)\log_2(2/3) = 0.9183$$

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

$$Gain(\text{Age} < 40) = 0.9911 - (6/9 * 1 + 3/9 * 0.9183) = \mathbf{0.0183}$$

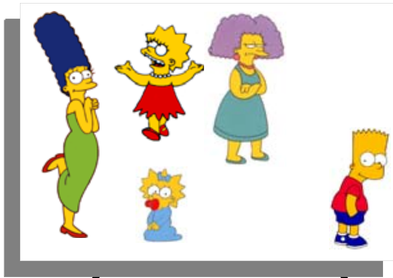
Of the 3 features we had, *Weight* was best. But while people who weigh over 161 are perfectly classified (as males), the under 161 people are not perfectly classified… So we simply recurse!

Person	Hair Length	Weight <161	Age <40	Clas s
Marge	Long	Yes	Yes	F
Bart	Short	Yes	Yes	M
Lisa	Long	Yes	Yes	F
Maggie	Short	Yes	Yes	F
Selma	Long	Yes	No	F



$$Entropy(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

$$Entropy(4\textbf{F},1\textbf{M}) = -(4/5)\log_2(4/5) - (1/5)\log_2(1/5)$$
$$= \mathbf{0.7219}$$



Short

Long

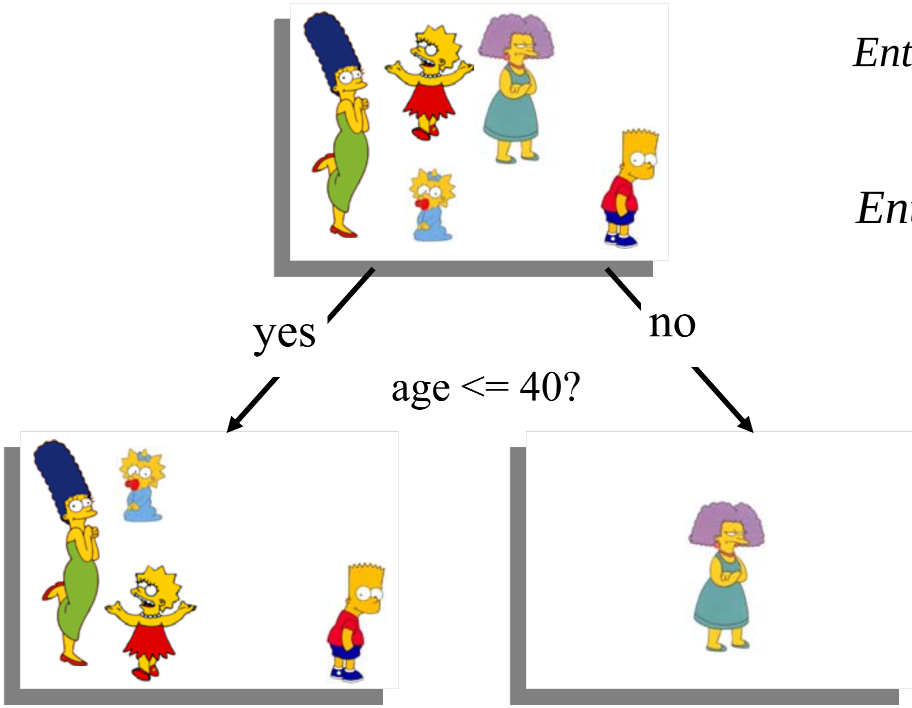
Let us try splitting on *Hair length*

$$Entropy(0\textbf{F},1\textbf{M}) = -(0/1)\log_2(0/1) - (1/1)\log_2(1/1)$$
$$= \mathbf{0}$$

$$Entropy(4\textbf{F},0\textbf{M}) = -(0/4)\log_2(4/4) - 0\log_2(0)$$
$$= \mathbf{0}$$

$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

$$Gain(\text{Hair Length, Weight} < 161) = 0.7219 - (1/5 * 0 + 3/5 * 0) = \mathbf{0.7219}$$



$$Entropy(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

$Entropy(4\mathbf{F},1\mathbf{M}) = -(4/5)\log_2(4/5) - (1/5)\log_2(1/5)$
$= \mathbf{0.7219}$

Let us try splitting on *Age*

$Entropy(3\mathbf{F},1\mathbf{M}) = -(3/4)\log_2(3/4) - (1/4)\log_2(1/4)$
$= \mathbf{0.8113}$

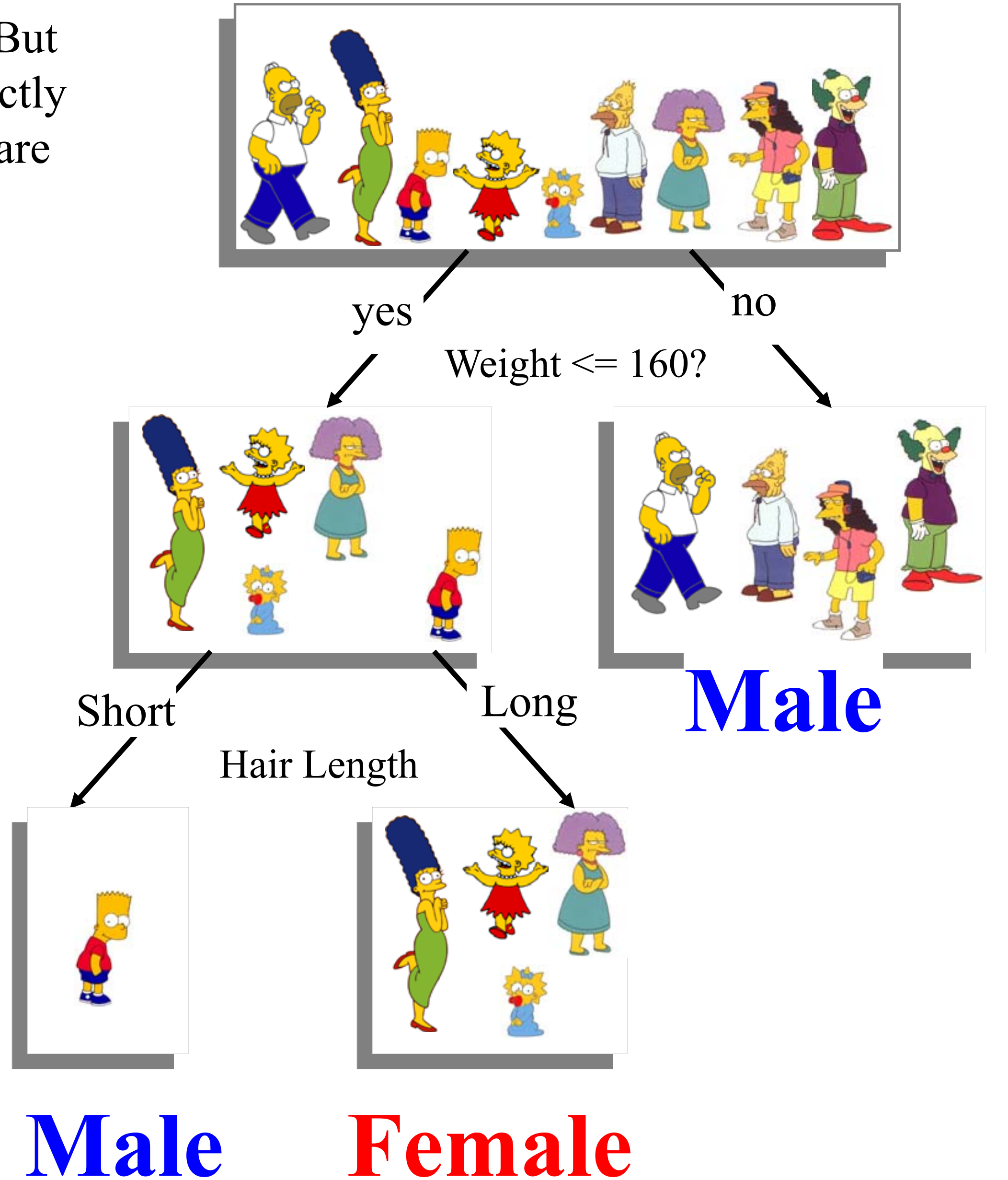
$Entropy(1\mathbf{F},0\mathbf{M}) = -(1/1)\log_2(1) - (0/1)\log_2(0/1)$
$= \mathbf{0}$

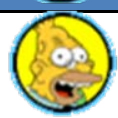
$$Gain(A) = E(Current\ set) - \sum E(all\ child\ sets)$$

$Gain(\text{Age, Weight} < 161) = \mathbf{0.7219} - (3/4 * \mathbf{0.8113} + 1/4 * \mathbf{0}) = \mathbf{0.1134}$

Of the 3 features we had, *Weight* was best. But while people who weigh over 161 are perfectly classified (as males), the under 161 people are not perfectly classified… So we simply recurse!

This time we find that we can split on *Hair length and we done*.!



Person	Hair Length	Weight <161	Age <40	Class
 Homer	Short	No	Yes	M
 Marge	Long	Yes	Yes	F
 Bart	Short	Yes	Yes	M
 Lisa	Long	Yes	Yes	F
 Maggie	Long	Yes	Yes	F
 Abe	Short	No	No	M
 Selma	Long	Yes	No	F
 Otto	Long	No	Yes	M
 Krusty	Long	No	No	M
 Comic	Long	No	Yes	?

Hair Length	Weight <161	Age <40	Class	Count
Short	No	Yes	**M**	**1**
Long	Yes	Yes	**F**	**3**
Short	Yes	Yes	**M**	**1**
Short	No	No	**M**	**1**
Long	Yes	No	**F**	**1**
Long	No	Yes	**M**	**1**
Long	No	No	**M**	**1**

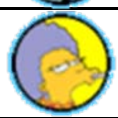
Building More General Decision Trees

- Build a decision tree (≥ 2 level) Step by Step.

Next

- Building a decision tree with continuous input feature.

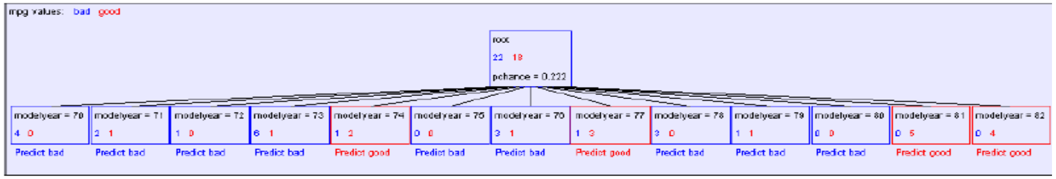
- Building a quad decision tree.

Person	Hair Length	Weight <161	Age <40	Class
 Homer	0”	No	Yes	M
 Marge	10”	Yes	Yes	F
 Bart	2”	Yes	Yes	M
 Lisa	6”	Yes	Yes	F
 Maggie	4”	Yes	Yes	F
 Abe	1”	No	No	M
 Selma	8”	Yes	No	F
 Otto	10”	No	Yes	M
 Krusty	6”	No	No	M

Real-Values input

What should we do if some of the input features are real-valued?

“One branch for each numeric value” idea:



Hopeless: with such high branching factor will shatter the dataset and over fit

After pruning, it would likely to end up with a single root node.

A better idea: thresholded splits

- Suppose X is real valued.
- Define $IG(Y|X:t)$ as $H(Y) - H(Y|X:t)$
- Define $H(Y|X:t) =$

  $$H(Y|X < t)\, P(X < t) + H(Y|X >= t)\, P(X >= t)$$

  - $IG(Y|X:t)$ is the information gain for predicting Y if all you know is whether X is greater than or less than t
- Then define $IG^*(Y|X) = max_t\, IG(Y|X:t)$
- For each real-valued attribute, use $IG^*(Y|X)$ for assessing its suitability as a split

$$Entropy(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

Example

Hair Length	0”	1”	2”	4”	6”	6”	8”	10”	10”
Class	**M**	**M**	**M**	**F**	**F**	**M**	**F**	**F**	**M**

$$Entropy(4F,5M) = -(4/9)\log_2(4/9) - (5/9)\log_2(5/9)$$
$$= \mathbf{0.9911}$$

To increase the complexity of a decision tree by the same amount for any decision, only binary splits of the form $hair-length < H$ vs. $hair-length \geq H$ are allowed.

$$\begin{aligned}
Entropy(H < 4) &= Entropy(0F, 3M) && (1)\\
&= -(\frac{0}{3}\log_2\frac{0}{3} + \frac{3}{3}\log_2\frac{3}{3}) && (2)\\
&= 0 && (3)\\
Entropy(H \geq 4) &= Entropy(4F, 2M) && (4)\\
&= -(\frac{4}{6}\log_2\frac{4}{6} + \frac{2}{6}\log_2\frac{2}{6}) && (5)\\
&= 0.9183 && (6)\\
Gain(H = 4) &= 0.9911 - (\frac{3}{9}\times(0) + \frac{6}{9}\times 0.9183) && (7)\\
&= 0.3789 && (8)
\end{aligned}$$

Example

Hair Length	0”	1”	2”	4”	6”	6”	8”	10”	10”
Class	M	M	M	F	F	M	F	F	M

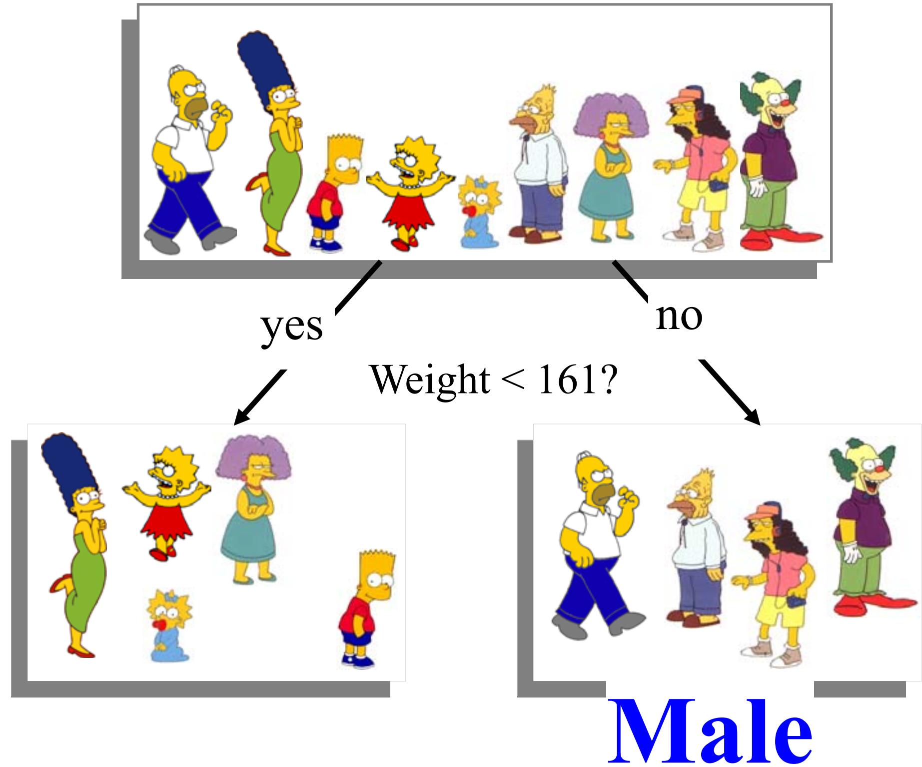
$$
\begin{aligned}
Gain(H=1) &= 0.9911 - (\frac{1}{9} \times 0 + \frac{8}{9} \times 1) && (1)\\
&= 0.1022 && (2)\\
Gain(H=2) &= 0.9911 - (\frac{2}{9} \times 0 + \frac{7}{9} \times 0.9852) && (3)\\
&= 0.2248 && (4)\\
Gain(H=4) &= 0.9911 - (\frac{3}{9} \times 0 + \frac{6}{9} \times 0.9183) && (5)\\
&= 0.3789 && (6)\\
Gain(H=6) &= 0.9911 - (\frac{4}{9} \times 0.8113 + \frac{5}{9} \times 0.9710) && (7)\\
&= 0.0911 && (8)\\
Gain(H=8) &= 0.9911 - (\frac{6}{9} \times 0.9183 + \frac{3}{9} \times 0.9183) && (9)\\
&= 0.0728 && (10)\\
Gain(H=10) &= 0.9911 - (\frac{7}{9} \times 0.9852 + \frac{2}{9} \times 1) && (11)\\
&= 0.0026 && (12)
\end{aligned}
$$

However,…

- *Gain*(Hair Length < 4") = **0.9911** – (3/9 * **0** + 6/9 * **0.9183**) = **0.3789**
- ***Gain*(Weight < 161)** = **0.9911** – (5/9 * **0.7219** + 4/9 * **0**) **= 0.5900**
- *Gain*(Age < 40) = **0.9911** – (6/9 * **1** + 3/9 * **0.9183**) = **0.0183**

Of the 3 features we had, *Weight* was best. But while people who weigh over 160 are perfectly classified (as males), the under 160 people are not perfectly classified… So we simply recurse!

Person	Hair Length	Weight <161	Age <40	Clas s
Marge	10”	Yes	Yes	F
Bart	2”	Yes	Yes	M
Lisa	6”	Yes	Yes	F
Maggie	4”	Yes	Yes	F
Selma	8”	Yes	No	F



Example

Hair Length	2”	4”	6”	8”	10”
Class	**M**	**F**	**F**	**F**	**F**

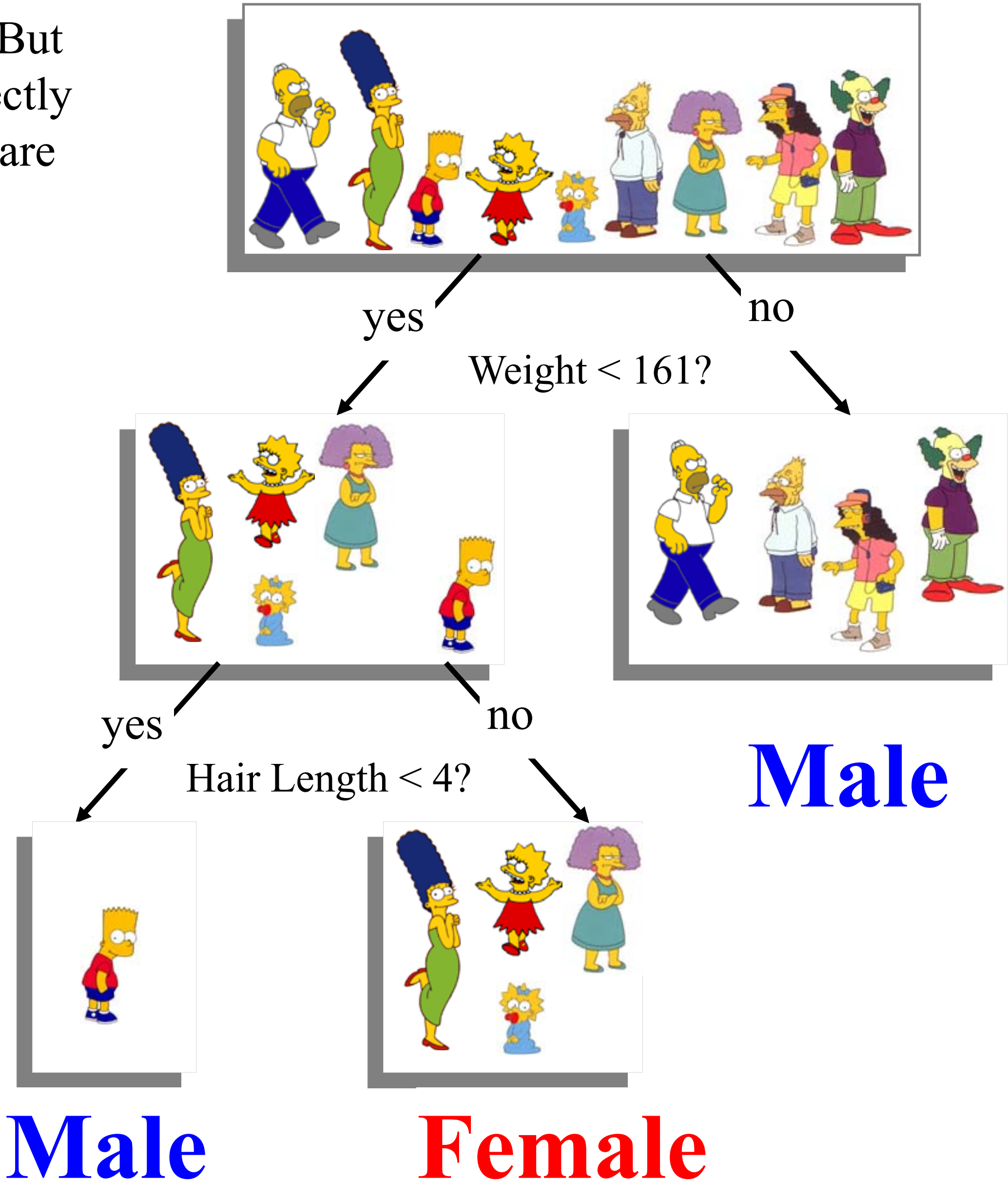
Entropy(4**F**,1**M**) = $-(4/5)\log_2(4/5) - (1/5)\log_2(1/5) =$ **0.7219**

Gain(Hair Length < 4”, Weight < 161) = **0.7219** – (1/5 * **0** + 4/5 * **0**) = **0.7219**

Gain(Age, Weight < 161) = **0.7219** – (3/4 * **0.8113** + 1/4 * **0**) = **0.1134**

Of the 3 features we had, *Weight* was best. But while people who weigh over 160 are perfectly classified (as males), the under 160 people are not perfectly classified… So we simply recurse!

This time we find that we can split on *Hair length*, and we are done!



Attributes with Many Values

Problem:

- If attribute has many values, $Gain$ will select it
- Imagine using $Date = Jun\_3\_1996$ as attribute

One approach: use $GainRatio$ instead

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

$$SplitInformation(S, A) \equiv - \sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

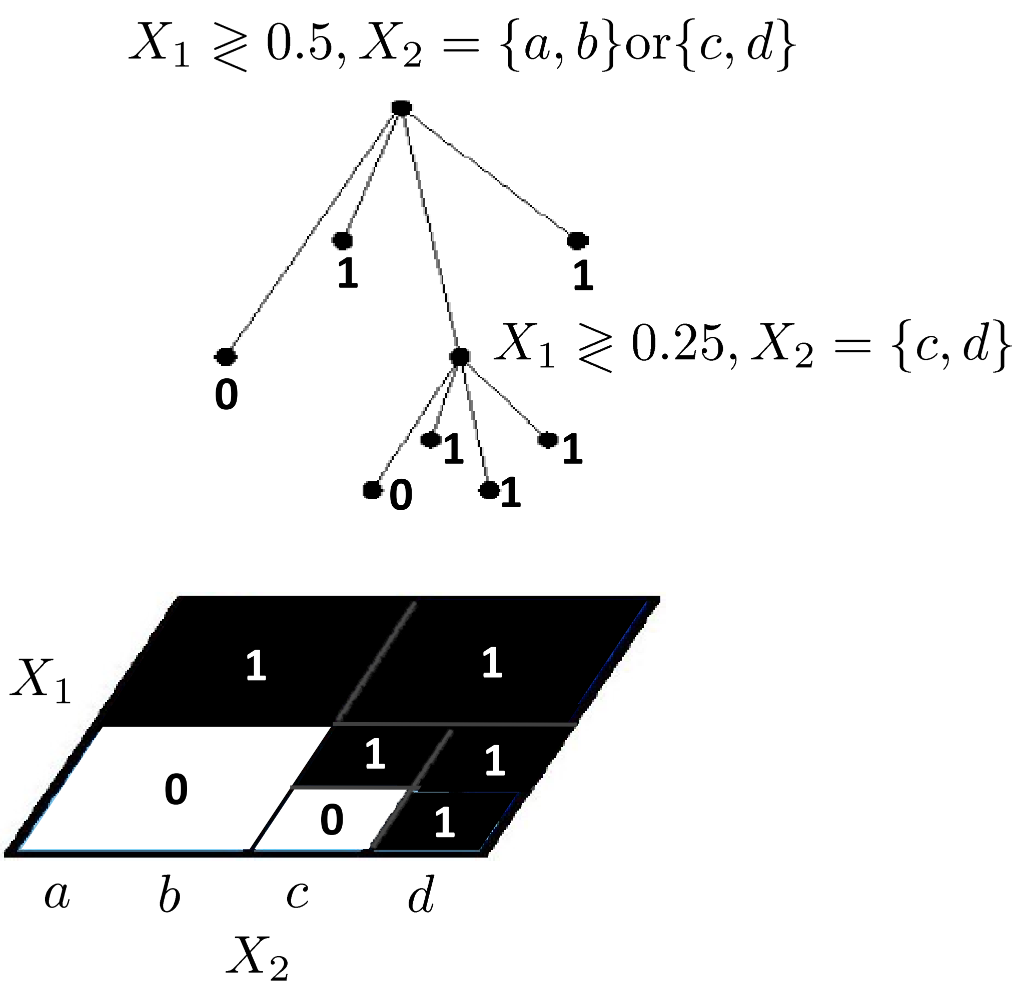
where S_i is subset of S for which A has value v_i

Building More General Decision Trees

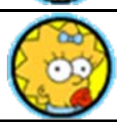
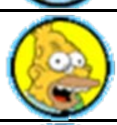
- Build a decision tree ($\geq$ 2 level) Step by Step.
- Building a decision tree with continuous input feature.
- Building a quad decision tree.

Next

Decision Tree more generally…



- Features can be discrete, continuous or categorical
- Each internal node: test some set of features $\{X_i\}$
- Each branch from a node: selects a set of value for $\{X_i\}$
- Each leaf node: predict Y

Person		Hair Length	Weight	Class
	Homer	0”	250	M
	Marge	10”	150	F
	Bart	2”	90	M
	Lisa	6”	78	F
	Maggie	4”	20	F
	Abe	1”	170	M
	Selma	8”	160	F
	Otto	10”	180	M
	Krusty	6”	200	M

Hair length:

0-2 Short

3-6 Medium

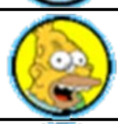
≥7 Long

Weight:

0-100 Light

100-175 Normal

≥175 Heavy

Person	Hair Length	Weight	Class
 Homer	S	Heavy	M
 Marge	L	Normal	F
 Bart	S	Light	M
 Lisa	M	Light	F
 Maggie	M	Light	F
 Abe	S	Normal	M
 Selma	L	Normal	F
 Otto	L	Heavy	M
 Krusty	M	Heavy	M

Hair length:

0-2 Short

3-6 Medium

≥7 Long

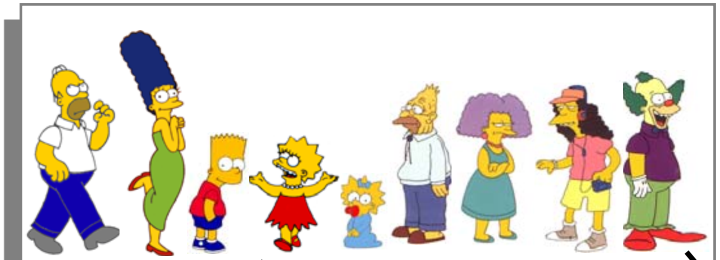
Weight:

0-100 Light

100-175 Normal

≥175 Heavy

$$Entropy(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$



$$Entropy(4\mathbf{F},5\mathbf{M}) = -(4/9)\log_2(4/9) - (5/9)\log_2(5/9) = \mathbf{0.9911}$$

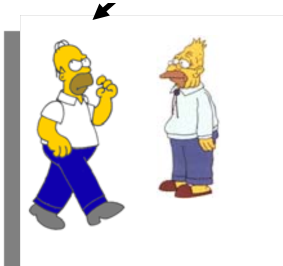
Short, {Normal, Heavy}

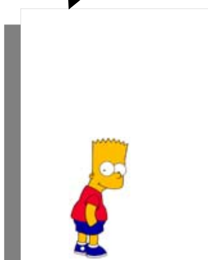
Short, Light

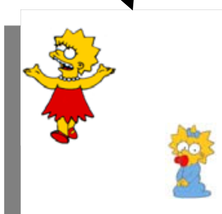
{M, L} Light

{M, L} {Normal, Heavy}

Let us try splitting on *Hair length {S} vs. {M, L}* and *Weight: {Light} vs. {Normal, Heavy}*







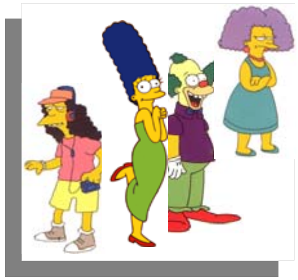
$$Entropy(0\mathbf{F},2\mathbf{M}) = -(0/2)\log_2(0/2) - (2/2)\log_2(2/2) = \mathbf{0}$$

$$Entropy(0\mathbf{F},1\mathbf{M}) = \mathbf{0}$$

$$Entropy(2\mathbf{F},0\mathbf{M}) = \mathbf{0}$$

$$Entropy(2\mathbf{F},2\mathbf{M}) = \mathbf{1}$$

*{M, L} * {Normal, Heavy}*
Let us try re-splitting on *Hair length: M vs. L*
and *Weight: Normal vs. Heavy*

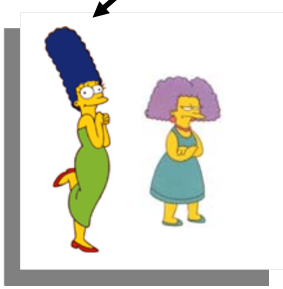


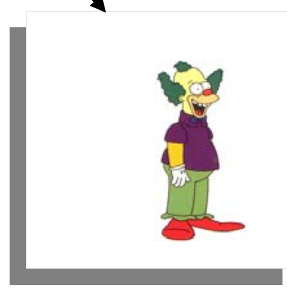
L, Normal

L, Heavy

M, Normal

M, Heavy





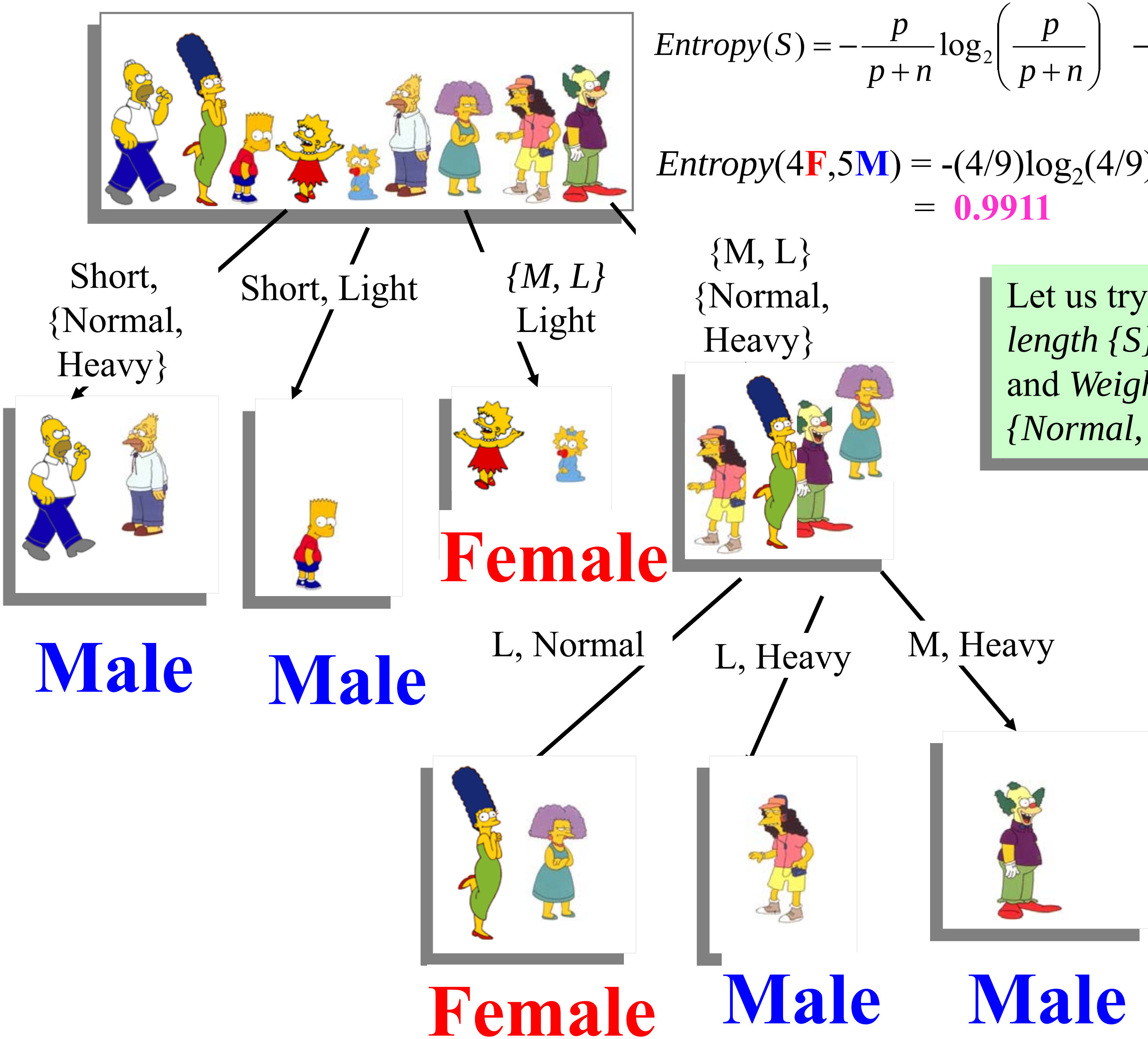
Entropy(2**F**,0**M**) = ***0***

Entropy(0**F**,1**M**) = ***0***

Entropy(2**F**,2**M**) = ***1***

$$Entropy(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

$$Entropy(4\mathbf{F},5\mathbf{M}) = -(4/9)\log_2(4/9) - (5/9)\log_2(5/9) = \mathbf{0.9911}$$

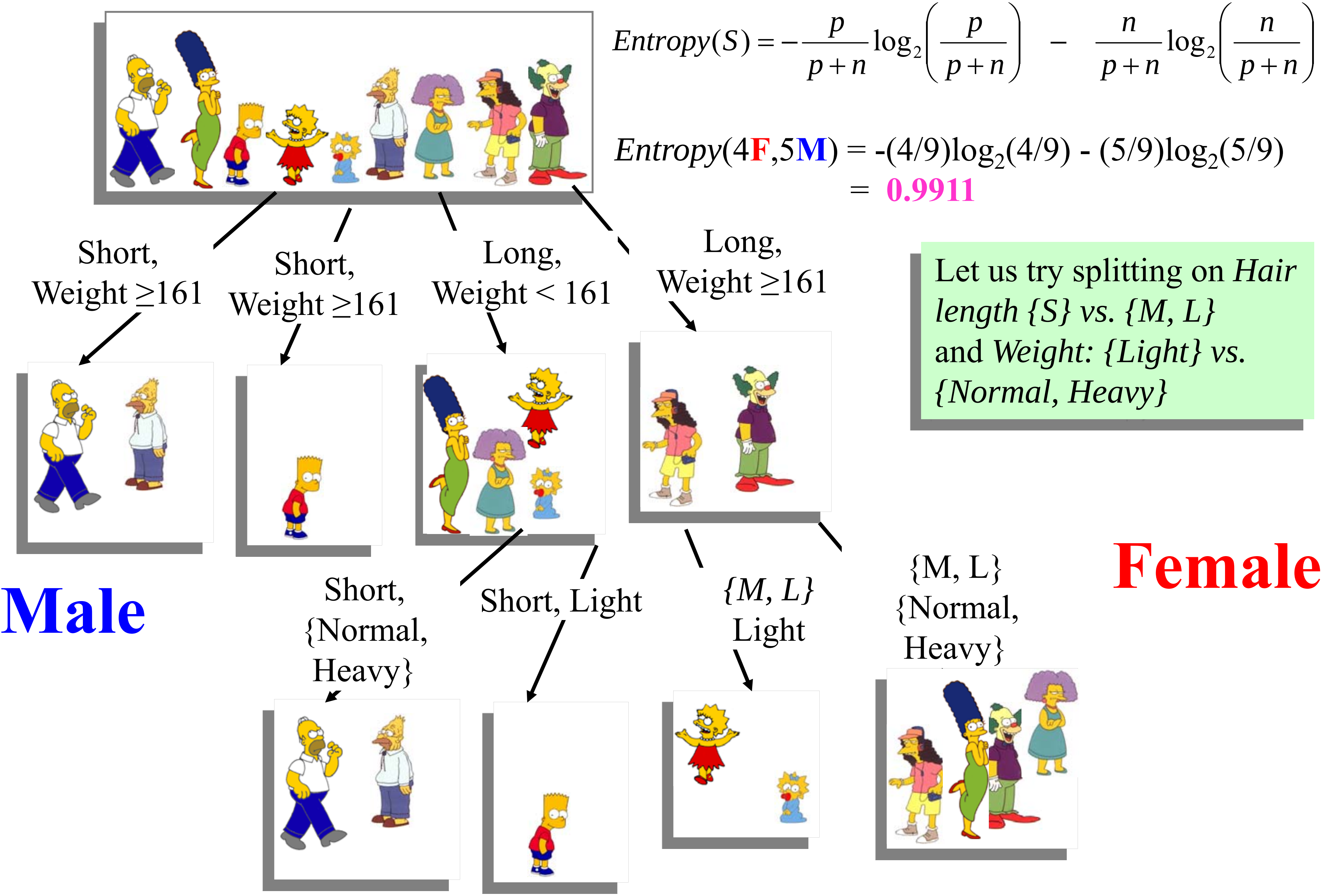


Let us try splitting on *Hair length {S} vs. {M, L}* and *Weight: {Light} vs. {Normal, Heavy}*

$$Entropy(S) = -\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log_2\left(\frac{n}{p+n}\right)$$

$$Entropy(4\mathbf{F},5\mathbf{M}) = -(4/9)\log_2(4/9) - (5/9)\log_2(5/9)$$
$$= \mathbf{0.9911}$$

Let us try splitting on *Hair length {S} vs. {M, L}* and *Weight: {Light} vs. {Normal, Heavy}*



Model and Selection: AIC & BIC

Bias, Variance, and Model Complexity

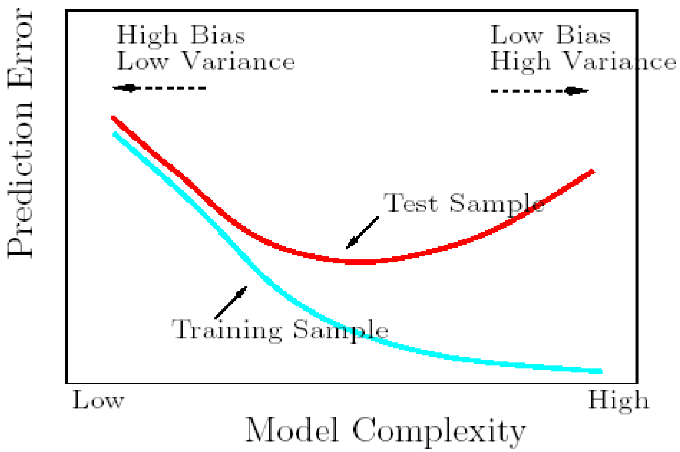


Figure 7.1: *Behavior of test sample and training sample error as the model complexity is varied.*

- Bias-Variance trade-off again
- Generalization: test sample vs. training sample performance
  - Training data usually monotonically increasing performance with model complexity

Measuring Performance

- target variable Y
- Vector of inputs X
- Prediction model $\hat{f}(X)$

- Typical Choices of Loss function

$$L\left(Y, \hat{f}(X)\right) = \begin{cases} \left(Y - \hat{f}(X)\right)^2 & \text{squared error} \\ \left|Y - \hat{f}(X)\right| & \text{absolute error} \end{cases}$$

Generalization Error

- Test error aka. Generalization error

$$\boxed{Err = E\left[L\left(Y, \hat{f}(X) \right) \right]}$$

- Note: This expectation averages anything that is random, including the randomness in the training sample that it produced

- Training error

- $$\boxed{\overline{err} = \frac{1}{N} \sum_{i=1}^{n} L\left(y_i, \hat{f}(x_i) \right)}$$
  - average loss over training sample
  - not a good estimate of test error (next slide)

Training Error

- Training error - Overfitting
  - not a good estimate of test error
  - consistently decreases with model complexity
  - drops to zero with high enough complexity

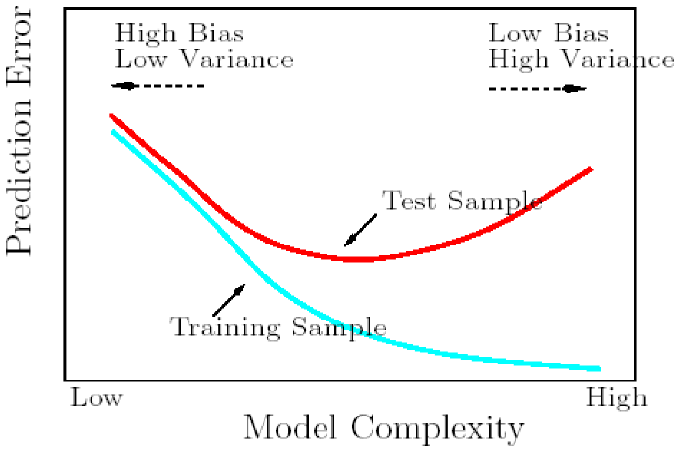


Figure 7.1: *Behavior of test sample and training sample error as the model complexity is varied.*

Categorical Data

- same for categorical responses
- $$p_k(X) = pr(G = k \mid X)$$
  $$\hat{G}(X) = \arg\max_k \hat{p}_k(X)$$
- Typical Choices of Loss functions:

Test Error again:

$$Err = E[L(G, \hat{p}(x))]$$

Training Error again:

$$\overline{err} = \frac{-2}{N}\sum_{i=1}^{N} \log \hat{p}_{g_i}(x_i)$$

$$L\left(G, \hat{G}(X)\right) = I\left(G \neq \hat{G}(X)\right) \quad 0-1\ loss$$

$$L\left(G, \hat{p}(X)\right) = -2\sum_{k=1}^{K} I(G = k) \log \hat{p}_k(X) = -2 \log \hat{p}_G(X) \quad \log-likelihood$$

Log-likelihood = cross-entropy loss = deviance

Loss Function for General Densities

- For densities parameterized by theta:
- Log-likelihood function can be used as a loss-function

$\Pr_{\theta(X)}(Y)$ density of Y with predictor X

$$\boxed{L\big(Y,\theta(X)\big) = -2\log \Pr_{\theta(X)}(Y)}$$

Two separate goals

- Model selection:
  - Estimating the performance of different models in order to choose the (approximate) best one
- Model assessment:
  - Having chosen a final model, estimating its prediction error (generalization error) on new data

- Ideal situation: split data into the 3 parts for *training*, *validation (est. prediction error+select model)*, and *testing (assess model)*
- Typical split*: 50% / 25% / 25%*

- Remainder of the chapter: Data-poor situation
- => *Approximation of validation* step either analytically (AIC, BIC, MDL, SRM) or by efficient sample reuse (cross-validation, bootstrap)

Bias-Variance Decomposition

$$Y = f(X) + \varepsilon,\ \ E(\varepsilon) = 0,\ \ \ Var(\varepsilon) = \sigma_\varepsilon^2$$

- Then for an input point $X = x_0$ using unit-square loss and regression fit:

$$\begin{aligned} Err(x_0) &= E\left[\left(Y - \hat{f}(x_0)\right)^2 \mid X = x_0\right] \\ &= \sigma_\varepsilon^2 + \left[E\hat{f}(x_0) - f(x_0)\right]^2 + E\left[\hat{f}(x_0) - E\hat{f}(x_0)\right]^2 \\ &= \sigma_\varepsilon^2 + Bias\left[\hat{f}(x_0)\right]^2 + Var\left[\hat{f}(x_0)\right] \end{aligned}$$

Irreducible Error | Bias^2 | Variance

Irreducible Error	Bias^2	Variance
variance of the target around the true mean	Amount by which average estimate differs from the true mean	Expected deviation of f^ around its mean

Bias-Variance Decomposition

$$Err\left(x_0\right) = \sigma_\varepsilon^2 + Bias\left[\hat{f}\left(x_0\right)\right]^2 + Var\left[\hat{f}\left(x_0\right)\right]$$

kNN: $$Err\left(x_0\right) = \sigma_\varepsilon^2 + \left[f\left(x_o\right) - \frac{1}{k}\sum_{l=1}^{k} f\left(x_{(l)}\right)\right]^2 + \sigma_\varepsilon{}^2 / k$$

Linear Model Fit: $\hat{f}_p\left(x\right) = \hat{\beta}^T x$

$$Err\left(x_0\right) = \sigma_\varepsilon^2 + \left[f\left(x_o\right) - E\hat{f}_p\left(x_0\right)\right]^2 + \left\|h\left(x_0\right)\right\|^2 \sigma_\varepsilon^2$$

$$\text{where } \mathrm{h}\left(\mathrm{x}_0\right) = \left(X^T X\right)^{-1} X^T y$$

Bias-Variance Decomposition

Linear Model Fit: $\hat{f}_p(x) = \hat{\beta}^T x$

$$Err(x_0) = \sigma_\varepsilon^2 + \left[f(x_o) - E\hat{f}_p(x_0) \right]^2 + \left\| h(x_0) \right\|^2 \sigma_\varepsilon^2$$

$$\text{where } \mathrm{h}(\mathrm{x}_0) = \left(X^T X \right)^{-1} X^T y \quad \text{... N-dim weight vector}$$

$$\text{average over sample values } \mathrm{x}_\mathrm{i} :$$

$$\frac{1}{N}\sum_{i=1}^{N} Err(x_i) = \sigma_\varepsilon^2 + \frac{1}{N}\sum_{i=1}^{N}\left[f(x_i) - E\hat{f}(x_i) \right]^2 + \frac{p}{N}\sigma_\varepsilon^2 \quad \text{... in-sample error}$$

Model complexity is directly related to the number of parameters p

Bias-Variance Decomposition

$$Err\left(x_0\right) = \sigma_\varepsilon^2 + Bias\left[\hat{f}\left(x_0\right)\right]^2 + Var\left[\hat{f}\left(x_0\right)\right]$$

For ridge regression and other linear models, variance same as before, but with diff't weights.

Parameters of the best fitting linear approximation

$$\beta_* = \arg\min_\beta E\left(f\left(X\right) - \beta^T X\right)^2$$

Further decompose the bias:

$$E_{x_0}[f(x_0) - E\hat{f}_\alpha(x_0)]^2 = E_{x_0}[f(x_0) - \beta_*^T x_0]^2 + E_{x_0}[\beta_*^T x_0 - E\beta_\alpha^T x_0]^2$$

$$= Ave[\text{Model Bias}]^2 + Ave[\text{Estimation Bias}]^2$$

Least squares fits best linear model -> no estimation bias
Restricted fits -> positive estimation bias in favor of reduced variance

Optimism of the Training Error Rate

- Typically: training error rate < true error
- (same data is being used to fit the method and assess its error)

$$\underbrace{\overline{err} = \frac{1}{N}\sum_{i=1}^{n} L\left(y_i, \hat{f}(x_i)\right)}_{\text{overly optimistic}} \quad < \quad Err = E\left[L\left(Y, \hat{f}(X)\right)\right]$$

Optimism of the Training Error Rate

Err … kind of extra-sample error: test features don't need to coincide with training feature vectors

Focus on in-sample error:

$$Err_{in} = \frac{1}{N}\sum_{i=1}^{N} E_Y E_{Y^{new}} L\left(Y_i^{new}, \hat{f}\left(x_i\right)\right)$$

Y^{new} … observe N **new** response values at each of training points x_i, i=1, 2, …,N

$$\text{optimism: } op \equiv Err_{in} - E_y\left(\overline{err}\right)$$

for squared error 0-1 and other loss functions:

$$op = \frac{2}{N}\sum_{i=1}^{N} Cov\left(\hat{y}_i, y_i\right)$$

The amount by which $\overline{\text{err}}$ underestimates the true error depends on how strongly y_i affects its own prediction.

Optimism of the Training Error Rate

Summary: $$Err_{in} = E_y\left(\overline{err}\right) + \frac{2}{N}\sum_{i=1}^{N} Cov\left(\hat{y}_i, y_i\right)$$

The harder we fit the data, the greater $Cov\left(\hat{y}_i, y_i\right)$ will be, thereby increasing the optimism.

- For linear fit with d indep inputs/basis funcs:

$$Err_{in} = E_y\left(\overline{err}\right) + \frac{2}{N} d\sigma_{\varepsilon}^{2}$$

  - optimism $\uparrow$ linearly with # d of basis functions
  - Optimism $\downarrow$ as training sample size $\uparrow$

Optimism of the Training Error Rate

- Ways to estimate prediction error:
  - Estimate optimism and then add it to training error rate $\overline{\mathrm{err}}$
    - AIC, BIC, and others work this way, for a special class of estimates that are linear in their parameters
  - Direct estimates of the sample error Err
    - Cross-validation, bootstrap
    - Can be used with any loss function, and with nonlinear, adaptive fitting techniques
-

Estimates of In-Sample Prediction Error

- General form of the in-sample estimate:

$$\boxed{\hat{\text{E}}\text{rr}_{in} = \overline{err} + \hat{o}p}$$

with estimate of optimism

- For linear fit and with $Err_{in} = E_y\left(\overline{err}\right) + \frac{2}{N} d\sigma_\varepsilon^2$:

$$\boxed{C_p = \overline{err} + \frac{2d}{N}\hat{\sigma}_\varepsilon^2, \text{ so called } C_p \text{ statistic}}$$

$\hat{\sigma}_\varepsilon^2$ ... estimate of noise variance, from mean-squared error of low-bias model

d... # of basis functions

N... training sample size

Estimates of In-Sample Prediction Error

- Similarly: Akaike Information Criterion (AIC)
  - More applicable estimate of Err_{in} when log-likelihood function is used

$$\text{For } N \to \infty: \quad -2E\left[\log \Pr_{\hat{\theta}}(Y)\right] \approx -\frac{2}{N} E[\text{loglik}] + 2\frac{d}{N}$$

$\Pr_{\theta}(Y)$... family density for Y (containing the true density)

$\hat{\theta}$... ML estimate of θ

$$\text{loglik} = \sum_{i=1}^{N} \log \Pr_{\hat{\theta}}(y_i)$$

Maximized log-likelihood due to ML estimate of theta

AIC

$$\text{For } N \to \infty: \quad -2E\left[\log \Pr_{\hat{\theta}}(Y)\right] \approx -\frac{2}{N} E\left[\log \text{lik}\right] + 2\frac{d}{N}$$

For example, for logistic regression model, using binomial log-likelihood:

$$\boxed{AIC = -\frac{2}{N} \cdot \text{loglik} + 2 \cdot \frac{d}{N}}$$

To use AIC for model selection: choose the model giving smallest AIC over the set of models considered.

$$\boxed{AIC(\alpha) = \overline{err}(\alpha) + 2\frac{d(\alpha)}{N}\hat{\sigma}_{\varepsilon}^{2}}$$

$f_{\hat{\alpha}}(x)$... set of models, α... tuning parameter

$\overline{\text{err}}(\alpha)$... training error, $d(\alpha)$... # parameters

AIC

- Function AIC(α) estimates test error curve
- If basis functions are chosen adaptively with d<p inputs:

$$\sum_{i=1}^{N} Cov(\hat{y}_i, y_i) = d\sigma_\varepsilon^2$$

- no longer holds => optimism exceeds

$$(2d / N)\sigma_\varepsilon^2$$

- effective number of parameters fit > d

Using AIC to select the # of basis functions

- Input vector: log-periodogram of vowel; Quantized to 256 uniformly spaced f
- Linear logistic regression model
- Coefficient function $\beta(f) = \sum_{m=1}^{M} h_m(f)\theta_m$
  - Expansion of M spline basis functions
  - For any M, a basis of natural cubic splines is used for the knots h_m chosen uniformly over the range of frequencies, i.e. $d(\alpha) = d(M) = M$
- AIC approximately minimizes Err(M) for both entropy and 0-1 loss
-

$$\frac{2}{N}\sum_{i=1}^{N} Cov(\hat{y}_i, y_i) = \frac{2d}{N}\sigma_\varepsilon^2 \text{... simple formula for linear case}$$

Using AIC to select the # of basis functions

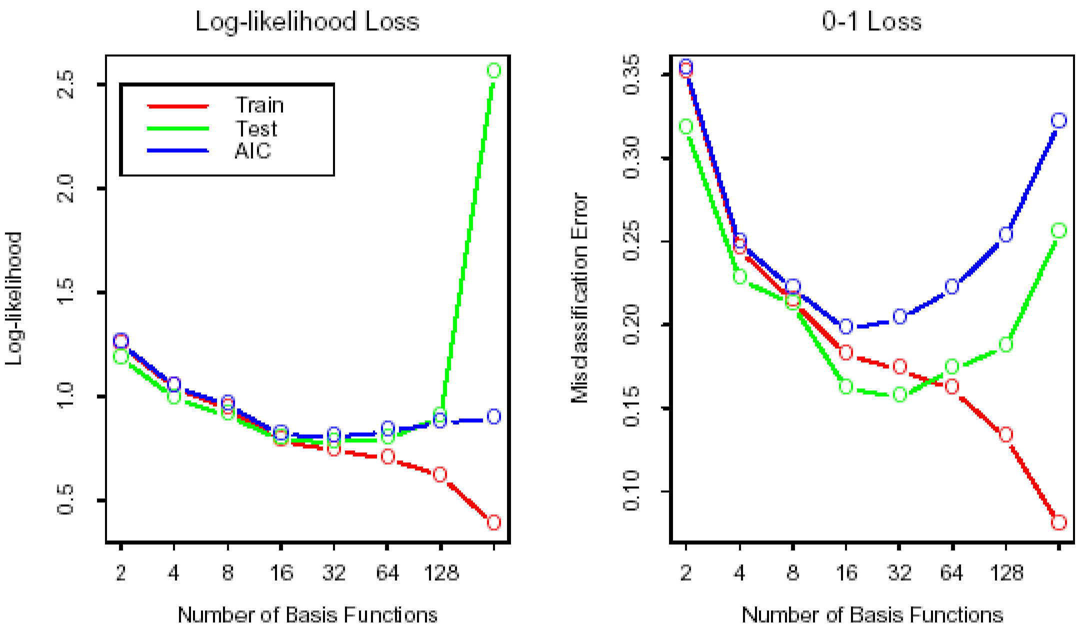


Figure 7.4: *AIC used for model selection for the phoneme recognition example of Section 5.2.3.*

$$\frac{2}{N}\sum_{i=1}^{N} Cov(\hat{y}_i, y_i) = \frac{2d}{N}\sigma_\varepsilon^2$$

Approximation does not hold, in general, for 0-1 case, but it does o.k. (Exact only for linear models w/ additive errors and sq err loss)

Effective Number of Parameters

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

Vector of Outcomes, similarly for predicitons

$\hat{y} = Sy$ Linear fit (e.g. linear regression, quadratic shrinkage – ridge, splines)

$S \ldots \mathrm{N} \times \mathrm{N}$ matrix, depends on input vector $\mathrm{x_i}$ but not on $\mathrm{y_i}$

$\boxed{\text{effective number of parameters: } \mathrm{d}(\mathrm{S}) = trace(S)}$

c.f. $Cov(\hat{y}, y)$

d(s) is the correct *d* for C_p

$$\boxed{C_p = \overline{err} + \frac{2d}{N}\hat{\sigma}_\varepsilon^2}$$

Bayesian Approach and BIC

- Like AIC used in when fitting by max log-likelihood

Bayesian Information Criterion (BIC):

$$\boxed{BIC = -2\,\text{loglik} + (\log N)d}$$

Assuming Gaussian model : σ_ε^2 known,

$$-2 \cdot \text{loglik} \approx \sum_i (y_i - \hat{f}(x_i))^2 / \sigma_\varepsilon^2 = N \cdot \overline{err} / \sigma_\varepsilon^2$$

$$\text{then } BIC = \frac{N}{\sigma_\varepsilon^2}[\overline{err} + (\log N) \cdot \frac{d}{N} \sigma_\varepsilon^2]$$

BIC proportional to AIC except for log(N) rather than factor of 2. For $N>e^2$ (approx 7.4), BIC penalizes complex models more heavily.

BIC Motivation

- Given a set of candidate models $\mathbf{M}_m, m = 1 \ldots M$ and model parameters θ_{m}
- Posterior probability of a given model: $\Pr(\mathbf{M}_m \mid \mathbf{Z}) \propto \Pr(\mathbf{M}_m) \cdot \Pr(\mathbf{Z} \mid \mathbf{M}_m)$
- Where $\mathbf{Z}$ represents the training data $\{x_i, y_i\}_1^N$
- To compare two models, form the posterior odds:

$$\frac{\Pr(\mathbf{M}_m \mid \mathbf{Z})}{\Pr(\mathbf{M}_l \mid \mathbf{Z})} = \frac{\Pr(\mathbf{M}_m)}{\Pr(\mathbf{M}_l)} \cdot \frac{\Pr(\mathbf{Z} \mid \mathbf{M}_m)}{\Pr(\mathbf{Z} \mid \mathbf{M}_l)}$$

- If odds > 1, then choose model m. Prior over models (left half) considered constant. Right half, contribution of data (Z) to posterior odds, is called the Bayes factor BF(Z).
- Need to approximate $Pr(Z|M_m)$. Various chicanery and approximations (pp. 207) gets us BIC.
- Can est. posterior from BIC and compare relative merits of models.

BIC: How much better is a model?

- But we may want to know how various models stack up (not just *ranking*) relative to one another:
- Once we have the BIC:
- Denominator normalizes the result and now we can assess the relative merits of each model

$$\text{estimate of } \Pr(\mathbf{M}_m \mid \mathbf{Z}) \equiv \frac{e^{-\frac{1}{2}\cdot BIC_m}}{\sum_{l=1}^{M} e^{-\frac{1}{2}\cdot BIC_l}}$$