Table of Contents

- Representation Learning
- Neural Networks
- Convolutional Neural Networks
Conventional Machine Learning
Conventional Machine Learning

Data
Conventional Machine Learning
Conventional Machine Learning

Data → Learning Algorithms ← Prediction
Conventional Machine Learning

Data -> Learning Algorithms -> Prediction
Conventional Machine Learning

Data → Features → Prediction

Learning Algorithms
Conventional Machine Learning

Data — Feature Engineering — Features — Learning Algorithms — Prediction
Conventional Machine Learning

Data \rightarrow Feature Engineering \rightarrow Features \rightarrow Prediction

Learning Algorithms
Conventional Machine Learning

Data → Feature Engineering → Features → Learning Algorithms → Prediction
Example: Facial Recognition

Face detection → Prediction → Stalin
Internal Representation

RGB Representation

Sub-matrix that contains faces

Predicted Label

Stalin
Feature engineering is hard

https://alitarhini.wordpress.com/2010/12/05/face-recognition-an-introduction/
Feature engineering is hard

https://alitarhini.wordpress.com/2010/12/05/face-recognition-an-introduction/
Feature engineering is hard

Background

Feature engineering is hard

multiple faces/ no faces

https://alitarhini.wordpress.com/2010/12/05/face-recognition-an-introduction/
Feature engineering is hard

• Feature engineering is hard

  • Requires domain-knowledge;

  • Hard to find a general feature extractor

• Feature engineering and prediction are two separate steps.
Representation Learning

• Learning representations of the raw data that makes it easier to extract useful information when building prediction models.
Some existing representation examples

• **PCA:**
  - reduce the dimension of the raw data.
  - does not work when data does not line on linear manifold

Some existing representation examples

- PCA:
  - reduce the dimension of the raw data.
  - does not work when data does not line on linear manifold

Some existing representation examples

- PCA:
  - reduce the dimension of the raw data.
  - does not work when data does not line on linear manifold

Some existing representation examples

• PCA:
  • reduce the dimension of the raw data.
  • does not work when data does not line on linear manifold.

Some existing representation examples

• Kernel tricks:
  • project the data to higher dimensions to make them linear separable.
  • does not explicitly learn representations.
Some existing representation examples

- **Kernel tricks:**
  - Project the data to higher dimensions to make them linear separable.
  - Does not explicitly learn representations.
Some existing representation examples

- Kernel tricks:
  - project the data to higher dimensions to make them linear separable.
  - does not explicitly learn representations.
Deep Learning

- Deep Neural Networks
  - General feature extractor, with multiple levels of representation.
  - Automatically discover the representations needed for subsequent prediction tasks.
Neurons

A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Multi-layer Neural Networks

- $f()$: activation function
  - Activation function are usually nonlinear
  - When they are linear, reduces to linear models.
- $w_{ij}$: weights
- $z_j, z_k$: activations, weighted sums of previous layer’s units
- $y_j, y_k$: hidden units
- Each unit is obtained by applying the activation function on activations
Learning

• Goal: with inputs and outputs observed, learn the weights.

• Algorithm:
  • Gradient descent
  • Error Back propagation
https://www.youtube.com/watch?v=b4Vyma9wPHo
Gradient descent algorithm

Initial weight

Gradient $\frac{\partial \text{loss}}{\partial w}$

Global loss minimum

$w = w - \alpha \frac{\partial \text{loss}}{\partial w}$

https://www.youtube.com/watch?v=b4Vyma9wPHo
Error Back Propagation

- To use gradient descent, we need to know the value of $\frac{\partial E_l}{\partial w_{kl}}$.
- $\frac{\partial E_l}{\partial w_{kl}} = \frac{\partial E_l}{\partial z_l} \frac{\partial z_l}{\partial w_{kl}}$
- $\frac{\partial E_l}{\partial z_l} = \frac{\partial E_l}{\partial y_l} \frac{\partial y_l}{\partial z_l} = \frac{\partial E_l}{\partial y_l} \frac{\partial f(z_l)}{\partial z_l}$
- $\frac{\partial E_l}{\partial y_l} = \frac{1}{2} (y_l - t_l)^2$  
  $= y_l - t_l$ error
- $\frac{\partial z_l}{\partial w_{kl}} = \frac{\partial \sum_k w_{kl} y_k}{\partial w_{kl}} = y_k$

Neural Network: learning

- Initialize: randomly assign some weights (often small random values around 0).

1. **Forward pass**: take some input units X and calculate activations of all layers

2. **Back propagation**: obtain partial derivatives of weights using back prop

3. **Update weights** using gradient descent

4. Repeat 1 - 3 until convergence.
Neural Network: learning

- Initialize: randomly assign some weights (often small random values around 0).

1. **Forward pass**: take some input units $X$ and calculate activations of all layers

2. **Back propagation**: obtain partial derivatives of weights using back prop

3. **Update weights** using gradient descent

4. Repeat 1 - 3 until convergence.
Neural Network: learning

- Initialize: randomly assign some weights (often small random values around 0).

1. **Forward pass**: take some input units $X$ and calculate activations of all layers

2. **Back propagation**: obtain partial derivatives of weights using back prop

3. **Update weights** using gradient descent

4. Repeat 1 - 3 until convergence.
Neural Network: learning

- Initialize: randomly assign some weights (often small random values around 0).

1. **Forward pass**: take some input units X and calculate activations of all layers

2. **Back propagation**: obtain partial derivatives of weights using back prop

3. **Update weights** using gradient descent

4. Repeat 1 - 3 until convergence.
**Diagram (c)**

Output units

Hidden units H2

Hidden units H1

Input units

\[
y_j = f(z_j)
\]

\[
z_j = \sum_{i \in \text{Input}} w_{ji} x_i
\]

\[
y_k = f(z_k)
\]

\[
z_k = \sum_{j \in \text{H1}} w_{jk} y_j
\]

\[
y_l = f(z_l)
\]

\[
z_l = \sum_{i \in \text{H2}} w_{kl} y_k
\]

**Diagram (d)**

Compare outputs with correct answer to get error derivatives

\[
\frac{\partial E}{\partial y_l} = y_l - t_l
\]

\[
\frac{\partial E}{\partial y_k} = \sum_{i \in \text{out}} w_{ki} \frac{\partial E}{\partial z_i}
\]

\[
\frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial z_k}
\]

\[
\frac{\partial E}{\partial y_j} = \sum_{k \in \text{H2}} w_{jk} \frac{\partial E}{\partial z_k}
\]

\[
\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial z_j}
\]
Choice of activation function

- Activation function adds non-linearity to the linear weighted sum of hidden units.

- Common choices: Sigmoid and ReLu (Rectified Linear Unit)

- Sigmoid has the “vanishing gradient” problem: \[
\frac{df(x)}{dx} = f(x) \cdot (1 - f(x))
\]
Basic neural networks

- Free parameters:
  - the number of hidden layers
  - the number of units per layer.
  - # of hidden units increase ->
    - model complexity increases
    - more likely to overfit.
Convolutional Neural Networks

- **Sparse connectivity** (local connections): each unit depends on only on local regions of the previous layer.
  - rationale: local groups of values are often highly correlated
    - n-grams in 1D texts and speeches;
    - subregions in 2D photos
    - video clip from a longer video (3D).

Figure 9.2, Goodfellow et al., 2015
Convolutional Neural Networks

- Parameter sharing

- rationale: a particular layer fulfills some feature extraction tasks.

- This task should be invariant to the location of subregion.

http://cs231n.github.io/convolutional-networks/
Example of Convolution

Kernel is also known as filter

Figure 9.1, Goodfellow et al., 2015

Example of Convolution

Kernel is also known as filter

Input

\[
\begin{array}{cccc}
  a & b & c & d \\
  e & f & g & h \\
  i & j & k & l \\
\end{array}
\]

Kernel

\[
\begin{array}{cc}
  w & x \\
  y & z \\
\end{array}
\]

Output

\[
\begin{array}{cccc}
  aw + bx + ey + fz & bw + cx + fy + gz & cw + dx + gy + hz & \\
  ew + fx + iy + jz & fw + gx + jy + kz & gw + hx + ky + lz & \\
\end{array}
\]

Image

\[
\begin{array}{cccc}
  1_{x1} & 1_{x0} & 1_{x1} & 0 \ 0 \\
  0_{x0} & 1_{x1} & 1_{x0} & 1 \ 0 \\
  0_{x1} & 0_{x0} & 1_{x1} & 1 \ 1 \\
  0 \ 0 & 1 \ 1 & 1 \ 0 \\
  0 \ 1 \ 1 & 0 \ 0 \\
\end{array}
\]

Convolved Feature

4

Figure 9.1, Goodfellow et al., 2015

Example of Convolution

Kernel is also known as filter

Figure 9.1, Goodfellow et al., 2015

## Filter Examples

<table>
<thead>
<tr>
<th>Operation</th>
<th>Filter</th>
<th>Convolved Image</th>
</tr>
</thead>
</table>
| **Identity**  | \[
|               | \[
|               | \[
|               | \[
| **Edge detection** | \[
|               | \[
|               | \[
|               | \[
| **Sharpen**   | \[
|               | \[
|               | \[
| **Box blur**  | \[
| (normalized)  | \[
| **Gaussian blur** | \[
| (approximation) | \[


Feature Maps

• Each convolution extracts one type of feature; it is also called a feature map

• Weights of a feature map are the same (parameter sharing)

• There are often multiple feature maps; each aims to capture a different feature (e.g., edge, circle, lines).
One convolutional Layer

Figure 2, LeCun et al., 2015
Pooling

- why we need pooling:
  - translation invariant: small changes in feature location does not matter.
  - down sampling; reduce complexity.
- max-pooling is popular, but there are other types (e.g., average pooling).

Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. Left: In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. Right: The most common downsampling operation is max, giving rise to max pooling, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).

http://cs231n.github.io/convolutional-networks/
Convolutional Layer + Pooling Layer
Why multiple layers

- Multiple levels of representation
  - lower layers capture basic motifs such as edges and circles.
  - Upper layers capture combinations of motifs.

Neural Networks vs Conv Neural Networks
Neural Networks vs Conv Neural Networks

- Neural networks are wide
- Fully connected
- Weights between layers are all different
Neural Networks vs Conv Neural Networks

- Neural networks are wide
- Fully connected
- Weights between layers are all different

- Conv Nets are deep
- Sparsely connected
- Weights in each feature map is the same: extract the same type of feature well.
- weights across feature maps are different: extract many different types of features.
Last step: fully connected layers

- After several rounds of (Conv Layers + pooling layers), the last (or last several) layers produce outputs.

- Without the feature extraction part (Conv Layers + pooling layers), the network is equivalent to basic neural networks.

- We can skip the last fully connected layers, and use the extracted features as input for other algorithms (e.g., SVM or logistic regression).
AlexNet, 2012

CNN as a feature extractor

- Visualize the last layer: should contain representation of the object itself.

- A good representation of feature should naturally put similar objects in similar positions.
Test image  L2 Nearest neighbors in **feature** space

Regularization

- traditional: add regularizer terms, to the loss function $||w||^2$
- early stopping: a kind of cross-validation strategy.
  - split the data into train, validation and test
  - stop when test accuracy begins to drop
- dropout: memorizing the past is not always useful
  - randomly remove some nodes in the network
Regularization

- traditional: add regularizer terms, to the loss function $||w||^2$
- early stopping: a kind of cross-validation strategy.
  - split the data into train, validation and test
  - stop when test accuracy begins to drop
- dropout: memorizing the past is not always useful
  - randomly remove some nodes in the network
Regularization

• traditional: add regularizer terms, to the loss function $||w||^2$

• early stopping: a kind of cross-validation strategy.
  
  • split the data into train, validation and test
  
  • stop when test accuracy begins to drop

• dropout: memorizing the past is not always useful
  
  • randomly remove some nodes in the network
Regularization

- traditional: add regularizer terms, to the loss function $||w||^2$
- early stopping: a kind of cross-validation strategy.
  - split the data into train, validation and test
  - stop when test accuracy begins to drop
- dropout: memorizing the past is not always useful
  - randomly remove some nodes in the network
Lots of parameter choices

• Number of layers:
  • More layers means better representation ability
  • may overfit; increase computational complexity.

• Number of feature maps: usually double by each convolutional layers.

• Filter size, stride size.

• Learning rates

• Batch size: how many data points the learning algorithm sees each time

• Number of epochs: # of full pass of data.
Some history
Some history

Some history

- Rumelhart, Hinton, and Ronald(1986): Back Prop
Some history


• Rumelhart, Hinton, and Ronald (1986): Back Prop

Some history

- Rumelhart, Hinton, and Ronald (1986): Back Prop
- ImageNet: 14 million images organized according to the WordNet hierarchy of nouns
Some history

- Rumelhart, Hinton, and Ronald (1986): Back Prop
- ImageNet: 14 million images organized according to the WordNet hierarchy of nouns
  - [http://image-net.org/about-overview](http://image-net.org/about-overview)
Some history

- Rumelhart, Hinton, and Ronald (1986): Back Prop
- ImageNet: 14 million images organized according to the WordNet hierarchy of nouns
  - http://image-net.org/about-overview
Some history

- Rumelhart, Hinton, and Ronald (1986): Back Prop
- ImageNet: 14 million images organized according to the WordNet hierarchy of nouns
  - http://image-net.org/about-overview
- ImageNet Challenge: 1M images and 1000 categories.
Some history

- Rumelhart, Hinton, and Ronald (1986): Back Prop
- ImageNet: 14 million images organized according to the WordNet hierarchy of nouns
  - http://image-net.org/about-overview
- ImageNet Challenge: 1M images and 1000 categories.
  - Image classification
Some history

- Rumelhart, Hinton, and Ronald (1986): Back Prop
- ImageNet: 14 million images organized according to the WordNet hierarchy of nouns
  - [http://image-net.org/about-overview](http://image-net.org/about-overview)
- ImageNet Challenge: 1M images and 1000 categories.
  - Image classification
  - Single-object localization: try to find a bounding box indicating the position and scale of one instance of each object category
Some history

- Rumelhart, Hinton, and Ronald (1986): Back Prop
- ImageNet: 14 million images organized according to the WordNet hierarchy of nouns
  - http://image-net.org/about-overview
- ImageNet Challenge: 1M images and 1000 categories.
  - Image classification
  - Single-object localization: try to find a bounding box indicating the position and scale of one instance of each object category
  - Object detection: find a list of object categories present in the image
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners
What if you do not have many data

- Transfer Learning
  - ConvNet as a **feature extractor**: feed images into existing models, and obtain the output before the last fully connected layer.
  - work if your images is similar to that of ImageNet
- **Fine-tuning** ConvNet:
  - Freeze the weights of beginning layers of existing models
  - Train the weights of later layers
CNN in other contexts

- Text Classification

Summary

- Learning representation itself is important
- Deep learning provides a general feature extractor
- Extracted features can be used for prediction tasks