Linear classifiers:
Scaling up learning via SGD

Stochastic gradient descent:
Learning, one data point at a time
Stochastic gradient ascent

\[ w^{(t)} \rightarrow w^{(t+1)} \rightarrow w^{(t+2)} \rightarrow w^{(t+3)} \rightarrow w^{(t+4)} \]

- **Update coefficients**
- **Use only small subsets of data**
- **Compute gradient**
- **Many updates for each pass over data**

---

Stochastic gradient ascent for logistic regression

\[ \text{init } w^{(1)} = 0, \ t = 1 \]

until converged

\[ \begin{align*}
\text{for } j &= 0, \ldots, D \\
\text{partial}[j] &= \sum_{i=1}^{N} \left( h_j(x_i) \left( \mathbb{1}[y_i = +1] - P(y = +1 | x_i, w^{(t)}) \right) \right) \\
\text{w}_j^{(t+1)} &\leftarrow w_j^{(t)} + \eta \text{ partial}[j] \\
t &\leftarrow t + 1
\end{align*} \]

- **Each time, pick different data point } i \)
Why would stochastic gradient ever work???

Gradient is direction of steepest ascent

Gradient is “best” direction, but any direction that goes “up” would be useful
In ML, steepest direction is sum of “little directions” from each data point

\[ \nabla \ell(w) = \sum_{i=1}^{N} \frac{\partial \ell_i(w)}{\partial w_j} \]

For most data points, contribution points “up”

Stochastic gradient:
Pick a data point and move in direction

\[ \frac{\partial \ell(w)}{\partial w_j} \approx \frac{\partial \ell_i(w)}{\partial w_j} \]

Most of the time, total likelihood will increase
Stochastic gradient ascent:
Most iterations increase likelihood, but
sometimes decrease it ➔
On average, make progress

until converged
for i=1,…,N
for j=0,…,D

\[ w_j^{(t+1)} \leftarrow w_j^{(t)} + \eta \frac{\partial \ell_i(w)}{\partial w_j} \]

\[ t \leftarrow t + 1 \]
Convergence paths

- Gradient
- Stochastic gradient

Stochastic gradient convergence is “noisy”

- Stochastic gradient makes “noisy” progress
- Gradient usually increases likelihood smoothly
- Total time proportional to # passes over data
- Stochastic gradient achieves higher likelihood sooner, but it’s noisier

Better Avg. log likelihood

Average log likelihood vs. # of passes over dataset.
Eventually, gradient catches up

Note: should only trust “average” quality of stochastic gradient

The last coefficients may be really good or really bad!! 😞

Stochastic gradient will eventually oscillate around a solution

Minimize noise: don’t return last learned coefficients

Output average:

\[ \hat{w} = \frac{1}{T} \sum_{t=1}^{T} w^{(t)} \]
Summary of why stochastic gradient works

- Gradient finds direction of steepest ascent
- Gradient is sum of contributions from each data point
- Stochastic gradient uses direction from 1 data point
- On average increases likelihood, sometimes decreases
- Stochastic gradient has “noisy” convergence

Online learning: Fitting models from streaming data
Batch vs online learning

**Batch learning**
- All data is available at start of training time

**Online learning**
- Data arrives (streams in) over time
  - Must train model as data arrives!

---

**Online learning example: Ad targeting**

Input: \( x_t \)
- User info, page text

Website
- Ad1
- Ad2
- Ad3

\( \hat{y} = \text{Suggested ads} \)

User clicked on Ad2

\( y_t = \text{Ad2} \)

ML algorithm

\( \hat{W}^{(t)} \) → \( \hat{W}^{(t+1)} \)
Online learning problem

• Data arrives over each time step $t$:
  - Observe input $x_t$
    • Info of user, text of webpage
  - Make a prediction $\hat{y}_t$
    • Which ad to show
  - Observe true output $y_t$
    • Which ad user clicked on

Need ML algorithm to update coefficients each time step!

Stochastic gradient ascent can be used for online learning!!!

• init $w^{(1)}=0$, $t=1$
• Each time step $t$:
  - Observe input $x_t$
  - Make a prediction $\hat{y}_t$
  - Observe true output $y_t$
  - Update coefficients:
  
  for $j=0,\ldots,D$
  
  $w_j^{(t+1)} \leftarrow w_j^{(t)} + \eta \frac{\partial \ell_t(w)}{\partial w_j}$
Summary of online learning

Data arrives over time

Must make a prediction every time new data point arrives

Observe true class after prediction made

Want to update parameters immediately

Summary of stochastic gradient descent
What you can do now...

- Significantly speedup learning algorithm using stochastic gradient
- Describe intuition behind why stochastic gradient works
- Apply stochastic gradient in practice
- Describe online learning problems
- Relate stochastic gradient to online learning
Predicting potential loan defaults

What makes a loan risky?

I want to buy a new house!

Credit History ★★★★★
Income ★★★
Term ★★★★★
Personal Info ★★★
Credit history explained

Did I pay previous loans on time?

**Example:** excellent, good, or fair

Credit History ★★★★★

Income ★★★

Term ★★★★★

Personal Info ★★★

Income

What’s my income?

**Example:** $80K per year

Credit History ★★★★★

Income ★★★

Term ★★★★★

Personal Info ★★★
Loan terms

How soon do I need to pay the loan?

**Example:** 3 years, 5 years,…

Personal information

Age, reason for the loan, marital status,…

**Example:** Home loan for a married couple
Intelligent application

Loan Applications

Intelligent loan application review system

Safe ✓

Risky ✗

Risky ✗

Classifier review

Loan Application

Classifier MODEL

Input: \( x_i \)

Output: Predicted class

\( \hat{y}_i = +1 \) Safe

\( \hat{y}_i = -1 \) Risky
This module ... decision trees

Scoring a loan application

\[ x_i = (\text{Credit} = \text{poor}, \text{Income} = \text{high}, \text{Term} = 5 \text{ years}) \]
Decision tree learning task

Decision tree learning problem

Training data: \( N \) observations \((x_i, y_i)\)

<table>
<thead>
<tr>
<th>Credit</th>
<th>Term</th>
<th>Income</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>excellent</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>5 yrs</td>
<td>low</td>
<td>risky</td>
</tr>
<tr>
<td>fair</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
<tr>
<td>poor</td>
<td>5 yrs</td>
<td>high</td>
<td>risky</td>
</tr>
<tr>
<td>excellent</td>
<td>3 yrs</td>
<td>low</td>
<td>risky</td>
</tr>
<tr>
<td>fair</td>
<td>5 yrs</td>
<td>low</td>
<td>safe</td>
</tr>
<tr>
<td>poor</td>
<td>3 yrs</td>
<td>high</td>
<td>risky</td>
</tr>
<tr>
<td>poor</td>
<td>5 yrs</td>
<td>low</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
</tbody>
</table>
Quality metric: Classification error

- Error measures fraction of mistakes

\[
\text{Error} = \frac{\text{# incorrect predictions}}{\text{# examples}}
\]

- Best possible value: 0.0
- Worst possible value: 1.0

How do we find the best tree?

Exponentially large number of possible trees makes decision tree learning hard!

Learning the smallest decision tree is an NP-hard problem [Hyafil & Rivest '76]
Greedy decision tree learning

Our training data table

Assume $N = 40$, 3 features

<table>
<thead>
<tr>
<th>Credit</th>
<th>Term</th>
<th>Income</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>excellent</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>5 yrs</td>
<td>low</td>
<td>risky</td>
</tr>
<tr>
<td>fair</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
<tr>
<td>poor</td>
<td>5 yrs</td>
<td>high</td>
<td>risky</td>
</tr>
<tr>
<td>excellent</td>
<td>3 yrs</td>
<td>low</td>
<td>risky</td>
</tr>
<tr>
<td>fair</td>
<td>5 yrs</td>
<td>low</td>
<td>safe</td>
</tr>
<tr>
<td>poor</td>
<td>3 yrs</td>
<td>high</td>
<td>risky</td>
</tr>
<tr>
<td>poor</td>
<td>5 yrs</td>
<td>low</td>
<td>safe</td>
</tr>
<tr>
<td>fair</td>
<td>3 yrs</td>
<td>high</td>
<td>safe</td>
</tr>
</tbody>
</table>
Start with all the data

Loan status:  
Safe  Risky

(all data)  

22 18  

# of Risky loans

# of Safe loans

N = 40 examples

Compact visual notation: Root node

Loan status:  
Safe  Risky

Root  

22 18  

# of Risky loans

# of Safe loans

N = 40 examples
Decision stump: Single level tree

Loan status: Safe Risky

Root

Split on Credit

Credit?

excellent

fair

poor

Subset of data with Credit = excellent

Subset of data with Credit = fair

Subset of data with Credit = poor

Visual notation: Intermediate nodes

Loan status: Safe Risky

Root

Credit?

excellent

fair

poor

Intermediate nodes
Making predictions with a decision stump

For each intermediate node, set $\hat{y} =$ majority value

Selecting best feature to split on
How do we learn a decision stump?

Loan status: Safe Risky

Root
22 18

Find the “best” feature to split on!

Credit?

excellent
9 0

fair
9 4

poor
4 14

How do we select the best feature?

Choice 1: Split on Credit

Loan status: Safe Risky

Root
22 18

Credit?

excellent
9 0

fair
9 4

poor
4 14

Choice 2: Split on Term

Loan status: Safe Risky

Root
22 18

Term?

3 years
16 4

5 years
6 14
How do we measure effectiveness of a split?

Idea: Calculate classification error of this decision stump

Error = \frac{\text{# mistakes}}{\text{# data points}}

Loan status:
Safe  Risky

Credit?

excellent  9  0
fair  9  4
poor  4  14

Idea: Calculate classification error of this decision stump

Root
22 18

Error = \frac{18}{22 + 18} = 0.45

Calculating classification error

• Step 1: \( \hat{y} \) = class of majority of data in node
• Step 2: Calculate classification error of predicting \( \hat{y} \) for this data

\( \hat{y} \) = majority class

Tree  Classification error
(root)  0.45
Choice 1: Split on Credit history?

Choice 1: Split on Credit

Loan status: Safe Risky

Root: 22 18

Does a split on Credit reduce classification error below 0.45?

Credit?

excellent 9 0

fair 9 4

poor 4 14

Split on Credit: Classification error

Choice 1: Split on Credit

Loan status: Safe Risky

Root: 22 18

Credit?

excellent

fair

poor

Safe

Safe

Risky

0 mistakes 4 mistakes 4 mistakes

Error = \frac{0 + 4 + 4}{40} = 0.2

<table>
<thead>
<tr>
<th>Tree</th>
<th>Classification error</th>
</tr>
</thead>
<tbody>
<tr>
<td>(root)</td>
<td>0.45</td>
</tr>
<tr>
<td>Split on credit</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Choice 2: Split on Term?

Choice 2: Split on Term

Loan status: Safe Risky

Root 22 18

Term?

3 years 16 4

Safe

5 years 6 14

Risky

Evaluating the split on Term

Choice 2: Split on Term

Loan status: Safe Risky

Root 22 18

Term?

3 years 16 4

Safe

4 mistakes

5 years 6 14

Risky

6 mistakes

Error = \frac{4 + 6}{4 + 6} = \frac{10}{10} = 0.25

<table>
<thead>
<tr>
<th>Tree</th>
<th>Classification error</th>
</tr>
</thead>
<tbody>
<tr>
<td>(root)</td>
<td>0.45</td>
</tr>
<tr>
<td>Split on credit</td>
<td>0.2</td>
</tr>
<tr>
<td>Split on term</td>
<td>0.25</td>
</tr>
</tbody>
</table>
Choice 1 vs Choice 2: Comparing split on Credit vs Term

<table>
<thead>
<tr>
<th>Tree</th>
<th>Classification error</th>
</tr>
</thead>
<tbody>
<tr>
<td>(root)</td>
<td>0.45</td>
</tr>
<tr>
<td>split on credit</td>
<td>0.2</td>
</tr>
<tr>
<td>split on loan term</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Choice 1: Split on Credit

Choice 2: Split on Term

Feature split selection algorithm

- Given a subset of data $M$ (a node in a tree)
- For each feature $h_i(x)$:
  1. Split data of $M$ according to feature $h_i(x)$
  2. Compute classification error split
- Chose feature $h^*(x)$ with lowest classification error
Recursion & Stopping conditions

We’ve learned a decision stump, what next?

Loan status: Safe Risky

Root  

excellent  

fair  

poor  

Credit?  

All data points are Safe ➔ nothing else to do with this subset of data
Tree learning = Recursive stump learning

Loan status: Safe Risky

Root

Credit?

excellent

fair

poor

Safe

Build decision stump with subset of data where Credit = fair

Build decision stump with subset of data where Credit = poor

Second level

Loan status: Safe Risky

Root

Credit?

excellent

fair

poor

3 years

5 years

Income?

high

Low

Safe

Risky

Build another stump these data points

Risky
Final decision tree

Loan status:
- Safe
- Risky

Credit?
- excellent
  - Safe
- Fair
  - Credit?
    - high
      - income?
        - low
          - Risky
        - high
          - Term?
            - 3 years
              - Risky
            - 5 years
              - Safe
          - safe
        - low
          - Term?
            - 3 years
              - Risky
            - 5 years
              - Safe
          - safe

Simple greedy decision tree learning

Pick best feature to split on

Learn decision stump with this split

For each leaf of decision stump, recurse

When do we stop???
Stopping condition 1: All data agrees on y

All data in these nodes have same y value ➔ Nothing to do

Stopping condition 2: Already split on all features

Already split on all possible features ➔ Nothing to do
Greedy decision tree learning

- **Step 1**: Start with an empty tree
  
- **Step 2**: Select a feature to split data
  
- For each split of the tree:
  - **Step 3**: If nothing more to, make predictions
  - **Step 4**: Otherwise, go to Step 2 & continue (recurse) on this split

Is this a good idea?

Proposed stopping condition 3:
Stop if no split reduces the classification error
Stopping condition 3: Don’t stop if error doesn’t decrease???

\[ y = x[1] \text{xor} x[2] \]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

\[ \hat{y} = \text{safe} \]

\[ \text{Error} = \frac{2}{4} = 0.5 \]

Consider split on \( x[1] \)

\[ y = x[1] \text{xor} x[2] \]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

\[ \text{Tree} \quad \text{Classification error} \]

| Root (root) | 0.5 |

\[ \text{Error} = \frac{1+1}{4} = 0.5 \]

<table>
<thead>
<tr>
<th>Tree</th>
<th>Classification error</th>
</tr>
</thead>
<tbody>
<tr>
<td>(root)</td>
<td>0.5</td>
</tr>
<tr>
<td>Split on x[1]</td>
<td>0.5</td>
</tr>
</tbody>
</table>
Consider split on $x[2]$

$y = x[1] \oplus x[2]$

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

$y$ values

<table>
<thead>
<tr>
<th>$x[2]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
</tr>
<tr>
<td>False</td>
</tr>
</tbody>
</table>

Root

2

2

Error = $\frac{1+1}{2+2} = 0.5$

Neither features improve training error...

Stop now???

Tree Classification error

<table>
<thead>
<tr>
<th>Tree</th>
<th>Classification error</th>
</tr>
</thead>
<tbody>
<tr>
<td>(root)</td>
<td>0.5</td>
</tr>
<tr>
<td>Split on $x[1]$</td>
<td>0.5</td>
</tr>
<tr>
<td>Split on $x[2]$</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Final tree with stopping condition 3

$y = x[1] \oplus x[2]$

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$y$ values</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
</tr>
<tr>
<td>False</td>
</tr>
</tbody>
</table>

Root

2

2

Predict True

Tree Classification error

<table>
<thead>
<tr>
<th>Tree</th>
<th>Classification error</th>
</tr>
</thead>
<tbody>
<tr>
<td>with stopping condition 3</td>
<td>0.5</td>
</tr>
</tbody>
</table>
Without stopping condition 3

**Condition 3** (stopping when training error doesn't improve) is not recommended!

\[ y = x[1] \text{ xor } x[2] \]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

**y values**

<table>
<thead>
<tr>
<th>True</th>
<th>False</th>
</tr>
</thead>
</table>

**Tree**

<table>
<thead>
<tr>
<th>Classification error</th>
</tr>
</thead>
<tbody>
<tr>
<td>with stopping condition 3</td>
</tr>
<tr>
<td>without stopping condition 3</td>
</tr>
</tbody>
</table>

Decision tree learning:
Real valued features
How do we use real values inputs?

<table>
<thead>
<tr>
<th>Income</th>
<th>Credit</th>
<th>Term</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>$105 K</td>
<td>excellent</td>
<td>3 yrs</td>
<td>Safe</td>
</tr>
<tr>
<td>$112 K</td>
<td>good</td>
<td>5 yrs</td>
<td>Risky</td>
</tr>
<tr>
<td>$73 K</td>
<td>fair</td>
<td>3 yrs</td>
<td>Safe</td>
</tr>
<tr>
<td>$69 K</td>
<td>excellent</td>
<td>5 yrs</td>
<td>Safe</td>
</tr>
<tr>
<td>$217 K</td>
<td>excellent</td>
<td>3 yrs</td>
<td>Risky</td>
</tr>
<tr>
<td>$120 K</td>
<td>good</td>
<td>5 yrs</td>
<td>Safe</td>
</tr>
<tr>
<td>$64 K</td>
<td>fair</td>
<td>3 yrs</td>
<td>Risky</td>
</tr>
<tr>
<td>$340 K</td>
<td>excellent</td>
<td>5 yrs</td>
<td>Safe</td>
</tr>
<tr>
<td>$60 K</td>
<td>good</td>
<td>3 yrs</td>
<td>Risky</td>
</tr>
</tbody>
</table>

Threshold split

Loan status: Safe Risky

Split on the feature Income

Income?

Subset of data with Income $\geq$ $60K$

Split on the feature Income

Income?

subset of data with Income $\geq$ $60K$
Finding the best threshold split

Infinite possible values of \( t \)

\[ \text{Income} = t^* \]

Income < \( t^* \)  
Income \( \geq t^* \)

Consider a threshold between points

Same classification error for any threshold split between \( v_A \) and \( v_B \)
Only need to consider mid-points

Finite number of splits to consider

Threshold split selection algorithm

- **Step 1**: Sort the values of a feature $h_j(x)$:
  
  Let $\{v_1, v_2, v_3, \ldots, v_N\}$ denote sorted values

- **Step 2**:
  - For $i = 1 \ldots N - 1$
    - Consider split $t_i = (v_i + v_{i+1}) / 2$
    - Compute classification error for threshold split $h_j(x) \geq t_i$
  - Chose the $t^*$ with the lowest classification error
Visualizing the threshold split

Threshold split is the line \( \text{Age} = 38 \)

Split on Age \( \geq 38 \)

\( \text{Predict Risky} \)

\( \text{Predict Safe} \)
Depth 2: Split on Income $\geq$ $60K$

Threshold split is the line \( \text{Income} = 60K \)

Each split partitions the 2-D space
Summary of decision trees

What you can do now

• Define a decision tree classifier
• Interpret the output of a decision trees
• Learn a decision tree classifier using greedy algorithm
• Traverse a decision tree to make predictions
  - Majority class predictions
  - Probability predictions
  - Multiclass classification