Stochastic Gradient Descent

Because N → is huge "big data"

Machine Learning – CSE446

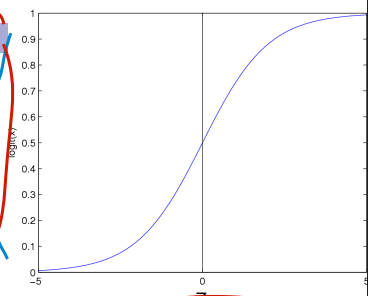
Carlos Guestrin

University of Washington

April 19, 2013

Logistic Regression

Logistic function (or Sigmoid): $\frac{1}{1+exp(-z)}$



- Learn P(Y|**X**) directly
  - Assume a particular functional form for link function
  - Sigmoid applied to a linear function of the input features:

$$P(Y=0|X,W) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

choice

z: linear just like in reg.

$\mathbb{R}$

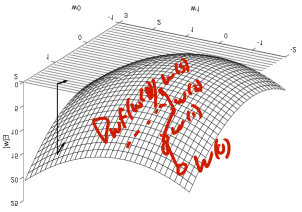
$w_0 + \sum_i w_i x_i$ ← not bounded, could be neg.

after logistic fcn, output is in $[0,1]$

Features can be discrete or continuous!

Optimizing concave function – Gradient ascent

- Conditional likelihood for Logistic Regression is concave. Find optimum with gradient ascent



Gradient: $\nabla_{\mathbf{w}} l(\mathbf{w}) = [\frac{\partial l(\mathbf{w})}{\partial w_0}, \ldots, \frac{\partial l(\mathbf{w})}{\partial w_n}]'$

Update rule: $\Delta \mathbf{w} = \eta \nabla_{\mathbf{w}} l(\mathbf{w})$

Step size, $\eta > 0$

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \frac{\partial l(\mathbf{w})^{(t)}}{\partial w_i}$$

- Gradient ascent is simplest of optimization approaches
  - e.g., Conjugate gradient ascent can be much better

Gradient Ascent for LR

Gradient ascent algorithm: iterate until change < ε

$$w_0^{(t+1)} \leftarrow w_0^{(t)} + \eta \sum_j [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w}^{(t)})]$$

For i=1,…,k,

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w}^{(t)})]$$

repeat

The Cost, The Cost!!! Think about the cost…

- What's the cost of a gradient update step for LR???

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \sum_{j=1}^{N} x_i^j [y^j - \hat{P}(Y^j = 1 \mid \mathbf{x}^j, \mathbf{w}^{(t)})] \right\}$$

Learning Problems as Expectations

- Minimizing loss in training data:
  - Given dataset:
    - Sampled iid from some distribution p(**x**) on features:
  - Loss function, e.g., hinge loss, logistic loss,…
  - We often minimize loss in training data:

$$\ell_{\mathcal{D}}(\mathbf{w}) = \frac{1}{N} \sum_{j=1}^{N} \ell(\mathbf{w}, \mathbf{x}^j)$$

- However, we should really minimize expected loss on all data:

$$\ell(\mathbf{w}) = E_{\mathbf{x}}\left[\ell(\mathbf{w}, \mathbf{x})\right] = \int p(\mathbf{x}) \ell(\mathbf{w}, \mathbf{x}) d\mathbf{x}$$

- So, we are approximating the integral by the average on the training data

Gradient ascent in Terms of Expectations

- "True" objective function:

$$\ell(\mathbf{w}) = E_{\mathbf{x}}\left[\ell(\mathbf{w}, \mathbf{x})\right] = \int p(\mathbf{x})\ell(\mathbf{w}, \mathbf{x})d\mathbf{x}$$

- Taking the gradient:

$$\nabla_{\mathbf{w}} \ell(\mathbf{w}) = \nabla_{\mathbf{w}}\left[E_{\mathbf{x}}[\ell(\mathbf{w},\mathbf{x})]\right] = E_{\mathbf{x}}\left[\nabla_{\mathbf{w}} \ell(\mathbf{w},\mathbf{x})\right]$$

- "True" gradient ascent rule:

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \eta \, E_{\mathbf{x}}\left[\nabla_{\mathbf{w}} \ell(\mathbf{w},\mathbf{x})\right]$$

- How do we estimate expected gradient?

can't compute this

approx. with sample data

SGD: Stochastic Gradient Ascent (or Descent)

- "True" gradient: $\nabla \ell(\mathbf{w}) = E_{\mathbf{x}}\left[\nabla \ell(\mathbf{w}, \mathbf{x})\right]$
- Sample based approximation: take iid samples x^j

$$\nabla_{\mathbf{w}} \ell(\mathbf{w}) \approx \frac{1}{N} \sum_{j=1}^{N} \nabla_{\mathbf{w}} \ell(\mathbf{w}, \mathbf{x}^j)$$

$\widetilde{\nabla \ell}$

- What if we estimate gradient with just one sample???
  - Unbiased estimate of gradient $E[\widetilde{\nabla \ell}] = \nabla \ell$
  - Very noisy!
  - Called stochastic gradient ascent (or descent)
    - Among many other names
  - VERY useful in practice!!!

Stochastic Gradient Ascent for Logistic Regression

- Logistic loss as a stochastic function:

$$E_{\mathbf{x}}\left[\ell(\mathbf{w},\mathbf{x})\right] = E_{\mathbf{x}}\left[\ln P(y|\mathbf{x},\mathbf{w}) - \lambda||\mathbf{w}||_2^2\right]$$

- Batch gradient ascent updates:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta\left\{-\lambda w_i^{(t)} + \frac{1}{N}\sum_{j=1}^{N} x_i^{(j)}[y^{(j)} - P(Y=1|\mathbf{x}^{(j)},\mathbf{w}^{(t)})]\right\}$$

O(KN) →

- Stochastic gradient ascent updates:
  - Online setting:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta_t\left\{-\lambda w_i^{(t)} + x_i^{(t)}[y^{(t)} - P(Y=1|\mathbf{x}^{(t)},\mathbf{w}^{(t)})]\right\}$$

∀ i: i: feature or coordinate

take one data point at a time: $x^{(t)}$

same data point ∀ i

gradient update for just $x^{(t)}$

Stochastic Gradient Ascent: general case

- Given a stochastic function of parameters: $f(w) = E_x[f(w,x)]$
  - Want to find maximum

$w^* = \arg\max_w f(w) = \arg\max_w E_x[f(w,x)]$

- Start from $\mathbf{w}^{(0)}$ e.g. $w^{(0)} = 0$
- Repeat until convergence:
  - Get a sample data point $\mathbf{x}^t$
  - Update parameters:

$$w^{(t+1)} \leftarrow w^{(t)} + \eta \nabla f_w(w^{(t)}, x^t)$$

- Works on the online learning setting!
- Complexity of each gradient step is constant in number of examples!
- In general, step size changes with iterations

η decreases with t, e.g. $\eta_t = \frac{\alpha}{t}$ for $\alpha > 0$ constant

1 in your homework use constant η

What you should know…

- Classification: predict discrete classes rather than real values
- Logistic regression model: Linear model
  - Logistic function maps real values to [0,1]
- Optimize conditional likelihood
- Gradient computation
- Overfitting
- Regularization
- Regularized optimization
- Cost of gradient step is high, use stochastic gradient descent

Decision Trees

Machine Learning – CSE446

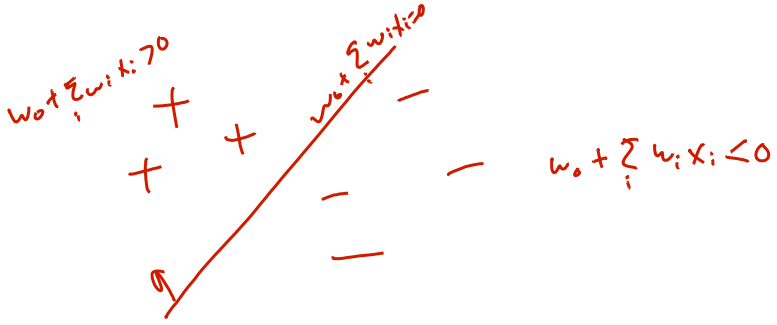
Carlos Guestrin

University of Washington

April 19, 2013

Linear separability

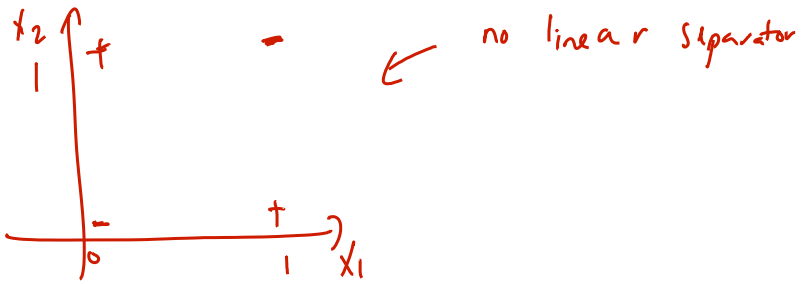
- A dataset is **linearly separable** iff there exists a **separating hyperplane**:
  - Exists **w**, such that:
    - $w_0 + \sum_i w_i x_i > 0$; if $\mathbf{x}=\{x_1,\ldots,x_k\}$ is a positive example
    - $w_0 + \sum_i w_i x_i < 0$; if $\mathbf{x}=\{x_1,\ldots,x_k\}$ is a negative example



Not linearly separable data

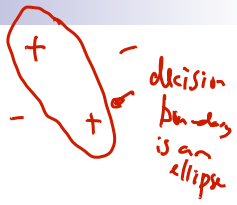
- Some datasets are **not linearly separable!**

$X_1 \oplus X_2$ $\quad X_1 \wedge \neg X_2 \vee \neg X_1 \wedge X_2$



Addressing non-linearly separable data – Option 1, non-linear features

- Choose non-linear features, e.g.,
  - Typical linear features: $w_0 + \sum_i w_i x_i$
  - Example of non-linear features:
    - Degree 2 polynomials, $w_0 + \sum_i w_i x_i + \sum_{ij} w_{ij} x_i x_j$
- Classifier $h_{\mathbf{w}}(\mathbf{x})$ still linear in parameters $\mathbf{w}$
  - As easy to learn ← convex/concave problem
  - Data is linearly separable in higher dimensional spaces
  - More discussion later this quarter



Addressing non-linearly separable data – Option 2, non-linear classifier

- Choose a classifier $h_{\mathbf{w}}(\mathbf{x})$ that is non-linear in parameters $\mathbf{w}$, e.g.,
  - Decision trees, boosting, nearest neighbor, neural networks…
- More general than linear classifiers
- But, can often be harder to learn (non-convex/concave optimization required)
- But, but, often very useful
- (BTW. Later this quarter, we'll see that these options are not that different)

A small dataset: Miles Per Gallon

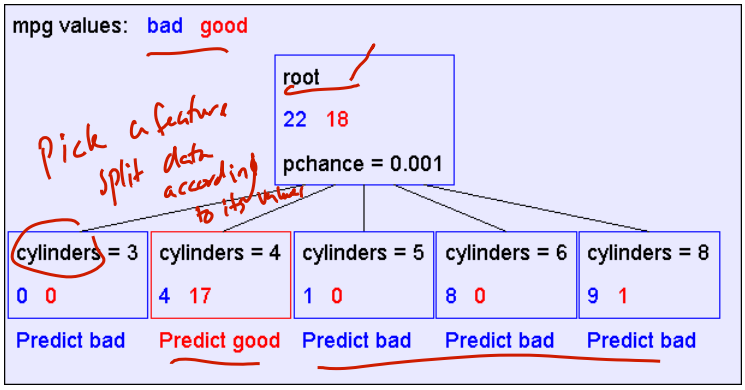
Suppose we want to predict MPG

mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europe
bad	5	medium	medium	medium	medium	75to78	europe

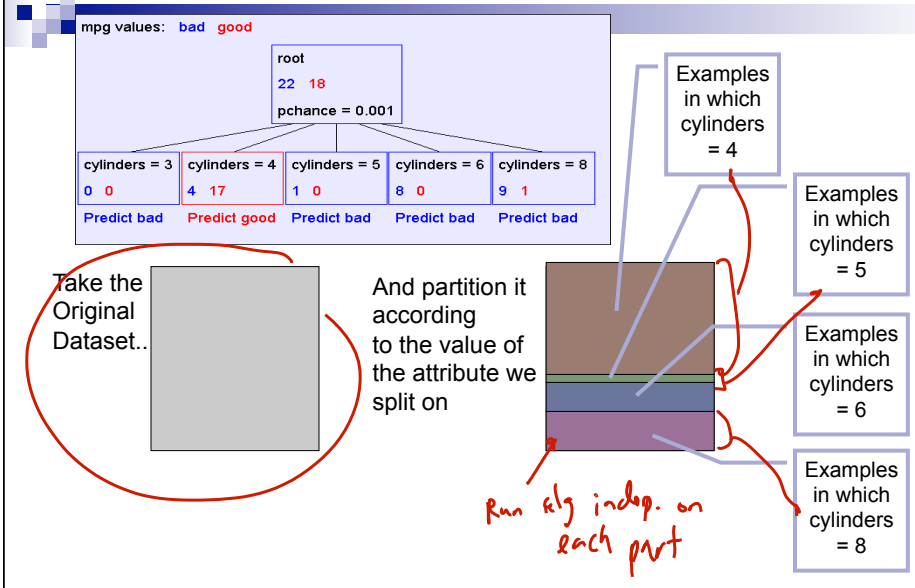
40 training examples

From the UCI repository (thanks to Ross Quinlan)

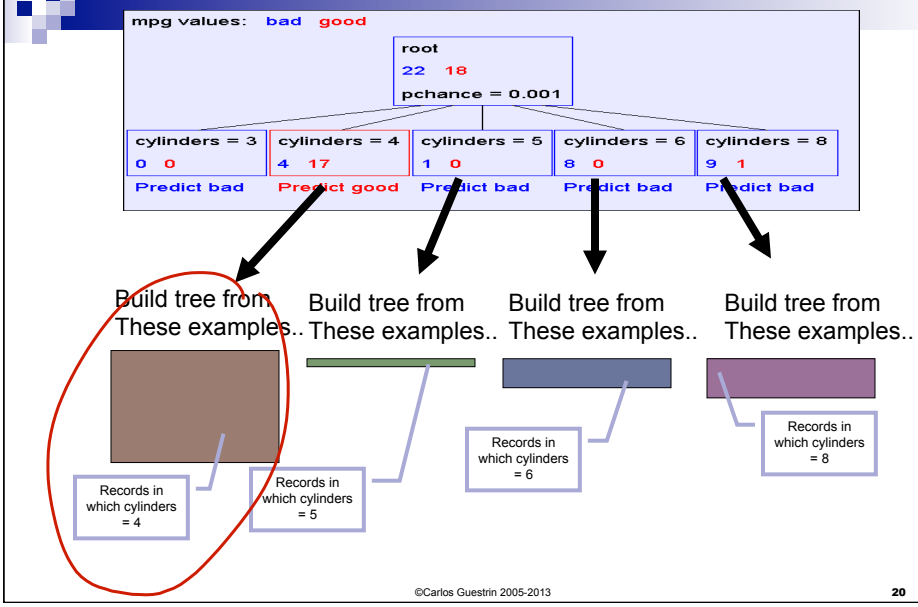
A Decision Stump



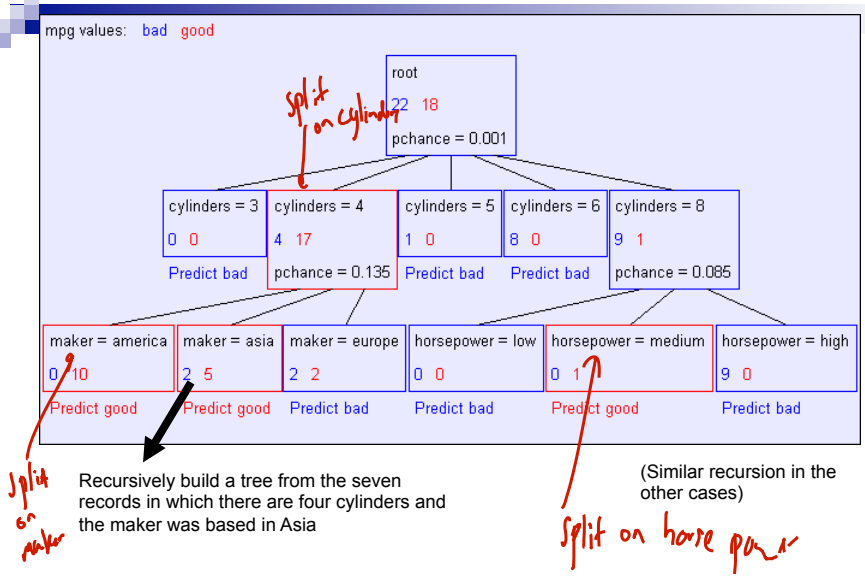
Recursion Step



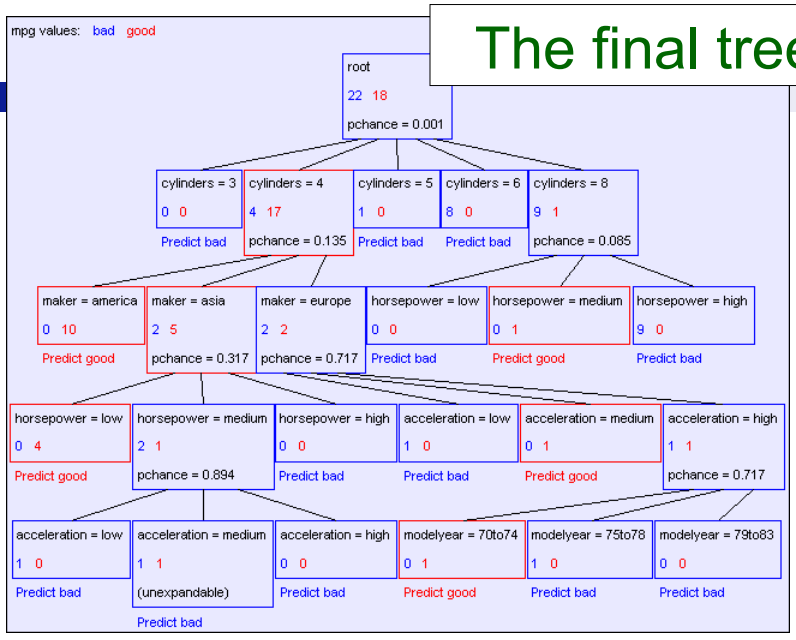
Recursion Step



Second level of tree

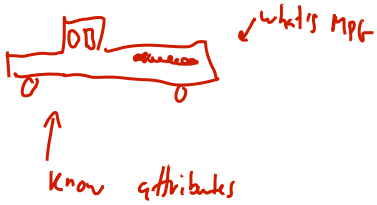


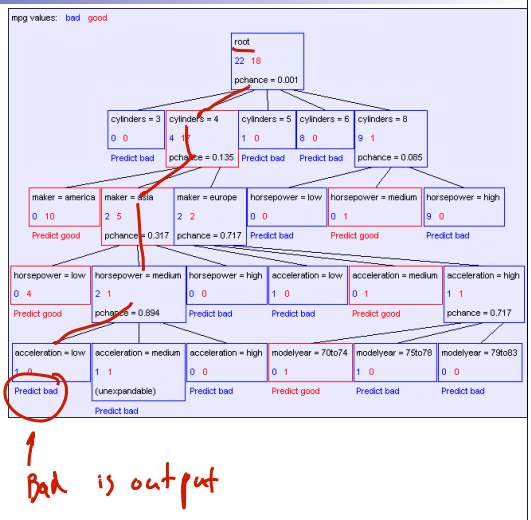
The final tree



Classification of a new example

- Classifying a test example – traverse tree and report leaf label

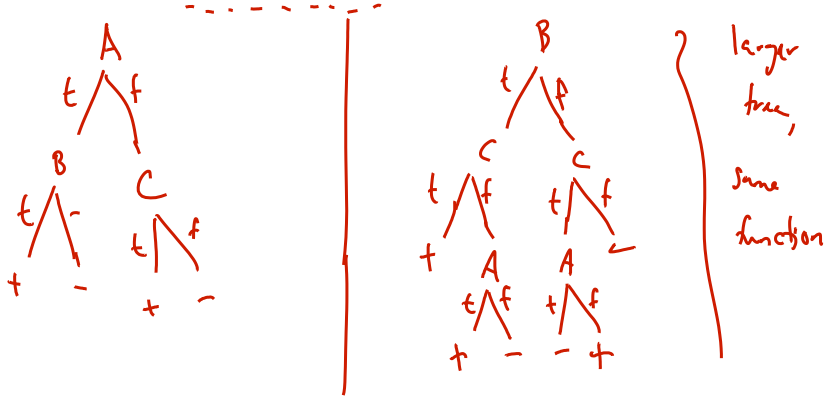




Bad is output

Are all decision trees equal?

- Many trees can represent the same concept
- But, not all trees will have the same size!
  - e.g., $\phi = A \wedge B \vee \neg A \wedge C$ ((A and B) or (not A and C))

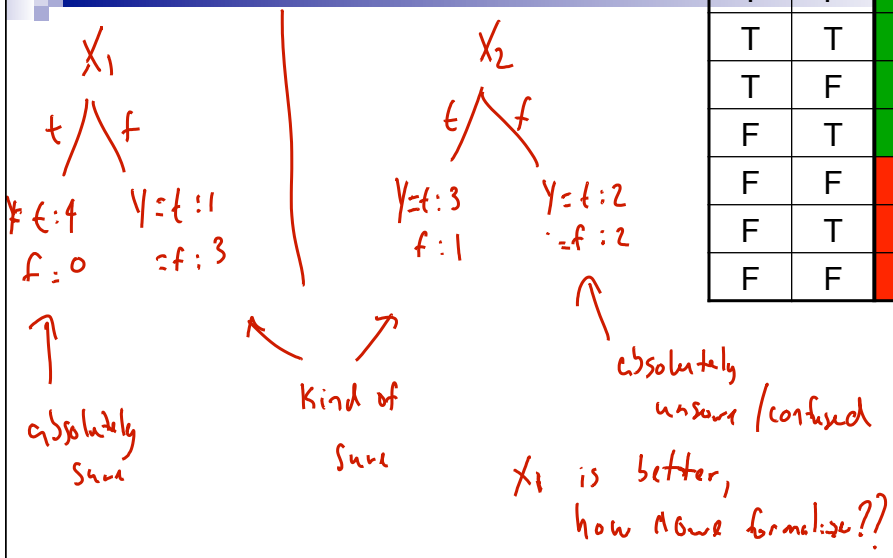


Learning decision trees is hard!!!

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest '76]
- Resort to a greedy heuristic:
  - Start from empty decision tree
  - Split on **next best attribute (feature)**
  - Recurse on subset of data associated with each value of the attribute

Choosing a good attribute

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F



Measuring uncertainty

- Good split if we are more certain about classification after split
  - Deterministic good (all true or all false)
  - Uniform distribution bad

P(Y=A) = 1/2	P(Y=B) = 1/4	P(Y=C) = 1/8	P(Y=D) = 1/8

P(Y=A) = 1/4	P(Y=B) = 1/4	P(Y=C) = 1/4	P(Y=D) = 1/4

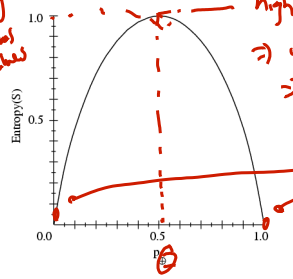
Entropy

Entropy $H(Y)$ of a random variable Y

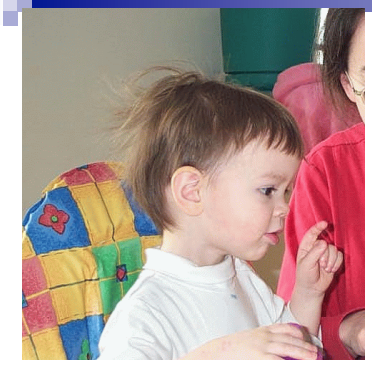
$$H(Y) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

More uncertainty, more entropy!

Information Theory interpretation: H(Y) is the expected number of bits needed to encode a randomly drawn value of Y (under most efficient code)



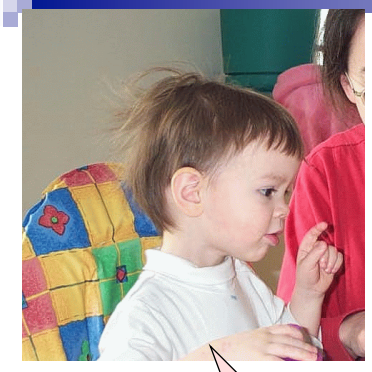
Andrew Moore's Entropy in a nutshell



Low Entropy

High Entropy

Andrew Moore's Entropy in a nutshell



Low Entropy

..the values (locations of soup) sampled entirely from within the soup bowl

High Entropy

..the values (locations of soup) unpredictable... almost uniformly sampled throughout our dining room

Information gain

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F

- Advantage of attribute – decrease in uncertainty
  - Entropy of Y before you split
  - Entropy after split
    - Weight by probability of following each branch, i.e., normalized number of records

$$H(Y \mid X) = -\sum_{j=1}^{v} P(X = x_j) \sum_{i=1}^{k} P(Y = y_i \mid X = x_j) \log_2 P(Y = y_i \mid X = x_j)$$

- Information gain is difference $IG(X) = H(Y) - H(Y \mid X)$