Introduction to AI (continued)

Lecture #5
What will we cover today?

• Feature Selection
• Neural Networks
• Unsupervised Learning
  – K-means
Feature Selection

• Task: Given a string, predict whether it is an email address.
  abc@gmail.com

• Feature selection: Given an input $x$, which properties might be useful for predicting $y$?

• Needs a feature selector $\phi(x)$
  - contains_@  1
  - endswith_.com  1
  - endswith_.org  0
  - contains_%  0

  ← This is feature vector $\phi(x)$
  ← binary features have a value of 1 if present, 0 if not
Feature weights

• contains_@: 3
• endswith_.com: 2.4
• endswith_.org: 1.3
• contains_%: -2
• !contains_alpha: -4

• This is weight vector $w$
• Weights typically normalized to $[-1, 1]$
Linear combination

• For a new example string, sum the weights
  – If the sum is $\geq 0$, output Yes
  – else No

  ▪ String 1: “sam@sccourse.com”

  ▪ String 2: “0.5%”

contains_@: 3
dendswith_.com: 2.4
dendswith_.org: 1.3
contains_%: -2
!contains_alpha: -4
Let’s add some math!

\[
y = \text{sign}(w \cdot \varphi(x)) = \text{sign}\left(\sum_{i=1}^{I} w_i \cdot \varphi_i(x)\right)
\]

- \(x\): the input
- \(\varphi(x)\): vector of feature functions \(\{\varphi_1(x), \varphi_2(x), \ldots, \varphi_I(x)\}\)
- \(w\): the weight vector \(\{w_1, w_2, \ldots, w_I\}\)
- \(y\): the prediction, +1 if “yes”, -1 if “no”
  - (\(\text{sign}(v)\) is +1 if \(v \geq 0\), -1 otherwise)
Feature Selection

• Seems arbitrary
  – But actually captures our knowledge
  – What if we do not really know?

• How can we select good features then?
• Feature Template
  – Rather than a specific feature, decide what *kind* of feature may be useful
    • E.g., presence of certain characters in a string
  – And then let the system tell us which are good!

contains_@: 3
endswith_.com: 2.4
endswith_.org: 1.3
contains_%: -2
!contains_alpha: -4
Feature Templates

• Allows us to define a set of related features
  – contains@, contains_a, contains_b....
• As a template with a variable:
  – E.g., contains_
  – where _ can be filled by any single character
• Less burden on the designer since we do not need to know which features to choose, only if they are useful cues or not
• But... it may produce lots of “useless” features
Feature Template for “ends with”

abc@gmail.com

endsWith_a : 0
endsWith_b : 0
endsWith_c : 0
endsWith_d : 0
endsWith_e : 0
endsWith_f : 0
endsWith_g : 0
endsWith_h : 0
endsWith_i : 0
endsWith_j : 0
endsWith_k : 0
endsWith_l : 0
endsWith_m : 1
endsWith_n : 0
endsWith_o : 0
endsWith_p : 0
endsWith_q : 0
endsWith_r : 0
endsWith_s : 0
endsWith_t : 0
endsWith_u : 0
endsWith_v : 0
endsWith_w : 0
endsWith_x : 0
endsWith_y : 0
endsWith_z : 0

Inefficient to represent all zeros!
But it is common for feature vectors to be quite sparse
Feature Vector Representations

• As an Array (or vector)

\[0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0\]

– Good for dense features
– Efficient in space and speed
– E.g. in computer vision, pixel intensity vectors are typically dense
Feature Vector Representations

• As a Map
  
  \{“contains_@”: 1, “endswith_m”: 1\}
  
  – Good for sparse features
    
    • Known also as “one hot” if only one feature is set
  
  – Features not in the map are assumed 0
  
  – Typical in NLP applications
  
  – Sometimes trillions of features
  
  – But less efficient than arrays, slower
Dimensionality Reduction – selecting only the features that matter

- Feature Vectors for a data set

$x_1: [0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0]$
$x_2: [0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0]$
$x_3: [0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0]$
$x_4: [0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0]$
$x_5: [0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0]$

......

$x_n: [0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1]$

2/14/2019
A Couple of Dimensionality Reduction Techniques

• **Missing Values Ratio**
  – Columns that have too many zeros are unlikely to carry much useful information, remove columns where # of zeros > threshold

• **High Correlation Filter**
  – Columns that have very similar trends tend to carry similar information. So we calculate the correlation between values of given columns, remove columns where correlation coefficient > threshold
How do we know $\phi$ is good?

- Score: $w \cdot \phi(x)$

- Learning chooses the optimal $w$

- *How does feature extraction affect quality of prediction?*
  - Assume the $w$ is optimally selected by an oracle
Feature Selection affects what we can learn!

A hypothesis class is the set of possible predictors with a fixed $\phi(x)$ and varying $\mathbf{w}$:

$$\mathcal{F} = \{ f_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^d \}$$

Question: does $\mathcal{F}$ contain a good predictor?
Feature Selection issues

• Hypothesis class – a subset of all possible predictors

• Feature extraction defines a hypothesis class – a subset from all set of all possible predictors

• If feature extraction defines a feature set/hypothesis class which does not contain any good predictors, then no amount of learning will help!!

• We need to select features which are powerful to express predictors which are good!

• But it is necessary to keep the hypothesis class small, because the computational cost, and we usually do not get optimal $w$
Linear Predictors

Linear predictor:

\[ \phi(x)_1 \quad \phi(x)_2 \quad \phi(x)_3 \]

Output:

\[ \text{score} = \mathbf{w} \cdot \phi(x) \]
Neural Networks

A sigmoid function that converts activation input into a signal (neuron like). More recently simple RELU function used (latest bio)

Neural network:

$$\phi(x)_1 \rightarrow h_1 \rightarrow \sigma \rightarrow \text{score}$$

Intermediate hidden units:

$$h_j = \sigma(v_j \cdot \phi(x)) \quad \sigma(z) = (1 + e^{-z})^{-1}$$

Output:

$$\text{score} = w \cdot h$$
Neural Networks

• Think of the intermediate hidden units as learned features of a linear predictor

• The key idea is feature learning
  – BEFORE: Manually specify features
  – AFTER: Automatically learn them from data

• Deep learning: pack sparse large dimensionality representation into a dense small dimensionality space.
  – Allows generalizations to be learned!

• Objective function: Minimize the loss!
Deep neural networks

1-layer neural network:
\[
\text{score} = w^T x
\]

2-layer neural network:
\[
\text{score} = \sigma(w^T \sigma(Vx))
\]

3-layer neural network:
\[
\text{score} = \sigma(w^T \sigma(U \sigma(Vx)))
\]
Deep learning Neural Networks

**FACIAL RECOGNITION**

Deep-learning neural networks use layers of increasingly complex rules to categorize complicated shapes such as faces.

Layer 1: The computer identifies pixels of light and dark.

Layer 2: The computer learns to identify edges and simple shapes.

Layer 3: The computer learns to identify more complex shapes and objects.

Layer 4: The computer learns which shapes and objects can be used to define a human face.
Deep Neural Networks

120 million parameters, 4 million faces, 4000 people

Humans: 97.53%
DeepFace: 97.35%
Supervised vs. Unsupervised

• Labeled training data is expensive!
  – For some tasks, it may not even be available

• Unsupervised learning overcomes the need for training data

• Key idea: Data itself contains lots of rich latent structures, unsupervised learning discovers this structure automatically
Types of unsupervised learning

Clustering (e.g., K-means):

Dimensionality reduction (e.g., PCA):
Clustering

• Input: training set of input points
\[ D_{\text{train}} = \{x_1, x_2, \ldots, x_n\} \]

• Output: assignment of each point to a cluster
\[ [z_1, z_2, \ldots, z_n] \] where \( z_i \) belongs to \( \{1, \ldots, K\} \)

• Want similar points in the same cluster, dissimilar points should be in different clusters
K-means

Setup:

- Each cluster $k = 1, \ldots, K$ is represented by a centroid $\mu_k \in \mathbb{R}^d$
- Intuition: want each point $\phi(x_i)$ close to its assigned centroid $\mu_{z_i}$

Objective function:

\[
\text{Loss}_{\text{k-means}}(z, \mu) = \sum_{i=1}^{n} \| \phi(x_i) - \mu_{z_i} \|^2
\]

Need to choose centroids $\mu$ and assignments $z$ jointly
K-means example

- Input $D_{\text{train}} = \{0, 2, 10, 12\}$
- Output $K = 2$ clusters with centroids $\mu_1, \mu_2$

If know centroids $\mu_1 = 1, \mu_2 = 11$:

- $z_1 = \arg \min \{(0 - 1)^2, (0 - 11)^2\} = 1$
- $z_2 = \arg \min \{(2 - 1)^2, (2 - 11)^2\} = 1$
- $z_3 = \arg \min \{(10 - 1)^2, (10 - 11)^2\} = 2$
- $z_4 = \arg \min \{(12 - 1)^2, (12 - 11)^2\} = 2$

If know assignments $z_1 = z_2 = 1, z_3 = z_4 = 2$:

- $\mu_1 = \arg \min_{\mu} (0 - \mu)^2 + (2 - \mu)^2 = 1$
- $\mu_2 = \arg \min_{\mu} (10 - \mu)^2 + (12 - \mu)^2 = 11$
K-means

• But we know neither the centroids nor the assignments!

• Break down the hard problem into two easy problems
K-means algorithm (Step 1)

**Goal:** given centroids $\mathbf{\mu}_1, \ldots, \mathbf{\mu}_K$, assign each point to the best centroid.

**Algorithm: Step 1 of K-means**

For each point $i = 1, \ldots, n$:

Assign $i$ to cluster with closest centroid:

$$z_i \leftarrow \arg \min_{k=1,\ldots,K} \| \phi(x_i) - \mathbf{\mu}_k \|^2.$$
K-means algorithm (Step 2)

**Goal:** given cluster assignments $z_1, \ldots, z_n$, find the best centroids $\mu_1, \ldots, \mu_K$.

**Algorithm:** Step 2 of K-means

For each cluster $k = 1, \ldots, K$:

Set $\mu_k$ to average of points assigned to cluster $k$:

$$
\mu_k \leftarrow \frac{1}{|\{i : z_i = k\}|} \sum_{i : z_i = k} \phi(x_i)
$$
K-means algorithm

Objective:

$$\min_{z} \min_{\mu} \text{Loss}_{k\text{means}}(z, \mu)$$

Algorithm: K-means

- Initialize $\mu_1, \ldots, \mu_K$ randomly.
- For $t = 1, \ldots, T$:
  - Step 1: set assignments $z$ given $\mu$
  - Step 2: set centroids $\mu$ given $z$
K-means example

Input: \( \mathcal{D}_{\text{train}} = \{0, 2, 10, 12\} \)
Output: \( K = 2 \) centroids \( \mu_1, \mu_2 \in \mathbb{R} \)

Initialization (random): \( \mu_1 = 0, \mu_2 = 2 \)

Iteration 1:
- Step 1: \( z_1 = 1, z_2 = 2, z_3 = 2, z_4 = 2 \)
- Step 2: \( \mu_1 = 0, \mu_2 = 8 \)

Iteration 2:
- Step 1: \( z_1 = 1, z_2 = 1, z_3 = 2, z_4 = 2 \)
- Step 2: \( \mu_1 = 1, \mu_2 = 11 \)
Local minima

• K-means is guaranteed to converge on local minimum but not guaranteed to find global minimum

• Solution: Run multiple times with different random initializations, then choose solution that has lowest loss
What will we cover today?

• Feature Selection
• Neural Networks
• Unsupervised Learning
  – K-means
Something to think about

• What might be good features for predicting homepages?

• What might be good features for spam recognition?
Assigned Reading

• No paper assigned today