Online Learning

Instructor: Jesse Davis

Outline

- Motivation and Background
- Naïve Bayes
- Logistic Regression
- Perceptron
- Hoeffding Trees

Inductive Learning

- **Inductive learning** or **Prediction**:
  - **Given:** Data $S = \{(x_1, y_1),\ldots,(x_n, y_n)\}$
  - **Learn:** Function $F: x \rightarrow y$

- Discrete *y:* Classification
- Continuous y*:* Regression
- Probability*(X):* Probability estimation

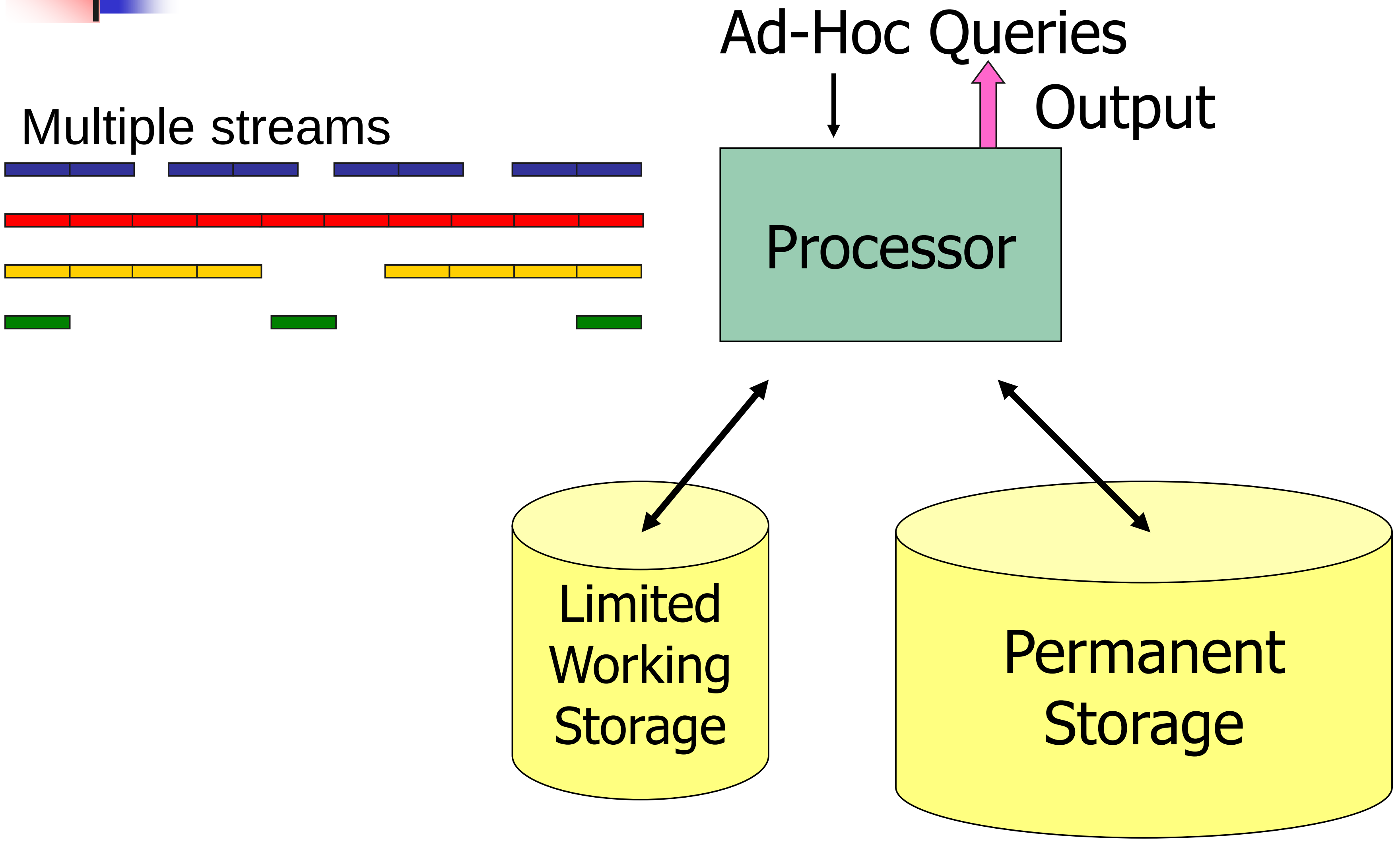
Standard Assumptions of Classical Machine Learning Algorithms

- Can access example whenever needed
- Data set is fixed
- Feature set is defined in advance
- Test and train examples are drawn from same distribution

Motivation I: Lots of Examples

- Many data sets contain a large number of examples
- Even if data fits in memory, it can be very slow to iterate over data multiple times

Motivation II: Streaming Data



Data Streams

- In a stream
  - Large (infinite?) number of examples
  - Data continuously arrives
  - Data may change over time
- Many applications characterized by this type of data
  - Sensor networks
  - Stock market
  - Etc.

Challenges with Data Streams

- Traditional algorithms rely on
  - (Random) Access to entire data set
  - Make multiple pass over the data
  - Much time per data item
- In data streams
  - Impossible to store all data
  - Random access is expensive
  - Streaming nature of data limits computation time per each example

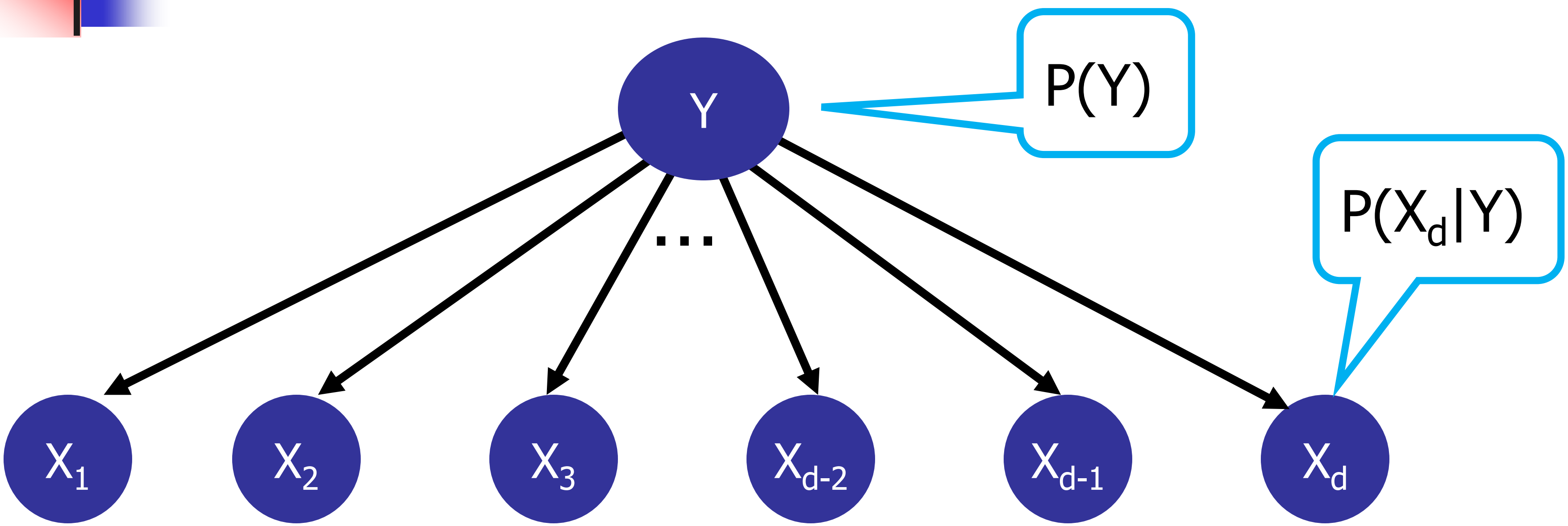
Notation

- Data $T = ((x_1, y_1),\ldots,(x_n,y_n))$
- Single example: x, y
- All examples: X, Y
- X has d dimensions
- Y has two values: 0, 1
- j ranges over examples
- i ranges over attributes

Batch vs. Online Learning

- Batch:
  - Receive all training examples at once
  - Learn model
  - Apply learned model to test examples
- Online
  - Receive examples one at a time
  - Apply model to example, receive true label
  - Update model

Naïve Bayes



$$\log \frac{P(Y_j=1 \mid x_j)}{P(Y_j=0 \mid x_j)} = \log \frac{P(Y=1)}{P(Y=0)} + \sum_i \log \frac{P(X_i = x_{j,i} \mid Y_j=1)}{P(X_i = x_{j,i} \mid Y_j=0)}$$

Training Naïve Bayes: Binary Classification

- Called estimating parameters
- Just involves estimating: $P(X_i \mid Y)$ and $P(Y)$
- Just involves computing counts in the data

Number of positive examples where feature X_i has value x_i

$$P(X_i = x_i \mid Y = \text{Positive}) = \frac{\#X_i = x_i \text{ when } Y = \text{Positive}}{\sum_{v \text{ is a value of } X_i} \#X_i = v \text{ when } Y = \text{Positive}}$$

$$P(Y = \text{Positive}) = \frac{\#\text{Positive Examples}}{\#\text{Positive Examples} + \#\text{Negative Examples}}$$

Example: Training Naïve Bayes

Color	Shape	Size	Class
Red	•	big	+
Blue	Δ	small	+
red	□	small	+
red	Δ	big	–
blue	•	small	–

P(red|+) = 2 / 3
P(blue|+) = 1 / 3
P(red|-) = 1 / 2
P(blue|-) = 1 / 2

Not ideal…

P(+) = 1 for any example with □

Zero probability

P(•|+) = 1 / 3
P(Δ|+) = 1 / 3
P(□|+) = 1 / 3

P(•|-) = 1 / 2
P(Δ|-) = 1 / 2
P(□|-) = 0 / 2

Laplace smoothing: All counts start at 1

Example: Training Naïve Bayes with Laplace Smoothing

Color	Shape	Size	Class
Red	•	big	+
Blue	Δ	small	+
red	□	small	+
red	Δ	big	–
blue	•	small	–

P(red|+) = 3 / 5
P(blue|+) = 2 / 5
P(red|-) = 2 / 4
P(blue|-) = 2 / 4

P(•|+) = 2 / 6
P(Δ|+) = 2 / 6
P(□|+) = 2 / 6

P(•|-) = 2 / 5
P(Δ|-) = 2 / 5
P(□|-) = 1 / 5

Naïve Bayes in Practice

- Empirically, estimates <u>relative</u> probabilities more reliably than absolute ones:

$$\frac{P(Pos \mid Features)}{P(Neg \mid Features)} = \frac{P(Features \mid Pos) * P(Pos)}{P(Features \mid Neg) * P(Neg)}$$

- Better than

$$P(Pos \mid Features) = P(Features \mid Pos) * P(Pos)$$

- Naïve Bayes tends to push probability estimates towards either 0 or 1

Training Naïve Bayes

ex = 0;

for each (x_j, y_j) do

    ex++;

    $p[y_j]$++;

    for each variable i do

        $c[y_j][i][x_{j,i}]$++;

Two important implementation details:
1) Always work with log
2) Avoid zero counts!

Inference in Naïve Bayes

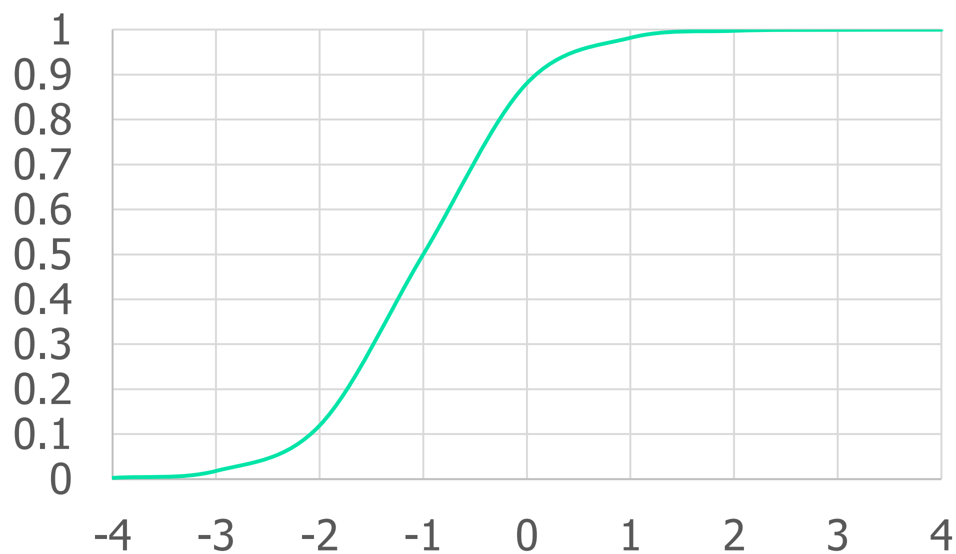
Color	Shape	Size	Category
blue	Δ	small	?

$$P(Y_j=1 \mid x_j) = \frac{P(Y=1) * P(\text{blue} \mid 1) * P(\Delta \mid 1) * P(\text{small} \mid 1)}{(P(Y=1) * P(\text{blue} \mid 1) * P(\Delta \mid 1) * P(\text{small} \mid 1) + P(Y=0) * P(\text{blue} \mid 0) * P(\Delta \mid 0) * P(\text{small} \mid 0))}$$

Logistic Regression

- Discriminative model for learning: $P(Y \mid X)$
- **Form:** Sigmoid applied to linear function of data

$$P(Y_j=0 \mid X_j, w) = \frac{1}{1 + \exp(w_0 + \Sigma_i \boxed{w_i}\, \boxed{x_{j,i}})}$$



Real value

Features can be continuous or discrete

Understanding Logistic Regression

$$P(Y=0 \mid X) = \frac{1}{1 + \exp(w_0 + \Sigma\, w_i x_i)}$$

$$P(Y=1 \mid X) = \frac{\exp(w_0 + \Sigma\, w_i x_i)}{1 + \exp(w_0 + \Sigma\, w_i x_i)}$$

$$\frac{P(Y=1 \mid X)}{P(Y=0 \mid X)} = \exp(w_0 + \Sigma\, w_i x_i)$$

$$\ln \frac{P(Y=1 \mid X)}{P(Y=0 \mid X)} = w_0 + \Sigma\, w_i x_i$$

Logistic Regression Training Task

- Given data: $D = ((x_1, y_1),\ldots,(x_n,y_n))$
- Find weights that maximize $P(Y \mid X)$

$$\text{Argmax}_w \; \Sigma_j \, P(Y_j \mid X_j, w)$$

The Objective Function: I

$$P(Y \mid X) = \Sigma_j^N P(Y_j \mid X_j)$$

$$= \Sigma_j^N \boxed{y_j \ln p_j} + (1 - y_j) \ln (1 - p_j)$$

$P(Y_j = 1 \mid X_j)$

$$y_j \ln \boxed{\frac{\exp(w_0 + \Sigma_i^d w_i x_{j,i})}{1 + \exp(w_0 + \Sigma_i^d w_i x_{j,i})}}$$

$$y_j \, [w_0 + \Sigma_i^d w_i x_{j,i} - \ln(1 + \exp(w_0 + \Sigma_i^d w_i x_{j,i}))]$$

The Objective Function: II

$$P(Y \mid X) = \Sigma_j^N P(Y_j \mid X_j)$$

$$= \Sigma_j^N y_j \ln p_j + \boxed{(1 - y_j) \ln (1 - p_j)}$$

$P(Y_j = 0 \mid X_j)$

$$(1 - y_j) \ln \boxed{\frac{1}{1 + \exp(w_0 + \Sigma_i w_i x_{j,i})}}$$

$$(y_j - 1) \ln (1 + \exp(w_0 + \Sigma_i w_i x_{j,i}))]$$

The Objective Function: Put It Together

$$P(Y \mid X) = \Sigma_j^N P(Y_j \mid X_j)$$

$$= \Sigma_j^N y_j \ln p_j + (1 - y_j) \ln (1 - p_j)$$

$$= \Sigma_j\, y_j\, [w_0 + \Sigma_i\, w_i x_{j,i} - \ln (1 + \exp(w_0 + \Sigma_i\, w_i x_{j,i}))]$$
$$+ (y_j - 1) \ln (1 + \exp(w_0 + \Sigma_i\, w_i x_{j,i}))$$

Underlined terms cancel

$$= \Sigma_j\, y_j\, (w_0 + \Sigma_i\, w_i x_{j,i}) \underline{- y_j \ln (1 + \exp(w_0 + \Sigma_i\, w_i x_{j,i}))}$$
$$\underline{+ \; y_j \ln (1 + \exp(w_0 + \Sigma_i\, w_i x_{j,i}))}$$
$$- \ln (1 + \exp(w_0 + \Sigma_i\, w_i x_{j,i}))$$

$$= \Sigma_j\, y_j\, (w_0 + \Sigma_i\, w_i x_{j,i}) - \ln (1 + \exp(w_0 + \Sigma_i\, w_i x_{j,i})$$

How Do We Select Weights?

- **Objective function:**

$$\text{Argmax}_w \Sigma_j\, y_j\, (w_0+\Sigma_i\, w_i x_{j,i}) - \ln\,(1+\exp(w_0 + \Sigma_i\, w_i x_{j,i}))$$

- **Good:** this function is concave
- **Bad:** no closed-form solution
- **Good:** Concave functions are easy to optimize

Gradient Ascent

- General optimization technique for concave functions

- **Gradient:** $\nabla_w l(w) = \left[\frac{\partial l(w)}{\partial w_0}, \ldots, \frac{\partial l(w)}{\partial w_d} \right]$

- **Update:** $w_{i+1} \longleftarrow w_i + \eta \frac{\partial l(w)}{\partial w_i}$

Note on Derivative

Easy to derive the gradient using:

1. Chain rule:

   let $F(f \circ g\ (x))$ then

   $F' = f'(g(x))\ g'(x)$

2. $$\frac{\partial \ln(x)}{\partial x} = \frac{1}{x}$$

3. $$\frac{\partial \exp(x)}{\partial x} = \exp(x)$$

Derivative of Loss Function

$$l(w) = \Sigma_j y_j (w_0+\Sigma_i w_i x_i) - \ln (1+\exp(w_0 + \Sigma_i w_i x_i))$$

$$\frac{\partial l(w)}{\partial w_i} = \Sigma_j y_j x_{j,i} - x_{j,i} \quad \frac{\exp(w_0 + \Sigma_i w_i x_i)}{1 + \exp(w_0 + \Sigma_i w_i x_i)}$$

$$\frac{\partial l(w)}{\partial w_i} = \Sigma_j x_{j,i} [y_j - P(Y_j = 1 \mid x_{j,i}, w)]$$

Putting It All Together

while (change < ε) do

$$w_0^{t+1} \leftarrow w_0^t + \eta \, \Sigma_j \, [y_j \; - P(Y_j = 1 \mid x_j, w)]$$

for each variable i do

$$w_i^{t+1} \leftarrow w_i^t + \eta \, \boxed{\Sigma_j} \, x_{j,i} \, [y_j \; - P(Y_j = 1 \mid x_j, w)]$$

Iterate over ALL examples: SLOW!!!

Idea: Stochastic Gradient

Approximate gradient with one example!!!

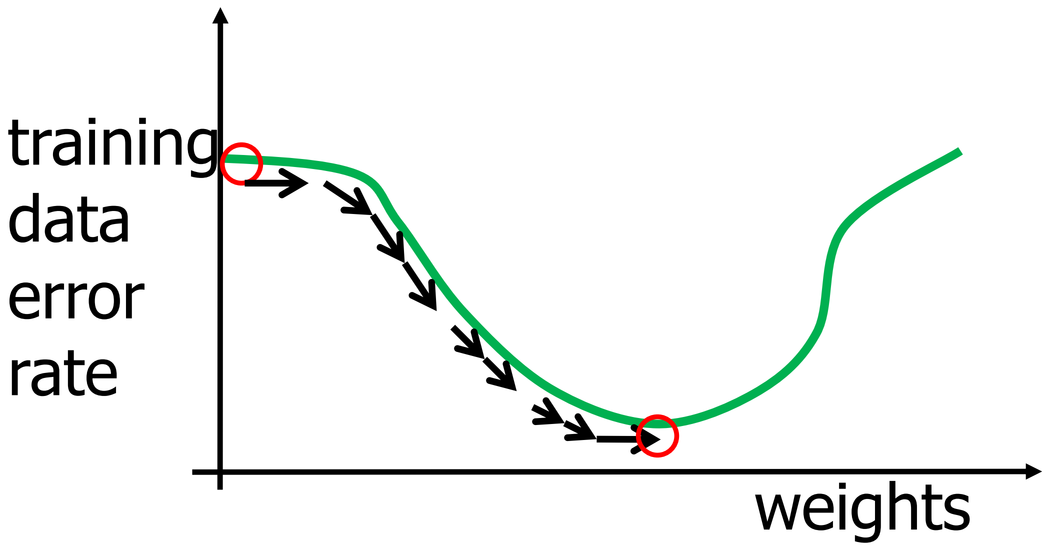
for each example j do

$$w_0^{t+1} \leftarrow w_0^t + \boxed{\eta\, [y_j - P(Y_j = 1 \mid x_j, w)]}$$

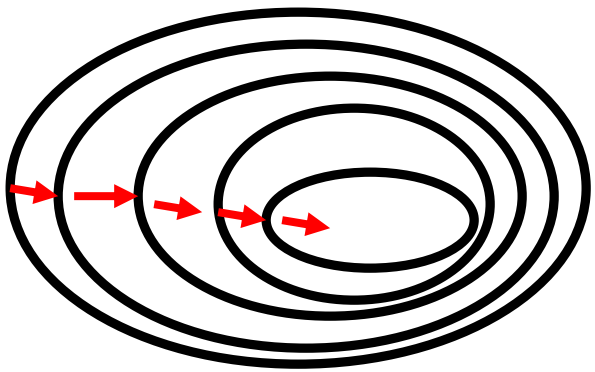
for each variable i do

$$w_i^{t+1} \leftarrow w_i^t + \eta\, x_{j,i}\, [y_j - P(Y_j = 1 \mid x_j, w)]$$

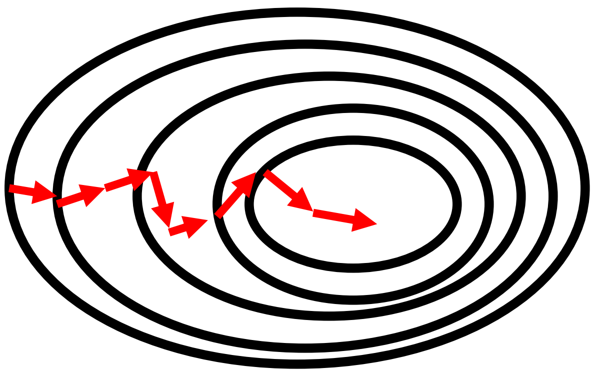
Gradient Descent Pictorially



Gradient Descent



Stochastic Gradient Descent



Issue: Overfitting

- Weights go to infinity, if data is linearly separable => overfitting ☹

L2 penalty: More on these in DM class

- Idea: Penalize high weights

$$\text{Argmax}_w \left[\Sigma_j \ln P(y_j \mid x_j)\right] - (\lambda/2)\|w\|_2^2$$

Conditional likelihood

$(\lambda/2)\ \Sigma_i\, w^2$

- Gradient update rule:

$$W_i^{t+1} \leftarrow w_i^t + \eta\ \{-\lambda\, w_i + x_{j,i}\ [y_j\ - P(Y_j = 1 \mid x_j, w)]\}$$

- Note: Do not regularize w_0

General Stochastic Gradient

Multiple iterations possible

Shuffle example order

while (change < ε) do

for each example j do

$$w_0^{t+1} \leftarrow w_0^t + \eta\, [y_j - P(Y_j = 1 \mid x_j, w)]$$

for each variable i do

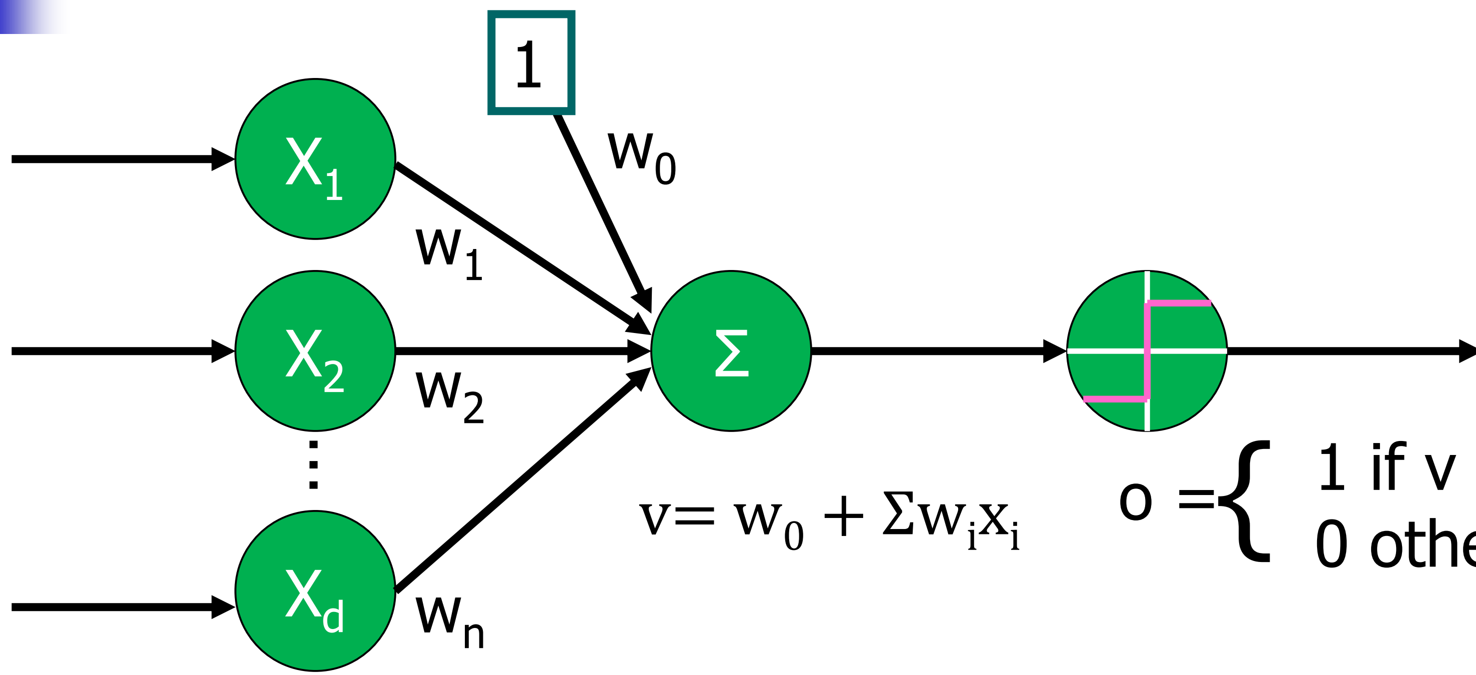
$$w_i^{t+1} \leftarrow w_i^t + \boxed{\eta}\, x_{j,i}\, [y_j - P(Y_j = 1 \mid x_j, w)]$$

Vary with each outer loop

Stochastic Gradient Details

- Each update: noisy estimate of gradient
- General technique that is widely used
  - Ascent for concave functions
  - Descent for convex functions
- Strong statistical theory behind algorithm
  - View loss function as expectation (average)
  - Theoretical guarantees on convergence

Perceptron



$$v = w_0 + \Sigma w_i x_i$$

$$o = \begin{cases} 1 \text{ if } v > 0 \\ 0 \text{ otherwise} \end{cases}$$

$$o(x_1,\ldots,x_N) = \begin{cases} 1 \text{ if } w_0 + w_1 x_1 + \ldots + w_n x_n > 0 \\ 0 \text{ otherwise} \end{cases}$$

Vector Notation

$$o(\vec{x}) = \begin{cases} 1 \text{ if } \vec{w} \cdot \vec{x} > 0 \\ -1 \text{ otherwise} \end{cases}$$

Perceptron Training

Two ways to do this:

- Perceptron rule which works in the unlikely case that the data is linearly separable

$$o = \begin{cases} 1 \text{ if } v > 0 \\ 0 \text{ otherwise} \end{cases}$$

- Delta rule which is most often used in practice

$$o = w_0 + \Sigma w_i x_i$$

$$E = \Sigma_j \, 0.5(y_j - o_j)^2$$

Unthresholded perceptron or linear unit

Online Learning for Perceptron (Delta Rule)

Set initial weights to random value

for each example (x_j, y_j) do

let o_j be the prediction with current weights

for each variable i do

$$w_i \leftarrow w_i + \eta \boxed{(y_j - o_j)} x_{j,i}$$

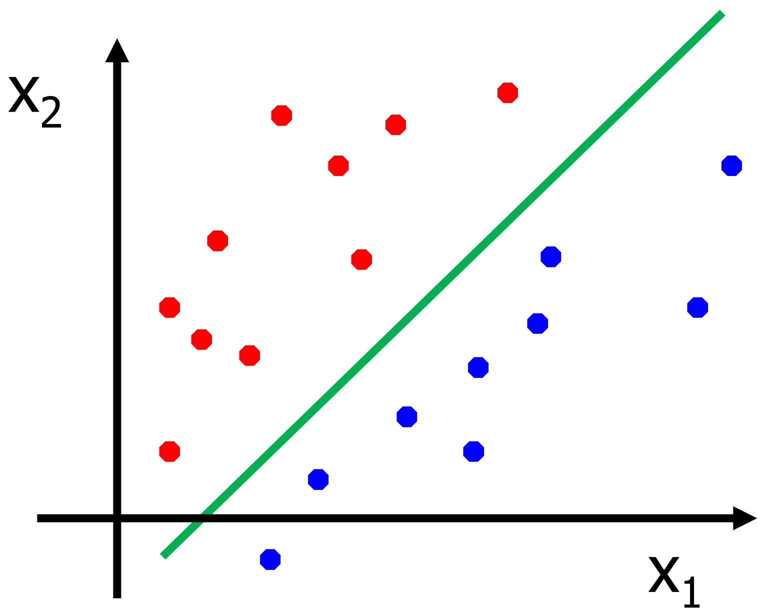
If label is correct, do nothing

If weights are too high, decrease them

If weights are too low, increase them

How Are These Algorithms Related?

- All are linear classifiers!
- Can only perfectly classify data if a hyperplane can separate positive and negative examples!



What Is The Difference?

The optimization function!

- Naïve Bayes: $l(\Theta) = \Sigma_j P(Y_j, X_j \mid \Theta)$

  Maximize likelihood

- Logistic Regression: $l(w) = \Sigma_j P(Y_j \mid X_j, w)$

  Maximize conditional likelihood

- Perceptron: $l(w) = ½ \Sigma_j (y_j\text{-}o_j)^2$

  Minimize squared error

Summary

- Despite all being linear, results can vary greatly among the three for a given dataset
  - Many equally good hypotheses
  - Each algorithm has different bias
- Always good to include linear baseline
- Linear classifiers more robust to concept drift
- All approaches generalize to multiclass setting

Outline

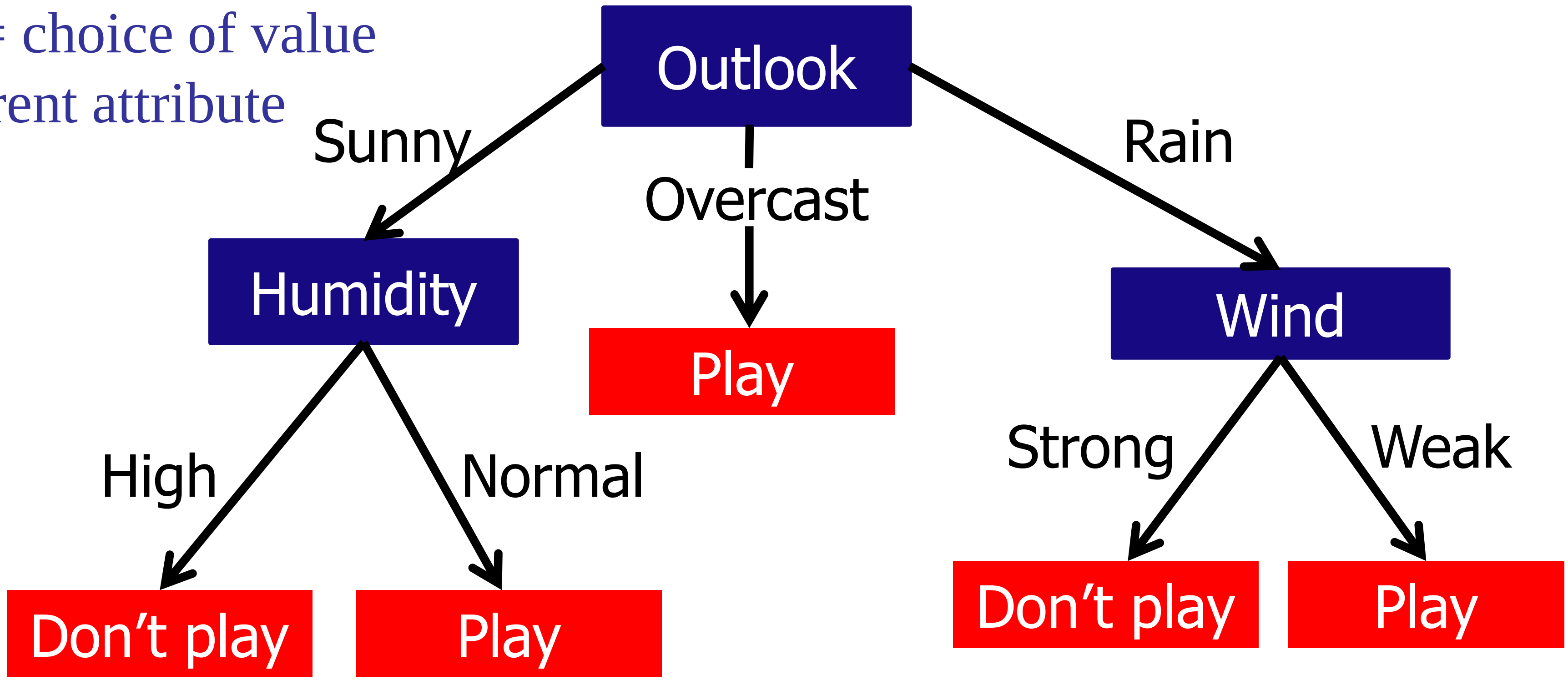
- Motivation and Background
- Naïve Bayes
- Logistic Regression
- Perceptron
- Hoeffding Trees

Decision Tree Representation

Good day for tennis?

Leaves = classification

Arcs = choice of value for parent attribute



Basic Decision Tree Algorithm

```
BuildTree(TraingData)
        Split(TrainingData)

Split(D)
        If (all points in D are of the same class)
                Then Return
        For each attribute A
                Evaluate splits on attribute A
                Use best split to partition D into D1, D2
                Split(D1)
                Split(D2)
```

Split Criteria: Information Gain

$$\text{Gain}(S,A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} (|S_v| / |S|)\ \text{Entropy}(S_v)$$

Where:

1) $\text{Entropy}(S) = -P \log_2(P) - N \log_2(N)$
2) P is proportion of positive example
3) N is proportion of negative examples

Challenge

- Question: Can you think of you to make a decision tree learner that only makes one pass over the data?
- Discuss in groups of 3-4 for five minutes

Very Fast Decision Tree: VFDT

- Goal: Process each example at most once in a small constant time
- Learns tree by considering a sample of the data at each node
  - Use first examples to pick root
  - Pass future example to second level and use them to select next attributes
- At each node, uses Hoeffding bound to select best attribute to split on

Hoeffding Bound

- For random variable X, with range R
- Given: n observations of X
- Compute empirical mean: $\overline{x}$
- Hoeffding bound: With probability 1-δ, the true mean of x is at least $\overline{x} \pm \varepsilon$ where
  - δ is a user defined parameter

$$\varepsilon = \sqrt{\frac{R^2 \ln(2/\delta)}{2n}}$$

- Independent of X's true distribution

Making Decisions

- $G(X_i)$ is the heuristic to pick attribute to split on
  - X_a: best attribute after n examples
  - X_b: second best attribute after n examples
  - $\Delta\overline{G} = G(X_a) - G(X_b)$
- If $\Delta\overline{G} > \varepsilon$, where ε is computed using the Hoeffding bound given a user provided δ
- Then true $\Delta G > \Delta\overline{G} - \varepsilon$ with probability $1 - \delta$
- If this holds, then split on X_a!

The Splitting Algorithm

- At each node, calculate $\Delta\overline{G}$ between the two best attributes
  - Need to calculate $G(X_i)$ for potential attribute at node
  - Pre-pruning: Can consider not splitting node
- If $\Delta\overline{G} > \varepsilon$ holds at a node, then split on the X_a
  - Create child nodes based on values of X_a
- If $\Delta\overline{G} < \varepsilon$, continue accumulating examples at this node

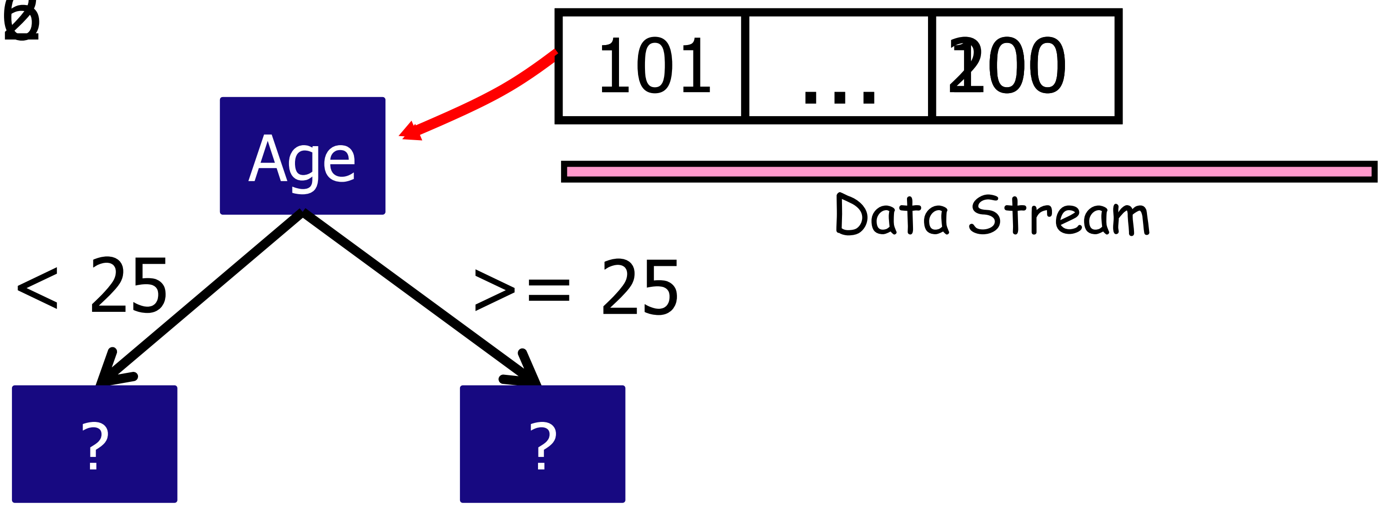
Algorithmic Details

- For efficiency, only recompute ΔG after seeing a pre-specified number of examples
- If ΔG is less than a user a threshold then split
  - If two attributes are similar it does not matter which one we split on
  - Don't waste time deciding between them
- Memory management
  - Dominated by storing sufficient statistics
  - Can deactivate leaves that are not promising

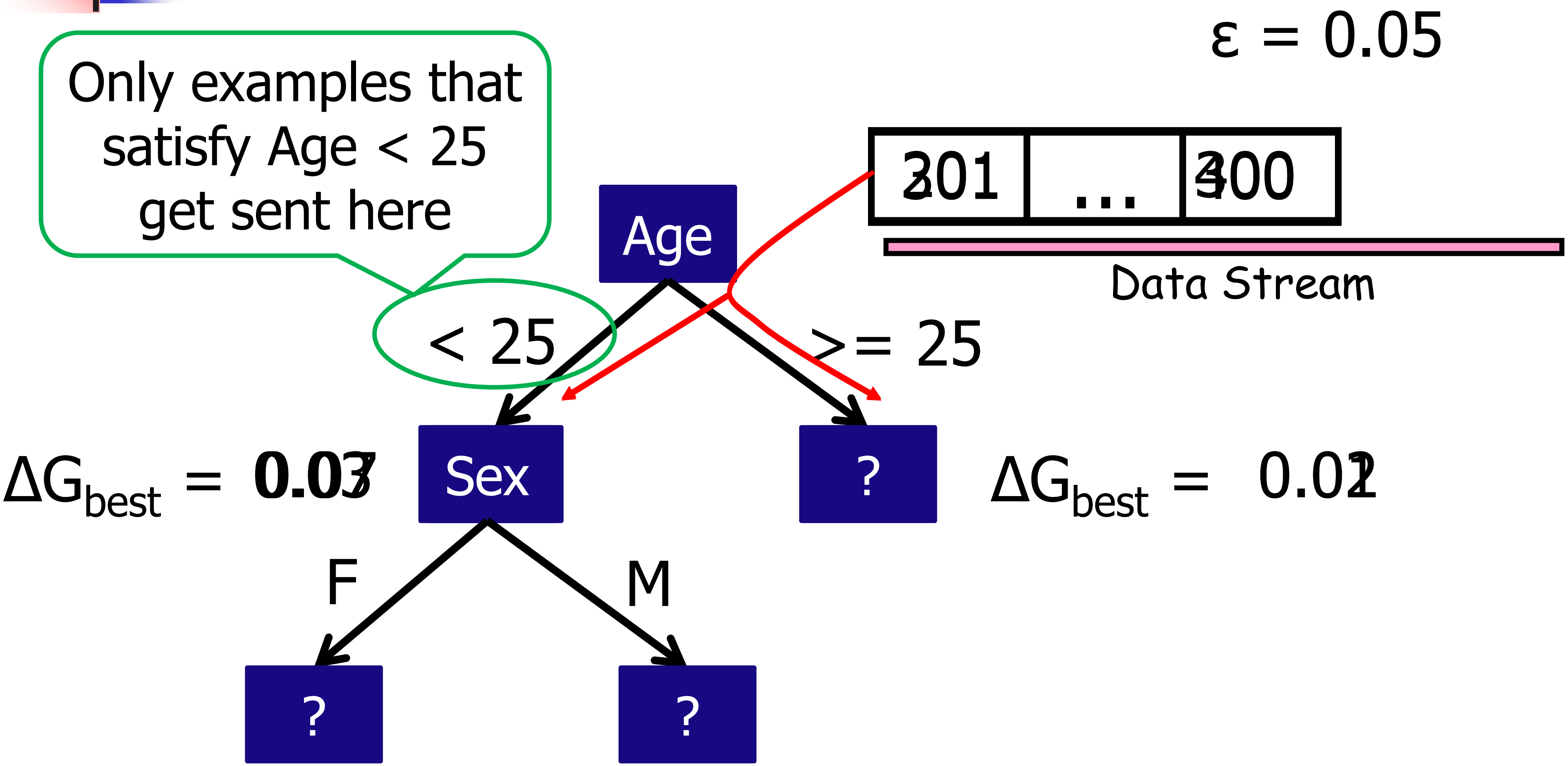
Pictorial Overview

$\varepsilon = 0.05$

$\Delta G_{best} = 0.02$

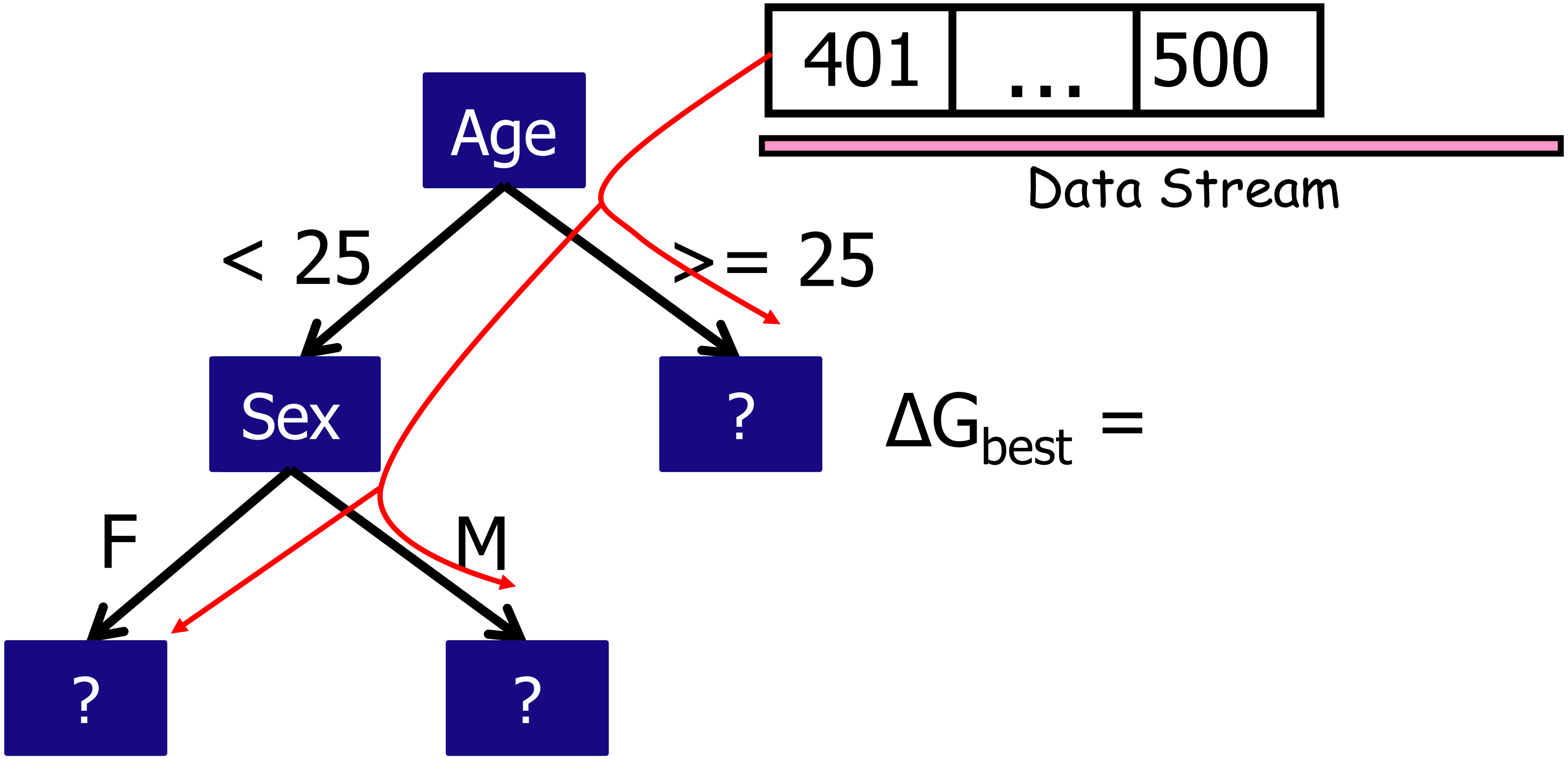


Pictorial Overview



Pictorial Overview

$\varepsilon = 0.05$



$\Delta G_{best} =$

$\Delta G_{best} =$

$\Delta G_{best} =$

VFDT Summary

- Scales better than pure memory-based or pure disk-based learners
  - Access data sequentially
  - Sampling potentially requires less than one scan of the data
- VFDT is incremental and anytime
  - Can easily incorporate new examples
  - Learned model after relatively few examples, which is progressively defined

Interlude: Spam Filtering

- Question: How would you tackle the problem of spam filtering for email?
  - What features would you use?
  - What model?
  - What challenges are there?

- Discuss quickly in groups

Spam Filtering Challenge: Changing Vocabulary

- Spam is inherently an adversarial setting
  - Filters: Improve and adopt models to catch new spam
  - Spammers: Change messages to defeat new models
  - Iterate
- Easiest change: modify words
  - Viagra, VIAgra, Vi8agra, etc.
  - Means feature space isn't fixed!!

Solution: Hashing!

- Original feature space: Words, tokens, etc.
- New feature space: hash values of features
- Transform example and represent it as a (multi)set of hash values

Simple Solution

From: Branded anti-ED Pills <otubu9068@telesp.net.br>
To: andrey.kolobov@gmail.com
Date: Fri, Apr 2, 2010 at 7:23 PM
Subject: Hot Sale, andrey.kolobov! 77% off on top goods Emen
Mailed-by: telesp.net.br

Why	aren't	you	on	our	site	andrey.kolobov	We	have	off	today

Bag of Words:

are	why	site	buy	viagra	…	have
0	1	1	0	0	…	1

Simple hashing:
1. Select hash function h with range [0, p-1]
2. Hash each token t and set v[h(t)] = 1

H(why) = 10
H(aren't) = 4
…
H(today) = 9

0	1	2	3	4	5	6	7	8	9	…	p-1
0	0	0	0	1	0	0	0	0	1	…	0

Advice on Assignment

- Follow the instructions!
- Start working on this early
- Carefully test your algorithm
- Critically think about what you can put in the report and make you sure you structure it well

Advice on Reports: High-Level

- Think about the structure of a report: Sections, subsections, etc.

- Make sure graphs, tables are good
  - Fonts big enough
  - Axes labeled
  - Titles meaningful
  - Key does not overlap with lines

Advice on Reports: Experiments

- Pose experimental questions you will address, state how you will answer them
- State all relevant parameter settings
- Good to explore the space of parameters: They greatly effect performance!!
- Employ good machine learning methodology

Questions?