Feature Design

- Appropriately designed features are the critical factor of the success of machine learning
  - Mad cow disease Detection
    - “Temperature” is a better feature than “Gender” of a cow
- Wrong features lead to bad performance no matter how advance the classification technique is
Feature Design

- How features are designed?
  - Domain Expert
  - Experience
  - Ad-hoc

- Manually designed features may be
  - Unrelated to the task
  - Not enough to achieve good performance
  - Redundant to other features

- Feature set should be refined iteratively

Feature Refinement

- Feature Selection
  - Select a proper subset
  - Each feature is meaningful
  - Filter Method: Based on dataset property
    - E.g. Correlation, mutual information
  - Wrapper Method: Based on a classifier
    - E.g. Classification Error

- Feature Extraction
  - Build new dimensions, E.g. Height x 3 + blood pressure / 5
  - New dimensions may not be meaningful
  - E.g. LDA, PCA
Feature Design
Feature Selection: Filter

- No classifier is required and only based on characteristics of a dataset
- Measure the relation between a feature \(X\) and a class ID \(Y\)
  - Correlation
    \[
    \text{COR}(X, Y) = \frac{\text{cov}(X, Y)}{\sqrt{\text{var}(X)\text{var}(Y)}}
    \]
  - Mutual information
    \[
    I(X, Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \left( \frac{p(x, y)}{p(x)p(y)} \right)
    \]

\[
\begin{array}{c|c|c|c}
\text{s} & f_1 & f_2 & Y \\
\hline
s_1 & 4 & 2 & 1 \\
\hline
s_2 & 2 & 8 & 1 \\
\hline
s_3 & 8 & 4 & 2 \\
\hline
s_4 & 10 & 10 & 2 \\
\hline
s_5 & 6 & 6 & 2
\end{array}
\]

- \(\text{COR}(f_1, Y) = 0.87\)
- \(\text{COR}(f_2, Y) = 0.29\)
- \(f_1\) is more useful in classifying class 1 and 2 then \(f_2\) since the correlation value (COR) of \(f_1\) shows a stronger linear relationship between \(f_1\) and \(Y\)

Feature Design

Feature Selection: Filter

◆ Advantage
  ■ Independent of the classification algorithm
  ■ Easily scale to high dimensional datasets
  ■ Computationally simple and fast

◆ Disadvantage
  ■ Ignore the interaction with the classifier
  ■ Ignore feature dependency

Feature Design

Feature Selection: Wrapper

◆ Evaluate a feature set according to the generalization ability of a trained classifier
  ■ Forward Selection
    ■ Initial set: empty set
    ■ Add one feature in each iteration
    ■ Assume four features: \( f_1, f_2, f_3, f_4 \)
      - \( c(f_1) : 92\% \)
      - \( c(f_2) : 80\% \)
      - \( c(f_3) : 60\% \)
      - \( c(f_4) : 86\% \)
      Select \( f_1 \)
      Select \( f_3 \)
      Select \( f_2 \)

      - \( c(f_1, f_2) : 93\% \)
      - \( c(f_1, f_3) : 95\% \)
      - \( c(f_1, f_4) : 91\% \)
      - \( c(f_2, f_3, f_4) : 95\% \)
      - \( c(f_1, f_3, f_4) : 94\% \)

      Selection: \( f_1 > f_3 > f_2 > f_4 \)
Feature Design

Feature Selection: Wrapper

- Backward Elimination
  - Initial set: full set
  - Remove one feature iteratively
  - Assume four features: $f_1, f_2, f_3, f_4$
    - $c(f_2, f_3, f_4) : 90\%$
    - $c(f_1, f_3, f_4) : 93\%$
    - $c(f_1, f_2, f_4) : 91\%$
    - $c(f_1, f_2, f_3) : 80\%$
      - Remove $f_2$
      - Remove $f_3$
      - Remove $f_1$
  - Selection: $f_4 > f_1 > f_3 > f_2$

- Time complexity of Backward Elimination is larger than forward selection but usually has a better result
  - E.g. Backward Elimination considers $n$ classifiers with $n-1$ feature $n$ times but forward selection consider $n$ classifiers with 1 feature

- Advantage
  - Interaction between feature selection and model section
  - Take into account feature dependencies

- Disadvantage
  - Learning system is a black box
  - High risk of overfitting
  - Computationally intensive
Feature selection may not work sometimes

For example:

- Only using feature 1 or 2 cannot classify two classes accurately
- After rotating and transforming the data, new feature 2 is more useful for classification

Two most common methods

- Principal Component Analysis (PCA)
  \[ W^* = \arg \max_w W^T S W \]
  \[ s.t. \ W^T W = I \]
  where \( S = \text{Cov}(X, X) \)

- Linear Discriminant Analysis (LDA)
  \[ W^* = \arg \max_w \frac{W^T S_b W}{W^T S_w W} \]
  \[ s.t. \ W^T S_w W = I \]
  where \( S_b(S_w) \) is intra (inter) class distance
Feature Design

Feature Extraction

- **Advantage**
  - Project the original feature space to a new low dimension space

- **Disadvantage**
  - Alter original representation of the features
  - Hard to interpret the learning model

---

First Impression

- AlphaGo implemented by *DeepMind*
- **March 2016:** Beat Lee Sedol
- **Late 2016 – Early 2017:** Played as “Master” in the Internet and got 60w 0L, including playing with world #1 Mr. **Ke Jie**
Deep learning draws much public attention recently due to its satisfying performance in many difficult and complicated applications.

- **Computer Vision** (e.g. image classification, segmentation)
- **Speech Recognition**
- **Natural Language Processing** (e.g. sentiment analysis, translation)

What is Deep Learning?

- Branch of Machine Learning
- Generally refer to a neural network with multiple layers (deep architecture)
- Involve unsupervised and supervised learning in training the network
Why Deep Architecture?

- Our brain is a deep architecture
- A deep architecture can represent more complicated function than a shallow one
  - Many functions can be represented efficiently with a deep architecture but not with a shallow one

What’s New?

- Multilayer neural networks have been studied for more than 30 years
- What’s actually new?
- Old wine in a new bottle??
What’s New?

- No satisfying training method for deep architectures before 2006
  - Backpropagation algorithm loses its power in training deep-architecture NN as the cumulative backpropagated error signals decay exponentially wrt the number of layers
- Deep-architecture NN is less accurate than shallow one


What’s New?

- Breakthrough in 2006
- Three papers proposing new learning algorithms all containing:
  - Feature Learning by Unsupervised learning
    train a hidden layer each time
  - Prediction by Supervised learning
    train the last layer, and fine-tune all the layers

What’s New?

Brief History of Machine Learning

Digression of deep architecture due to the success of SVM

Type of Deep Learning

The common type of deep learning

- Stacked Autoencoder
- Convolution Neural Network
- Deep Belief Network
- Recurrent Neural Network

Stacked Autoencoder and Convolution Neural Network will be discussed in this lecture
How is a traditional Neural Network trained?

Initialize the weights

Artificial Intelligence and its applications - Lecture 10: Feature Design & Deep Learning
Stacked Autoencoder: Deep Learning VS Traditional NN

Traditional Learning Algorithm

How is a traditional Neural Network trained?

1. Error = | \( f(s) - y \) |
2. Update weights

Artificial Intelligence and its applications - Lecture 10: Feature Design & Deep Learning
How is a traditional Neural Network trained?

1. Error = | f(s_3) – y |
2. Update weights

Artificial Intelligence and its applications - Lecture 10: Feature Design & Deep Learning
**Stacked Autoencoder: Deep Learning VS Traditional NN**

**Traditional Learning Algorithm**

- **Gradient Descent** updates the weights by making many tiny adjustments
  - Each adjustment makes the network better at the recent samples but may be worse on others
  - Easily gets stuck in local minima
  - Search space too complex
  - Rely on weight initialization

![Graph showing the error landscape with local minima and global minima.](image)

**DL Algorithm**

- Weights are trained layer by layer
  - Intermediate layers are trained *(auto-encoder is used in this illustration)*
  - Final layer is trained to predict class
**Stacked Autoencoder: Deep Learning VS Traditional NN**

**DL Algorithm**

- **Auto-encoder**: it is forced to learn good features that describe the previous layer.

![Auto-encoder diagram](image)

We would like to train the weight

By minimizing the error (∥a – a’∥),

\[ b \] can represent \[ a \].

Next layer with (more) fewer units than the first one, this forces the units of the next layer to become good feature of the first layer.

- Weights are trained layer by layer.

![Weight training diagram](image)

Intermediate layers are trained to be auto-encoder (No Label is needed!)

Final layer is trained to predict class

Unsupervised Learning

Supervised Learning

Finetune can be applied
Hierarchies of features are learnt as non-linear transformations of data in DL
Need for manual feature design is reduced

Artificial Intelligence and its applications - Lecture 10: Feature Design & Deep Learning

Convolutional Neuronal Network

- In stacked autoencoder, all inputs connect to all neurons in the first hidden layer
- The performance of network may not be good for high dimensional problems (a lot of features)
  - Every input yields many connection
- For example: an image with 100x100 pixels has 10,000 dimensions

Convolutional Neuronal Network

- An input is connected to a number of neuron
  - For example, neurons only consider adjacent pixels in an image
- This is called Local network, locally connected networks, local receptive field network
Convolutional Neuronal Network

- Weight sharing is used for further reduction
  - Some parameters are equal
  - E.g. \( w_1 = w_4 = w_7, w_2 = w_5 = w_8, \)
     \[ w_3 = w_6 = w_9 \]

- It is called convolution

Convolutional Neuronal Network

- Another feature is Max-Pooling Layer
  - Also call subsampling layer
    - Reduce the size significantly
  - Translational invariance
    - Invariant to shift of inputs
- This is called convolutional neural network
Convolutional Neuronal Network

Other features (optional):
- Local Contrast Normalization (LCN)
  - Subtract the mean and divide the standard deviation of the incoming neurons
- Brightness invariance

Artificial Intelligence and its applications - Lecture 10: Feature Design & Deep Learning

39

Convolutional Neuronal Network

Feature Learning

Classifier

Convolution + ReLU
Max Pooling
Convolution + ReLU
Max Pooling

Dog (0.01)
Cat (0.04)
Boat (0.94)
Bird (0.02)

Fully Connected
### Convolution operation

#### Image

<table>
<thead>
<tr>
<th>x0</th>
<th>x1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Convolved Feature or Feature Map

<table>
<thead>
<tr>
<th>x0</th>
<th>x1</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

#### Operation

<table>
<thead>
<tr>
<th>Operation</th>
<th>Filter</th>
<th>Convolved Image</th>
</tr>
</thead>
</table>
| Identity    | \[
|             | \begin{bmatrix}
|             | 0 & 0 & 0 \\
|             | 0 & 1 & 0 \\
|             | 0 & 0 & 0 \\
|             | \end{bmatrix}
| | ![Identity Image](image1.png) |
| Edge detection | \[
|             | \begin{bmatrix}
|             | 1 & 0 & -1 \\
|             | 0 & 0 & 0 \\
|             | -1 & 0 & 1 \\
|             | \end{bmatrix}
| | ![Edge detection Image](image2.png) |
| Sharpen     | \[
|             | \begin{bmatrix}
|             | 0 & -1 & 0 \\
|             | -1 & 5 & -1 \\
|             | 0 & -1 & 0 \\
|             | \end{bmatrix}
| | ![Sharpen Image](image3.png) |
| Box blur (normalized) | \[
|             | \begin{bmatrix}
|             | \frac{1}{9} & 1 & 1 & 1 \\
|             | 1 & 1 & 1 & 1 \\
|             | \end{bmatrix}
| | ![Box blur Image](image4.png) |
| Gaussian blur (approximation) | \[
|             | \begin{bmatrix}
|             | \frac{1}{16} & 1 & 2 & 1 \\
|             | 2 & 4 & 2 \\
|             | 1 & 2 & 1 \\
|             | \end{bmatrix}
| | ![Gaussian blur Image](image5.png) |
Convolution operation

Rectified Feature Map

Pooling + ReLU (introduce non-linearity of convent which is linear)
Convolution operation

Rectified feature map

Deep Learning
Feature Learning

Face
Car
Elephants
Deep Learning Characteristics

- Learn from **raw** materials (Feature Learning)
- Very **deep** architecture (10 – 100 layers)
- Huge **training set** (big data era)
- Huge **computation complexity** (distributed system, GPU...)

Applications

**Image Recognition**

Imagenet

- > 1m color images
- 1000 classes
- different sizes (avg 482x415)
Applications
Recognition on Simple Drawing

Our neural net figured out 5 of your doodles. But it saw something else in the other 1. Select one to see what it saw.

https://quickdraw.withgoogle.com/

Applications
Object Recognition

1.31 dog
0.31 plays
0.45 catch
-0.02 with
0.25 white
1.62 ball
-0.10 near
-0.07 wooden
0.22 fence

1.12 woman
-0.28 in
1.23 white
1.45 dress
0.06 standing
-0.13 with
3.58 tennis
1.81 racket
0.06 two
0.05 people
-0.14 in
0.30 green
-0.09 behind
-0.14 her
Applications

Image Caption Generation

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"girl in pink dress is jumping in air."

"black and white dog jumps over fence."

"young girl in pink shirt is swimming in pool."

Applications

Scene Parsing
Applications

Scene Parsing

Auto Pilot Car
Applications

Automatic Colorization

Applications

Automatic Machine Translation
Applications
Create Images from Sketches

Synthesized Image

#NeuralDoodle

Applications
Game Playing

◆ Reinforcement Learning

Breakout

Mario