Feedback from surveys

- Still a little too fast
- Breakout rooms aren’t working
- What should I know before each lecture?
- Use Friday’s more for review/reinforcement
  - And HW
- Organize quizzes better on canvas
- HW2: ….
Neural Networks: Deep Learning
Lyle Ungar

Multilevel network: architecture, link functions
CNNs: local receptive fields, max pooling

----------------------------------
Regularization: \( L_2 \), early stopping, dropout
Gradient Descent (again)
Semi-supervised and transfer learning
Quick tensor background

What’s a tensor?

- As in “tensorflow”
- Or “Tensor Processing Unit” (TPU)
- As in the basic data structure in pytorch
  - Aside: there is a worksheet with more than you need to know about pytorch
All machine learning is optimization

\[ \hat{y} = f(x; \theta) \]
\[ \arg\min_{\theta} ||y - \hat{y}|| \]

So what’s new this decade?

(Slightly) different loss functions
(Slightly) different optimization methods
GPU instead of CPU
Different, flexible, functional forms for \( f \)
which require regularization
Loss functions

\[ \hat{y} = f(x; \theta) \]

\[ \arg\min_\theta \| y - \hat{y} \| \]

\[ \| y - \hat{y} \|_2 \quad \| y - \hat{y} \|_1 \quad \| y - \hat{y} \|_0 \]

log-likelihood

cross-entropy (KL-divergence)

later in the course: hinge, exponential
Flexible model forms

\[ \hat{y} = f(x; \theta) \]

\( X \)
- Web page, ad
- Past purchases, ...
- Facebook posts

\( y \)
- Click on ad?
- NPV
- Age, Sex, Personality, …
Flexible model forms

X
biopsy image

y
Cancer present?
Flexible model forms

$X$  $y$

Camera image  Objects in it
<table>
<thead>
<tr>
<th>English sentence</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>I love machine learning</td>
<td>أحب تعلم الآلة</td>
</tr>
<tr>
<td>'uhibb taelam alala</td>
<td></td>
</tr>
</tbody>
</table>
Artificial Neural Nets

◆ Semi-parametric
  ● Flexible model form

◆ Used when there are vast amounts of data
  ● Hence popular (again) now
  ● But recently with smaller training sets.

◆ Deep networks
  ● Idea: representation should have many different levels of abstraction
Neural Nets can be

- **Supervised**
  - Generalizes *logistic regression* to a semi-parametric form
- **Unsupervised**
  - Generalizes *PCA* to a semi-parametric form
- **Adversarial**
- **Semi-supervised**
- **Reinforcement**

*Neural nets often have built-in structure*
“Real” and Artificial neuron

One neuron does logistic regression

\[ h_{w,b}(x) = f(w^T x + b) \]

\[ f(z) = \frac{1}{1 + e^{-z}} \]

Socher and Manning tutorial
Neural nets stack logistic regressions

Every line represents a parameter in the model.
Neural nets stack logistic regressions

Every line represents a parameter in the model
Training

- Mini-batch gradient descent
- “Backpropagate” error derivatives through the model = chain rule
ANNs do pattern recognition

- Map input “percepts” to output categories or actions
  - Image of an object → what it is
  - Image of a person → who it is
  - Picture → caption describing it
  - Board position → probability of winning
  - A word → the sound of saying it
  - Sound of a word → the word
  - Sequence of words in English → their Chinese translation
# MNIST

- Classify 28x28 images of handwritten digits
- **Train:** 50,000
- **Test:** 10,000

<table>
<thead>
<tr>
<th>Error (%)</th>
<th>Method</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.0</td>
<td>KNN</td>
<td>Lecun et al. (1998)</td>
</tr>
<tr>
<td>3.6</td>
<td>1k RBF + linear classifier</td>
<td>Lecun et al. (1998)</td>
</tr>
<tr>
<td>1.6</td>
<td>2-layer NN</td>
<td>Simard et al. (2003)</td>
</tr>
<tr>
<td>1.53</td>
<td>boosted stumps</td>
<td>Kegl et al. (2009)</td>
</tr>
<tr>
<td>1.4</td>
<td>SVM</td>
<td>Lecun et al. (1998)</td>
</tr>
<tr>
<td>0.79</td>
<td>DNN</td>
<td>Srivastava (2013)</td>
</tr>
<tr>
<td>0.45</td>
<td>conv-DNN</td>
<td>Goodfellow et al. (2013)</td>
</tr>
<tr>
<td>0.21</td>
<td>conv-DNN</td>
<td>Wi et al. (2013)</td>
</tr>
</tbody>
</table>
Street View House Numbers

- Classify 32x32 color images of digits
- Digits taken from house numbers in Google Street View
- **Train**: 604,388
- **Test**: 26,032

<table>
<thead>
<tr>
<th>Error (%)</th>
<th>Method</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.7</td>
<td>WDCH</td>
<td>Netzer et al. (2011)</td>
</tr>
<tr>
<td>15</td>
<td>HOG</td>
<td>Netzer et al. (2011)</td>
</tr>
<tr>
<td>9.4</td>
<td>KNN</td>
<td>Netzer et al. (2011)</td>
</tr>
<tr>
<td>2.47</td>
<td>conv-DNN</td>
<td>Goodfellow et al. (2013)</td>
</tr>
<tr>
<td>2</td>
<td>Human</td>
<td>Netzer et al. (2013)</td>
</tr>
<tr>
<td>1.92</td>
<td>conv-DNN</td>
<td>Lee et al. (2015)</td>
</tr>
</tbody>
</table>
CIFAR-100

- Classify 32x32 color images into 100 classes
- Images taken from TinyImages dataset at MIT
- **Train:** 50,000
- **Test:** 10,000

<table>
<thead>
<tr>
<th>Error (%)</th>
<th>Method</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>43.77</td>
<td>SVM</td>
<td>Jia et al. (2012)</td>
</tr>
<tr>
<td>39.20</td>
<td>OMP</td>
<td>Lin and Kung (2014)</td>
</tr>
<tr>
<td>38.57</td>
<td>conv-DNN</td>
<td>Goodfellow et al. (2013)</td>
</tr>
<tr>
<td>36.18</td>
<td>DNN</td>
<td>Srivastava and Alakhutdinov (2015)</td>
</tr>
<tr>
<td>34.57</td>
<td>conv-DNN</td>
<td>Lee et al. (2015)</td>
</tr>
</tbody>
</table>
ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
Ilya Sutskever
Geoffrey Hinton

University of Toronto
Canada

“AlexNet” 2012
Neural networks

- A neuron

\[ x = w_1 f(z_1) + w_2 f(z_2) + w_3 f(z_3) \]

\( x \) is called the total input to the neuron, and \( f(x) \) is its output

- A neural network

A neural network computes a differentiable function of its input. For example, ours computes: \( p(\text{label} \mid \text{an input image}) \)
Neurons

Traditional: sigmoidal e.g. logistic function

\[ f(x) = \tanh(x) \]

Hyperbolic tangent

Very bad (slow to train)

But one can use any nonlinear function

\[ f(x) = \max(0, x) \]

Rectified Linear Unit (ReLU)

Very good (quick to train)

\[ x = w_1 f(z_1) + w_2 f(z_2) + w_3 f(z_3) \]

\( x \) is called the total input to the neuron, and \( f(x) \) is its output.
Overview of our model

- **Deep**: 7 hidden “weight” layers
- **Learned**: all feature extractors initialized at white Gaussian noise and learned from the data
- Entirely supervised
- **More data = good**

**Convolutional layer**: convolves its input with a bank of 3D filters, then applies point-wise non-linearity

**Fully-connected layer**: applies linear filters to its input, then applies point-wise non-linearity
Local receptive fields

In vision, a neuron may only get inputs from a limited set of “nearby” neurons.
Local receptive fields

• spatial extent $F = 3$
• stride $S = 1$

• spatial extent $F = 3$
• stride $S = 2$

http://cs231n.github.io/convolutional-networks/
Local receptive fields

http://cs231n.github.io/convolutional-networks/
Local pooling

Max-pooling partitions the input image into local regions and outputs the maximum value for each.

Reduces the computational complexity
Provides translation invariance.
Max pooling

Spatial extent $F=2$

Stride $S=2$

http://cs231n.github.io/convolutional-networks/
Our model

- Max-pooling layers follow first, second, and fifth convolutional layers
- The number of neurons in each layer is given by 253440, 186624, 64896, 64896, 43264, 4096, 4096, 1000
Overview of our model

- Trained with stochastic gradient descent on two NVIDIA GPUs for about a week
- 650,000 neurons
- 60,000,000 parameters
- 630,000,000 connections
- **Final feature layer:** 4096-dimensional

**Convolutional layer:** convolves its input with a bank of 3D filters, then applies point-wise non-linearity

**Fully-connected layer:** applies linear filters to its input, then applies point-wise non-linearity
Training

Using stochastic gradient descent and the backpropagation algorithm (just repeated application of the chain rule)

One output unit per class
\[ x_i = \text{total input to output unit } i \]
\[ f(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{1000} \exp(x_j)} \]
We maximize the log-probability of the correct label, \( \log f(x_t) \)
\[ f(x_i) = \text{softmax} \]
\[ \log(f(x_t)) = \text{cross-entropy} \]
Data augmentation

- Our neural net has 60M real-valued parameters and 650,000 neurons
- It overfits a lot. Therefore we train on 224x224 patches extracted randomly from 256x256 images, and also their horizontal reflections.