CS 453X: Class 23

Jacob Whitehill
(Unsupervised) pre-training
Feature representations

- One of the reasons why NNs are so powerful is that they can learn **feature representations** of the raw input data.

- Classifying/regressing the target variable $\hat{y}$ is often easier once the raw data have been transformed into a different feature space.

- We saw this with XOR.
Feature representations

• One of the reasons why NNs are so powerful is that they can learn **feature representations** of the raw input data.

• Classifying/regressing the target variable $\hat{y}$ is often easier once the raw data have been transformed into a different feature space.

• With MNIST, the NN seemed to recognize brush-strokes:
Unlabeled data

- In some application domains (e.g., object recognition/detection in images), collecting *labeled* data is hard, but collecting *unlabeled* data is easy.

- How might oodles of unlabeled data help us to train better ML models?
Learning good features

• We can harness unsupervised learning algorithms to learn good feature representations from unlabeled data.

• **Unsupervised**: examples without labels.
Learning good features

- We can harness unsupervised learning algorithms to learn good feature representations from unlabeled data.

  - **Unsupervised**: examples without labels.

- Key intuition: a good representation captures the essence of the raw input data.

  - We can “compress” the data into a smaller representation.

  - We can “uncompress” it to *reconstruct* the original data.
Auto-encoders

• Let $\mathbf{x}$ be the raw input data.

• Let $\mathbf{h}$ be the hidden representation that captures the “essence” of $\mathbf{x}$. We say $\mathbf{h}$ has been encoded from $\mathbf{x}$.

• We can compute $\mathbf{h}$ using a neural network (just 1 layer in this example, but could be deeper).

$$\mathbf{h} = \sigma^{(1)} \left( \mathbf{W}^{(1)} \mathbf{x} + \mathbf{b}^{(1)} \right)$$
Auto-encoders

• If $h$ contains the essential features of $x$, then we can use $h$ to reconstruct (approximate) the original data $x$.

• Let $\hat{x}$ denote our reconstruction of $x$. We say $\hat{x}$ has been **decoded** from $h$.

$$\hat{x} = \sigma^{(2)} \left( W^{(2)} h + b^{(2)} \right)$$
Auto-encoders

• Putting the two components (encoder+decoder) together, we arrive at an auto-encoder.

\[
\hat{x} = \sigma^{(2)} \left( W^{(2)} \sigma^{(1)} \left( W^{(1)} x + b^{(1)} \right) + b^{(2)} \right)
\]
Auto-encoders: training loss function

- With auto-encoders, we optimize $W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)}$ to make our reconstructions as accurate as possible, i.e., minimize:

$$f_{\text{MSE}}(W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}) = \frac{1}{n} \sum_{i=1}^{n} (\hat{x}^{(i)} - x^{(i)})^2$$

$$= \frac{1}{n} \sum_{i=1}^{n} (\hat{x}^{(i)} - x^{(i)})^\top (\hat{x}^{(i)} - x^{(i)})$$

![Auto-encoder diagram]
Auto-encoders:
training loss function

- Notice that this loss function does not require any training labels — there is no mention of any $y$!

$$f_{\text{MSE}}(W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}) = \frac{1}{n} \sum_{i=1}^{n} (\hat{x}^{(i)} - x^{(i)})^2$$

$$= \frac{1}{n} \sum_{i=1}^{n} (\hat{x}^{(i)} - x^{(i)})^\top (\hat{x}^{(i)} - x^{(i)})$$
Exercise

- If we let $k = m$, then what is an easy (but useless) way to set the weights to give a perfect reconstruction (0 MSE)?
Exercise

• If we let $k=m$, then what is an easy (but useless) way to set the weights to give a perfect reconstruction (0 MSE)?

• If $k = m$, then we can just set $W^{(1)} = W^{(2)} = I$ (identity matrix). This gives 0 MSE but does not learn any interesting representation!
Exercise

• If we let $k=m$, then what is an easy (but useless) way to set the weights to give a perfect reconstruction (0 MSE)?

• For this reason, we usually set $k < m$; the hidden layer is then called a bottleneck.

*An important exception are de-noising auto-encoders.
Auto-encoders for unsupervised pre-training

• After training the auto-encoder NN, $W^{(1)}$ and $b^{(1)}$ have hopefully learned to encode $x$ into a representation that is useful for a variety of classification/regression problems.
Auto-encoders for unsupervised pre-training

• After training the auto-encoder NN, $W^{(1)}$ and $b^{(1)}$ have hopefully learned to encode $x$ into a representation that is useful for a variety of classification/regression problems.

• We can now just “chop off” the decoder layer(s)…
Auto-encoders for unsupervised pre-training

• After training the auto-encoder NN, $\mathbf{W}^{(1)}$ and $\mathbf{b}^{(1)}$ have hopefully learned to encode $\mathbf{x}$ into a representation that is useful for a variety of classification/regression problems.

• …and keep just the encoder layer(s).
Auto-encoders for unsupervised pre-training

• We now have a trained encoder network that can “compress” every input example $\mathbf{x}$ into its “essence” $\mathbf{h}$. 

![Encoder diagram]
Auto-encoders for unsupervised pre-training

- Now, suppose we also have a (typically smaller) set of labeled examples \( \{(x^{(i)}, y^{(i)})\}_{i=1}^n \).
Auto-encoders for unsupervised pre-training

• Now, suppose we also have a (typically smaller) set of labeled examples \( \{(x^{(i)}, y^{(i)})\}_{i=1}^{n} \).

• We can use the encoder network to convert each \( x \) to \( h \) to obtain \( \{(h^{(i)}, y^{(i)})\}_{i=1}^{n} \).
Auto-encoders for unsupervised pre-training

- Now, suppose we also have a (typically smaller) set of labeled examples \( \{(x^{(i)}, y^{(i)})\}_{i=1}^{n} \).

- We can use the encoder network to convert each \( x \) to \( h \) to obtain \( \{(h^{(i)}, y^{(i)})\}_{i=1}^{n} \).

- We then train a secondary NN (or any other ML model) to predict \( y \) from \( h \).
Auto-encoders for unsupervised pre-training

- After training, the two networks (encoder + secondary) can be seen as a single NN that analyzes each input $x$ to make a prediction $\hat{y}$. 

\[
\begin{align*}
& x_1 \\
& x_2 \\
& \cdots \\
& x_m \\
& \vdots \\
& h_1 \\
& h_2 \\
& \cdots \\
& h_k \\
& b^{(1)} \\
& w^{(1)} \\
& \hat{y}
\end{align*}
\]
Auto-encoders for unsupervised pre-training

• Why does this help?

• The first layers of the overall network were trained on a large amount of data.

• Compressing \( \mathbf{x} \) into \( \mathbf{h} \) makes the secondary predictions (hopefully) easier.

\[
\begin{align*}
\mathbf{x}_1 & \rightarrow \mathbf{W}^{(1)} & \mathbf{h}_1 \\
\mathbf{x}_2 & \rightarrow \mathbf{W}^{(1)} & \mathbf{h}_k \\
\vdots & & \vdots \\
\mathbf{x}_m & \rightarrow \mathbf{W}^{(1)} & \mathbf{b}^{(1)} \\
& \rightarrow & \\
& \rightarrow & \\
& \rightarrow & \\
& \rightarrow & \hat{\mathbf{y}}
\end{align*}
\]
Auto-encoders for unsupervised pre-training

• In addition to training the secondary NN, we can — optionally — adjust the parameters of the encoder network.

• Since the encoder was trained on a much larger (unlabeled) dataset, we don’t want to “mess up” its weights too much based on just a small labeled dataset.
Auto-encoders for unsupervised pre-training

• In addition to training the secondary NN, we can — optionally — adjust the parameters of the encoder network.

• Since the encoder was trained on a much larger (unlabeled) dataset, we don’t want to “mess up” its weights too much based on just a small labeled dataset.

• Hence, we often use a small learning rate ==> **fine-tuning.**
(Supervised) pre-training
Supervised pre-training

- An alternative strategy to finding good feature representations is to borrow a NN from a related task.

- For instance, there now exist high-accuracy networks for recognizing 1000+ object categories from images (next slide).

- We can “borrow” the feature representation from one ML model and apply it to another application domain…
Learning representations

- The first feature representation looks vaguely like the representation learned by my MNIST network.

http://www.deeplearningbook.org/contents/intro.html
Learning representations

- Each layer of the network finds successively more abstract feature representations.
- This was not “hard-coded” — it just turned out that these representations were useful for predicting the target labels.

http://www.deeplearningbook.org/contents/intro.html
Supervised pre-training

- Might one (or more) of the feature representations from this NN do well on a different but related problem, e.g., smile detection or age estimation?

- Strategy:
  1. Pre-train a NN on a large dataset for a general-purpose image recognition task.
  2. “Chop off” the final layer(s).
  3. Add a secondary network in place of the deleted layers, and train it for the new prediction task.
  4. Optional: fine-tune the rest of the NN.
Supervised pre-training

Replace with secondary network for new application domain.

http://www.deeplearningbook.org/contents/intro.html
Supervised pre-training

Chop off.
Supervised pre-training

• This strategy is known as supervised pre-training and can be highly effective for application domains for which only a small number of labeled data are available.
Convolution neural networks