1. Softmax Regression
2. Tracking Model Performance
3. Limits of Linear Classification
4. Midterm Review
5. Introducing Neural Networks
6. Expressivity of a Neural Network

---

Announcements
project: groups of 2-3
2 options
Progress During Learning

- Track progress during learning by plotting training curves.
- Chose the training criterion (e.g. squared error, cross-entropy) partly to be easy to optimize.
- May wish to track other metrics to measure performance (even if we can’t directly optimize them).

Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.
We can track **accuracy**, or fraction correctly classified.

- Equivalent to the average 0–1 loss, the **error rate**, or fraction incorrectly classified.
- Useful metric to track even if we couldn’t optimize it.

Another way to break down the accuracy:

\[
Acc = \frac{TP + TN}{P + N} = \frac{TP + TN}{(TP + FN) + (TN + FP)}
\]

- P=num positive; N=num negative;
- TP=true positives; TN=true negatives
- FP=false positive or a type I error
- FN=false negative or a type II error
Accuracy is Highly Sensitive to Class Imbalance

Suppose you are screening patients for a particular disease. It’s known that 1% of patients have that disease.

- What is the simplest model that can achieve 99% accuracy?

  predict everyone does not have disease
Sensitivity and Specificity

Useful metrics even under class imbalance!

\[
\text{Sensitivity} = \frac{TP}{TP + FN} \quad \text{[True positive rate]}
\]

\[
\text{Specificity} = \frac{TN}{TN + FP} \quad \text{[True negative rate]}
\]

What happens if our problem is not linearly separable? How do we pick a threshold for \( y = \sigma(x) \)?

\[
y = \begin{cases} 
1 & \text{if } \logit \geq 0.5 \\
0 & \text{if } \logit < 0.5
\end{cases}
\]
Designing Diagnostic Tests

- A binary model to predict whether someone has a disease.
- What happens to sensitivity and specificity as you slide the threshold from left to right?

![Diagram showing change in threshold]

- Change threshold to predict more positives
- Example: 0.5 → 0.4
- FP TP ↑
- TN FN ↓
Tradeoff between Sensitivity and Specificity

As we increase the criterion value (i.e. move from left to right), how do the sensitivity and specificity change?

The diagram illustrates the trade-off between sensitivity and specificity. As the criterion value increases, sensitivity increases and specificity decreases, approaching a threshold close to 1.
Area under the ROC curve (AUC) can quantify if a binary classifier achieves a good tradeoff between sensitivity and specificity.
Confusion Matrix for Multi-Class classification

- Visualizes how frequently certain classes are confused.
- $K \times K$ matrix; rows are true labels, columns are predicted labels, entries are frequencies.
- What does the confusion matrix for a perfect classifier look like?
1. Softmax Regression
2. Tracking Model Performance
3. Limits of Linear Classification
4. Midterm Review
5. Introducing Neural Networks
6. Expressivity of a Neural Network
XOR is Not Linearly Separable

Some datasets are not linearly separable, e.g. XOR.

Visually obvious, but how can we prove this formally?
Proof That XOR is Not Linearly Separable

Proof by Contradiction: \( a^T x < b \)

- Half-spaces are convex. That is, if two points lie in a half-space, the line segment connecting them also lie in the same half-space.
- Suppose that the problem is feasible.
- If the positive examples are in the positive half-space, then the green line segment must be as well.
- Similarly, the red line segment must lie in the negative half-space.
- But, the intersection can’t lie in both half-spaces. Contradiction!
Sometimes, we can overcome this limitation using feature maps, e.g., for XOR.

\[ \psi(x) = \begin{pmatrix} x_1 \\ x_2 \\ x_1x_2 \end{pmatrix} \]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>( \psi_1(x) )</th>
<th>( \psi_2(x) )</th>
<th>( \psi_3(x) )</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- This is linearly separable. (Try it!)
- Designing feature maps can be hard. Can we learn them?
1. Softmax Regression

2. Tracking Model Performance

3. Limits of Linear Classification

4. Midterm Review

5. Introducing Neural Networks

6. Expressivity of a Neural Network
1. Softmax Regression

2. Tracking Model Performance

3. Limits of Linear Classification

4. Midterm Review

5. Introducing Neural Networks

6. Expressivity of a Neural Network
Neurons in the Brain

Neurons receive input signals and accumulate voltage. After some threshold, they will fire spiking responses.

[Pic credit: www.moleculardevices.com]
A Simpler Neuron

For neural nets, we use a much simpler model for neuron, or unit:

$$y = \phi \left( w^\top x + b \right)$$

- Similar to logistic regression: $y = \sigma \left( w^\top x + b \right)$
- By throwing together lots of these simple neuron-like processing units, we can do some powerful computations!
A Feed-Forward Neural Network

- A directed acyclic graph (DAG)
- Units are grouped into layers

Model: \( f(x) \) expressible \( \nabla f(x) \)

optim.

no cycles

- Linear (neural nets)

A directed acyclic graph (DAG)

Units are grouped into layers

Depth

Input layer (pixels of image)

Second hidden layer

Cat or dog

Output layer

2 outputs

an output unit

\( y_1 \) \( y_2 \)

2 inputs

\( x_1 \) \( x_2 \)

\( h_1^{(1)} \) \( h_2^{(1)} \) \( h_3^{(1)} \)

\( h_1^{(2)} \) \( h_2^{(2)} \) \( h_3^{(2)} \)
A multi-layer network consists of fully connected layers. In a fully connected layer, all input units are connected to all output units. Each hidden layer $i$ connects $N_{i-1}$ input units to $N_i$ output units. The weight matrix is $N_i \times N_{i-1}$. The outputs are a function of the input units:

$$y = f(x) = \phi(Wx + b)$$

$\phi$ is applied component-wise.

### MNIST

- Input: 784
- 200 hidden units
- Output: 10
Some Activation Functions

Identity

\[ y = z \]

Rectified Linear Unit (ReLU)

\[ y = \max(0, z) \]

Soft ReLU

\[ y = \log(1 + e^z) \]
More Activation Functions

**Hard Threshold**

\[ y = \begin{cases} 
1 & \text{if } z > 0 \\
0 & \text{if } z \leq 0 
\end{cases} \]

*not used in practice*

**Logistic**

\[ y = \frac{1}{1 + e^{-z}} \]

\((0, 1)\)

**Hyperbolic Tangent**

\[ y = \frac{e^z - e^{-z}}{e^z + e^{-z}} \]

\((-1, 1)\)
Computation in Each Layer

Each layer computes a function.

\[
\begin{align*}
    h^{(1)} &= f^{(1)}(x) = \phi(W^{(1)}x + b^{(1)}) \\
    h^{(2)} &= f^{(2)}(h^{(1)}) = \phi(W^{(2)}h^{(1)} + b^{(2)}) \\
    & \vdots \\
    y &= f^{(L)}(h^{(L-1)})
\end{align*}
\]

If task is regression: choose
\[
y = f^{(L)}(h^{(L-1)}) = (w^{(L)})^\top h^{(L-1)} + b^{(L)}
\]

If task is binary classification: choose
\[
y = f^{(L)}(h^{(L-1)}) = \sigma((w^{(L)})^\top h^{(L-1)} + b^{(L)})
\]

\[
\sigma(x) = \begin{cases} 
1 & \text{if } x \geq 0 \\
0 & \text{if } x < 0
\end{cases}
\]

\[\in (0,1)\]
A Composition of Functions

The network computes a composition of functions.

\[ y = f^{(L)} \circ \cdots \circ f^{(1)}(x). \]

Modularity: We can implement each layer’s computations as a black box.
Feature Learning

Neural nets can be viewed as a way of learning features:

\[
\begin{align*}
\mathbf{y} &= \mathbf{W}^3 \psi(x) \\
\mathbf{h}^{(2)} &= \psi(x) \\
\mathbf{h}^{(1)} \\
\mathbf{x}
\end{align*}
\]

The goal:

poly

\[1, x_1, x_1^2, \ldots, x^n\]

\[\sigma (w_1 x_1 + w_2 x_2)\]

Not linearly separable
Feature Learning

- Suppose we’re trying to classify images of handwritten digits.
- Each image is represented as a vector of $28 \times 28 = 784$ pixel values.
- Each hidden unit in the first layer acts as a feature detector.
- We can visualize $w$ by reshaping it into an image.
  Below is an example that responds to a diagonal stroke.
Features for Classifying Handwritten Digits

Features learned by the first hidden layer of a handwritten digit classifier:

Unlike hard-coded feature maps (e.g., in polynomial regression), features learned by neural networks adapt to patterns in the data.
1 Softmax Regression
2 Tracking Model Performance
3 Limits of Linear Classification
4 Midterm Review
5 Introducing Neural Networks
6 Expressivity of a Neural Network
Expressivity

- A hypothesis space $\mathcal{H}$ is the set of functions that can be represented by some model.
- Consider two models $A$ and $B$ with hypothesis spaces $\mathcal{H}_A, \mathcal{H}_B$.
- If $\mathcal{H}_B \subseteq \mathcal{H}_A$, then $A$ is more expressive than $B$.
- $A$ can represent any function $f$ in $\mathcal{H}_B$.

- Some functions (XOR) can’t be represented by linear classifiers. Are deep networks more expressive?
Consider a linear layer: the activation function was the identity. The layer just computes an affine transformation of the input.

Any sequence of linear layers is equivalent to a single linear layer.

\[ y = W^{(3)} W^{(2)} W^{(1)} x \]

Deep linear networks can only represent linear functions — no more expressive than linear regression.
Multi-layer feed-forward neural networks with non-linear activation functions

Universal Function Approximators: They can approximate any function arbitrarily well, i.e., for any $f : X \rightarrow T$ there is a sequence $f_i \in \mathcal{H}$ with $f_i \rightarrow f$.

True for various activation functions (e.g. thresholds, logistic, ReLU, etc.)
Designing a Network to Classify XOR

Assume a hard threshold activation function.

\[ y = h_1 - h_2 \]

\[ \tilde{h}_1 = x_1 + x_2 - 0.5 \]

\[ h_1 = \text{ReLU}(\tilde{h}_1) \]

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( y )</th>
<th>( \tilde{h}_1 )</th>
<th>( h_1 )</th>
<th>( h_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1.5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Designing a Network to Classify XOR

$h_1$ computes $x_1 \lor x_2$

$$\mathbb{I}[x_1 + x_2 - 0.5 > 0]$$

$h_2$ computes $x_1 \land x_2$

$$\mathbb{I}[x_1 + x_2 - 1.5 > 0]$$

$y$ computes $h_1 \land (\neg h_2) = x_1 \oplus x_2$

$$\mathbb{I}[h_1 - h_2 - 0.5 > 0]$$

$$\equiv \mathbb{I}[h_1 + (1 - h_2) - 1.5 > 0]$$
Universality for Binary Inputs and Targets

- Hard threshold hidden units, linear output
- Strategy: $2^D$ hidden units, each of which responds to one particular input configuration

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Only requires one hidden layer, though it is extremely wide.
Expressivity of the Logistic Activation Function

- What about the logistic activation function?
- Approximate a hard threshold by scaling up $w$ and $b$.

$$y = \sigma(x)$$  \hspace{2cm}  $$y = \sigma(5x)$$

- Logistic units are differentiable, so we can learn weights with gradient descent.
What is Expressivity Good For?

- May need a very large network to represent a function.
- Non-trivial to learn the weights that represent a function.
- If you can learn any function, over-fitting is potentially a serious concern!

For the polynomial feature mappings, expressivity increases with the degree $M$, eventually allowing multiple perfect fits to the training data. This motivated $L^2$ regularization.

- Do neural networks over-fit and how can we regularize them?
The topic of over-fitting (when & how it happens, how to regularize, etc.) for neural networks is not well-understood, even by researchers!

- In principle, you can always apply $L^2$ regularization.
- You will learn more in CSC413.

A common approach is early stopping, or stopping training early, because over-fitting typically increases as training progresses.

Don’t add an explicit $\mathcal{R}(\theta)$ term to our cost.
Conclusion

- Multi-class classification
- Selecting good metrics to track performance in models
- From linear to non-linear models