CPSC 340:
Machine Learning and Data Mining

Neural Networks
Fall 2016
Admin

• Assignment 4:
  – 3 late days to hand in Monday.

• Assignment 5:
  – Out, due next Friday.

• Assignment 6:
  – Out, due last day of class.

• Final:
  – December 12 (8:30am – HEBB 100)
  – Covers Assignments 1-6.
  – Final from last year and list of topics will be posted.
  – Closed-book, cheat sheet: 4-pages each double-sided.
Supervised Learning Roadmap

  – We learned parameters ‘w’ based on the original features $x_i$ and target $y_i$.

• Part 3: Change of Basis.
  – We learned parameters ‘w’ based on a change of basis $z_i$ and target $y_i$.

• Part 4: Latent-Factor Models.
  – We learned parameters ‘W’ for basis $z_i$ based on only on features $x_i$.
  – You can then learn ‘w’ based on change of basis $z_i$ and target $y_i$.

• Part 5: Neural Networks.
  – Jointly learn ‘W’ and ‘w’ based on $x_i$ and $y_i$.
  – Learn basis $z_i$ that is good for supervised learning.
Neural Networks: Introducing Non-Linearity

• Natural choice of neural network regression objective would be:

\[ f(w, W) = \frac{1}{2} \sum_{i=1}^{n} (w^T z_i - y_i)^2 = \frac{1}{2} \sum_{i=1}^{\hat{n}} (w^T (W x_i) - y_i)^2 \]

• But we saw last time this gives a linear model.

\[ w^T W \text{ is a vector so equivalent to just having } w^T x_i \text{ for some } w! \]

• Typical fix is to introduce non-linearity ‘h’:

\[ f(w, W) = \frac{1}{2} \sum_{i=1}^{n} (w^T h(W x_i) - y_i)^2 \text{ where } 'h' \text{ has 'd' inputs and 'k' outputs.} \]

• Most common choice of ‘h’ is sigmoid applied to elements of \( W x_i \).

\[ z_{ic} = \frac{1}{1 + \exp(-W_c x_i)} \]
Notation for Neural Networks

We have our usual supervised learning notation:

\[
X = \begin{bmatrix}
  x_1^T \\
  x_2^T \\
  \vdots \\
  x_n^T
\end{bmatrix}_{n \times d} \quad y = \begin{bmatrix}
  y_1 \\
  y_2 \\
  \vdots \\
  y_n
\end{bmatrix}_{n \times 1}
\]

We have our latent features:

\[
Z = \begin{bmatrix}
  z_1^T \\
  z_2^T \\
  \vdots \\
  z_n^T
\end{bmatrix}_{n \times K}
\]

We have two sets of parameters:

\[
W = \begin{bmatrix}
  w_1 \\
  w_2 \\
  \vdots \\
  w_K
\end{bmatrix}_{k \times 1}
\]

\[
W = \begin{bmatrix}
  W_1 \\
  W_2 \\
  \vdots \\
  W_K
\end{bmatrix}_{k \times d}
\]
Supervised Learning Roadmap

Hand-engineered features:

- Requires domain knowledge and can be time-consuming.
- Good representation of $X_i$ might be bad for predicting $y_i$.

Learn a latent-factor model:

- "I think this basis will work."
- Use latent features in supervised model.

Learn $h$ and $W$ together:

- But still gives a linear model.

Neural network:

- Extra non-linear transformation $h$. 

Note: $y_i$, $Z_i$, $X_i$, $w_k$, $h(z_i)$.
Why Sigmoid?

• Consider setting ‘h’ to define binary features $z_i$ using:

$$z_i = I[W_c x_i \geq 0]$$

$$= \begin{cases} 
1 & \text{if } W_c x_i \geq 0 \\
0 & \text{if } W_c x_i < 0
\end{cases}$$

– Vector $z_i = h(Wx_i)$ can be viewed as binary features.
– $z_i$ can take $2^k$ possible values (combinatorial number of “concepts”).

• But non-differentiable and discontinuous so hard to optimize.
• Sigmoid is a smooth approximation to these binary features.
Why “Neural Network”? 

• Cartoon of “typical” neuron:

• Neuron has many “dendrites”, which take an input signal.
• Neuron has a single “axon”, which sends an output signal.
• With the right input to dendrites:
  — “Action potential” along axon (like a binary signal):

https://en.wikipedia.org/wiki/Neuron
https://en.wikipedia.org/wiki/Action_potential
Why “Neural Network”? 

If $w_c x_i \geq 0$ neuron sends signal along axon. We approximate binary signal with $\frac{1}{1 + \exp(-w_c x_i)}$. 

https://en.wikipedia.org/wiki/Neuron
Why “Neural Network”?

If \( W_c x_i > 0 \) neuron sends signal along axon.

We approximate binary signal with \( \frac{1}{1 + \exp(-W_c x_i)} \)

If \( W_c x_i > 0 \) neuron computes \( W_c x_i \) to send signal to other neurons.
Why “Neural Network”? 

- Predictions based on aggregation $W^T h(Wx_i)$ at $y_i$ "neuron"
- Synapse between $z_{ik}$ and $y_i$ "neuron"
- Binary signal $h(Wc_x_i)$ sent along "axon"
- Neuron aggregates signals $Wc_x_i$
- "Dendrites" for $z_{ik}$ "neuron" are receiving $x_{ij}$ values
“Artificial” Neural Nets vs. “Real” Networks Nets

• Artificial neural network:
  – $x_i$ is measurement of the world.
  – $z_i$ is internal representation of world.
  – $y_i$ is output of neuron for classification/regression.

• Real neural networks are more complicated:
  – **Timing** of action potentials seems to be important.
    • “Rate coding”: frequency of action potentials simulates continuous output.
  – Neural networks don’t reflect **sparsity** of action potentials.
  – How much computation is done **inside neuron**?
  – Brain is highly **organized** (e.g., substructures and cortical columns).
  – Connection **structure changes**.
  – **Different types** of neurotransmitters.
Deep Hierarchies in the Brain

DEEP HIERARCHIES IN THE VISUAL SYSTEM

<table>
<thead>
<tr>
<th>LAYER</th>
<th>PHOTORECEPTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retina</td>
<td></td>
</tr>
<tr>
<td>Thalamus</td>
<td></td>
</tr>
<tr>
<td>Thalamus</td>
<td>LGN</td>
</tr>
<tr>
<td>Thalamus</td>
<td>Lateral geniculate nucleus</td>
</tr>
<tr>
<td>Thalamus</td>
<td>Simple cell</td>
</tr>
<tr>
<td>Thalamus</td>
<td>Complex cell</td>
</tr>
<tr>
<td>Thalamus</td>
<td>Texture-defined contours</td>
</tr>
<tr>
<td>Thalamus</td>
<td>Nonspecific contours</td>
</tr>
<tr>
<td>Thalamus</td>
<td>Ownership</td>
</tr>
<tr>
<td>Thalamus</td>
<td>V4</td>
</tr>
<tr>
<td>Thalamus</td>
<td>V1</td>
</tr>
<tr>
<td>Thalamus</td>
<td>V2</td>
</tr>
<tr>
<td>thalamus</td>
<td>V3</td>
</tr>
<tr>
<td>thalamus</td>
<td>V4</td>
</tr>
<tr>
<td>thalamus</td>
<td>V1</td>
</tr>
</tbody>
</table>

http://www.strokenetwork.org/newsletter/articles/vision.htm
Deep Learning

Neural network:

Deep learning:

Second "layer" of latent features

You can add more "layers" to go "deeper"
Deep Learning

Linear model:
\[ y_i = w^T x_i \]

Neural network with 1 hidden layer:
\[ y_i = w^T h(Wx_i) \]

Neural network with 2 hidden layers:
\[ y_i = w^T h(W^{(2)} h(W^{(1)} x_i)) \]

Neural network with 3 hidden layers:
\[ y_i = w^T h(W^{(3)} h(W^{(2)} h(W^{(1)} x_i))) \]

Second "layer" of latent features
You can add more "layers" to go "deeper"
Cool Picture Motivation for Deep Learning

Deep Learning learns layers of features
Cool Picture Motivation for Deep Learning

• First layer of $z_i$ trained on 10 by 10 image patches:

• Attempt to visualize second layer:
  – Corners, angles, surface boundaries?

• Models require many tricks to work.
  – We’ll discuss these next time.
Cool Picture Motivation for Deep Learning

• First layer of $z_i$ trained on 10 by 10 image patches:

• Visualization of second and third layers trained on specific objects:

http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf
Cool Picture Motivation for Deep Learning

• First layer of $z_i$ trained on 10 by 10 image patches:

  ![First layer visualization](http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf)

• Visualization of second and third layers trained on specific objects:

  ![Second and third layer visualizations](http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf)
Cool Picture Motivation for Deep Learning

• First layer of $z_i$ trained on 10 by 10 image patches:

• Visualization of second and third layers trained on specific objects:

http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf
Cool Picture Motivation for Deep Learning

• First layer of $z_i$ trained on 10 by 10 image patches:

• Visualization of second and third layers trained on specific objects:

http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf
Cool Picture Motivation for Deep Learning

- First layer of \( z_i \) trained on 10 by 10 image patches:

- Visualization of second and third layers trained on specific objects:

http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf
ML and Deep Learning History

• 1950 and 1960s: Initial excitement.
  – **Perceptron**: linear classifier and stochastic gradient (roughly).
  – “the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.” New York Times (1958).
    • [https://www.youtube.com/watch?v=IEFRtz68m-8](https://www.youtube.com/watch?v=IEFRtz68m-8)

• Then drop in popularity:
  – Quickly realized **limitations of linear models**.
ML and Deep Learning History

• 1970 and 1980s: **Connectionism** (brain-inspired ML)
  - Connected networks of simple units.
    • Use parallel computation and distributed representations.
  - Adding hidden layers $z_i$ increases expressive power.
    • With 1 layer and enough sigmoid units, a universal approximator.
  - Success in optical character recognition.
ML and Deep Learning History

• 1990s and early-2000s: drop in popularity.
  – It proved really difficult to get multi-layer models working robustly.
  – We obtained similar performance with simpler models:
    • Rise in popularity of logistic regression and SVMs with regularization and kernels.
  – ML moved closer to other fields (CPSC 540):
    • Numerical optimization.
    • Probabilistic graphical models.
    • Bayesian methods.
ML and Deep Learning History

• Late 2000s: push to revive connectionism as “deep learning”.
  – Canadian Institute For Advanced Research (CIFAR) NCAP program:
    • “Neural Computation and Adaptive Perception”.
    • Led by Geoff Hinton, Yann LeCun, and Yoshua Bengio (“Canadian mafia”).
  – Unsupervised successes: “deep belief networks” and “autoencoders”.
    • Could be used to initialize deep neural networks.
    • https://www.youtube.com/watch?v=KuPai0ogiHk
2010s: DEEP LEARNING!!!

- Bigger datasets, bigger models, parallel computing (GPUs/clusters).
  - And some tweaks to the models from the 1980s.
  - All phones now have deep learning.
- Huge improvements in computer vision (2012).
  - Changed computer vision field almost instantly
  - This is now finding its way into products.

http://www.image-net.org/challenges/LSVRC/2014/
2010s: DEEP LEARNING!!!

• Media hype:
  – “Why Facebook is teaching its machines to think like humans” Wired (2013).

• 2015: huge improvement in language understanding.
ImageNet Challenge

- Millions of labeled images, 1000 object classes.

http://www.image-net.org/challenges/LSVRC/2014/
ImageNet Challenge

- Object detection task:
  - Single label per image.
  - Humans: ~5% error.

https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements/
ImageNet Challenge

• Object detection task:
  – Single label per image.
  – Humans: ~5% error.

https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements/
ImageNet Challenge

- Object detection task:
  - Single label per image.
  - Humans: ~5% error.

https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements/
ImageNet Challenge

- Object detection task:
  - Single label per image.
  - Humans: ~5% error.

https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements/
Object detection task:
- Single label per image.
- Humans: ~5% error.

ImageNet Challenge

GoogLeNet: 6.7% error
22 layers
ImageNet Challenge

- Object detection task:
  - Single label per image.
  - Humans: ~5% error.

- 2015 winner: Microsoft
  - 3.6% error.
  - 152 layers.

2010–2014 ILSVRC year

Switch to deep learning (8 layers)

More tweaks

Deeper!

GoogleNet: 6.7% error
22 layers
Adding a Bias Variable

Remember that in linear models we may have non-zero $y$-intercept:

$$ y_i = w^\top x_i + \beta $$

For neural networks we could have explicit bias:

$$ y_i = w^\top h(Wx_i) + \beta $$

We can just use $y_i = w^\top x_i$ if we fix $x_{ij} = 1$ for all $i$ for some $j$.

(constant $x_{ij}$ value)

Or we could set $W_c = 0$ for one row $W_c$ of $W$.

$$ \frac{1}{1 + \exp(-W_c x_i)} = \frac{1}{1 + \exp(0)} = \frac{1}{2} $$

(constant $Z_i$ value)
Artificial Neural Networks

• With squared loss, our objective function is:

\[
\mathcal{J}(w, W) = \frac{1}{2} \sum_{i=1}^{n} (w^\top h(Wx_i) - y_i)^2
\]

• Usual training procedure: **stochastic gradient**.
  – Compute gradient of random example ‘i’, update both ‘w’ and ‘W’.
  – Highly non-convex and can be difficult to tune.

• Computing the gradient is known as “backpropagation”.
Backpropagation

- Consider the loss for a single example:
  \[
  f(w, W) = \frac{1}{2} \left( \sum_{c=1}^{K} \begin{bmatrix} w_c \end{bmatrix} h(W_c x_i) - y_i \right)^2
  \]

- Derivative with respect to \(w_c\):
  \[
  \frac{2}{2w_c} \left[ f(w, W) \right] = \left( \sum_{c=1}^{K} \begin{bmatrix} w_c \end{bmatrix} h(W_c x_i) - y_i \right) h(W_c x_i)
  \]

- Derivative with respect to \(W_{cj}\):
  \[
  \frac{2}{2w_{cj}} \left[ f(w, W) \right] = \left( \sum_{c=1}^{K} \begin{bmatrix} w_c \end{bmatrix} h(W_c x_i) - y_i \right) w_c h'(W_c x_i) x_{ij}
  \]
Backpropagation

• Notice repeated calculations in gradients:

\[ \frac{∂^2}{∂w_c}[f(w, W)] = (\sum_{c=1}^{k} w_c h(W_c x_i) - y_i) h(W_c x_i) \]

\[ \overset{r_i}{\downarrow} \]

\[ = r_i h(W_c x_i) \]

\[ \overset{\text{same } r_i \text{ for all } 'c'}{\downarrow} \]

\[ 2 \frac{∂}{∂w_j}[f(w, W)] = (\sum_{c=1}^{k} w_c h(W_c x_i) - y_i) w_c h'(W_c x_i) x_{ij} \]

\[ \overset{r_i}{\downarrow} \]

\[ = r_i v_c x_{ij} \]

\[ \overset{\text{same } v_c \text{ for all } 'j'}{\downarrow} \]

\[ \overset{\text{same } r_i \text{ for all } 'c'}{\downarrow} \]
Backpropagation

- Calculation of gradient is split into two phases:
  1. "Forward" pass
     (a) Compute $h(W_c x_i)$ for all $c$
     (b) Compute residual $r_i = (\sum_{c_i} w_{ci} h(W_c x_i) - y_i)$
  2. "Backpropagation"
     (a) Compute $\frac{2f}{2w_c} = r_i h(W_c x_i)$ for all $c$
     (b) Compute $v_c = w_c h'(W_c x_i)$ for all $c$
     (c) Compute $\frac{2f}{2w_{cij}} = r_i v_c x_{ij}$ for all $c$ and $j$
Summary

• Biological motivation for (deep) neural networks.
• Deep learning considers neural networks with many hidden layers.
• Unprecedented performance on difficult pattern recognition tasks.
• Backpropagation computes neural network gradient via chain rule.

• Next time:
  – How deep learners fight the fundamental trade-off.