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 + Adagrad)
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

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

argmin\( \theta \) \[ ||y - \hat{y}|| \]

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
Camera image

y
Objects in it

nvidia
Flexible model forms

\( X \)

**English sentence**

I love machine learning

\( y \)

**Translation**

أحب تعلم الآلة

'uhibb taelam alala
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>Kegel 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 $S = 2$

[Image of convolutional network diagram]

Local receptive fields

Input Volume (+pad 1) (7x7x3)

Filter W0 (3x3x3)

Filter W1 (3x3x3)

Output Volume (3x3x2)

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.
Questions?
Neural Networks: Deep Learning (2)

Lyle Ungar

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

----------------------------------

Regularization: $L_2$, early stopping, dropout
Gradient Descent (again + Adgrad)
Semi-supervised and transfer learning
Modern deep nets

- Often use rectified linear units (ReLUs)
  - Faster, less problems of saturation than logistic
- Use a variety of loss functions
  - Cross-entropy with softmax
  \[ \sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for } j = 1, \ldots, K. \]
- Can be very deep
- Solved with mini-batch gradient descent
- Regularized using L_1 and L_2 penalty plus “dropout”
  - and partial convergence and ..
Regularization

- $L_2$ and/or $L_1$
- Early stopping
- Max norm ($L_\infty$)
  - Weight clipping
  - Gradient clipping
- Dropout

Dropout

- Randomly (temporarily) remove a fraction $\rho$ of the nodes (with replacement)
  - Usually, $\rho = 1/2$
- Repeatedly doing this samples (in theory) over exponentially many networks
  - Bounces the network out of local minima
- For the final network use all the weights, but shrink them by $\rho$
Gradient descent

- Gradient descent
  - Minibatch
  - Gradient clipping
- Momentum
- Learning rate adaptation
  - Adagrad and friends

\[
\frac{\delta Err}{\delta w} = \frac{Err(w+h) - Err(w-h)}{2h}
\]

\[
\Delta w^t = \eta \frac{\delta Err}{\delta w} + m \Delta w^{t-1}
\]
Learning rate adaption

- Adjust the learning rate over time

\[ \Delta w^t = \eta(t) \frac{\delta Err}{\delta w} \]

- Adagrad: make the learning rate depend on previous changes in each weight
  - increases the learning rate for more sparse parameters and decreases the learning rate for less sparse ones.

\[ \Delta w^t_j = \frac{\eta}{||\delta w^t_j||_2} \frac{\delta Err}{\delta w_j} \]
Learning rate

![Graph showing learning rate comparison](http://cs231n.github.io/neural-networks-3/)
Feature Scaling (standardizing)

- **Idea:** Ensure that features have similar scales

Before Feature Scaling

After Feature Scaling

- Can do this for hidden layer outputs as well for each minibatch
- Makes gradient descent converge much faster

Is deep learning scale invariant?
Lots of fancy network structures

- Convolutional (different sizes)
- Or fully connected
- Maxpool
- Concatenation
- Softmax
- Some layers use dropout
- googlenet
Validation classification

<table>
<thead>
<tr>
<th>mite</th>
<th>container ship</th>
<th>motor scooter</th>
<th>leopard</th>
</tr>
</thead>
<tbody>
<tr>
<td>black widow</td>
<td>lifeboat</td>
<td>go-kart</td>
<td>jaguar</td>
</tr>
<tr>
<td>cockroach</td>
<td>amphibian</td>
<td>moped</td>
<td>cheetah</td>
</tr>
<tr>
<td>tick</td>
<td>fireboat</td>
<td>bumper car</td>
<td>snow leopard</td>
</tr>
<tr>
<td>starfish</td>
<td>drilling platform</td>
<td>golfcart</td>
<td>Egyptian cat</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>grille</th>
<th>mushroom</th>
<th>cherry</th>
<th>Madagascar cat</th>
</tr>
</thead>
<tbody>
<tr>
<td>convertible</td>
<td>agaric</td>
<td>dalmatian</td>
<td>squirrel monkey</td>
</tr>
<tr>
<td>grille</td>
<td>mushroom</td>
<td>grape</td>
<td>spider monkey</td>
</tr>
<tr>
<td>pickup</td>
<td>jelly fungus</td>
<td>elderberry</td>
<td>titi</td>
</tr>
<tr>
<td>beach wagon</td>
<td>gill fungus</td>
<td>ffordshire bullterrier</td>
<td>indri</td>
</tr>
<tr>
<td>fire engine</td>
<td>dead-man's-fingers</td>
<td>currant</td>
<td>howler monkey</td>
</tr>
</tbody>
</table>
Validation classification
Validation localizations

chime
- wine bottle
- digital clock
- gar
- oboe
- typewriter
- keyboard

boathouse
- apiary
- mobile home
- boathouse
- picket fence
- patio

Scottish deerhound
- Scottish deerhound
- Irish wolfhound
- Leonberg
- German shepherd
- Tibetan mastiff

electric guitar
- violin
- carpenter's kit
- revolver
- Loafer
- corkscrew

motor scooter
- motor scooter
- moped
- snowmobile
- police van
- moving van

sturgeon
- leatherback turtle
- volcano
- wreck
- alp
- breakwater

violin
- violin
- cello
- acoustic guitar
- drumstick
- electric guitar

fire screen
- fire screen
- sundial
- mailbag
- umbrella
- purse
Retrieval experiments

First column contains query images from ILSVRC-2010 test set, remaining columns contain retrieved images from training set.
Now used for image search; Benefit: Good Generalization

Both recognized as “meal”

Jeff Dean, google
Sensible Errors (sometimes)

“snake”

“dog”

Jeff Dean, google
Now used for image search

Works in practice... for real users

Wow.
The new Google plus photo search is a bit insane.
I didn’t tag those... 😊

Jeff Dean, google
Now used for image search

Works in practice... for real users

Google Plus photo search is awesome. Searched with keyword 'Drawing' to find all my scribbles at once :D

Jeff Dean, google
Questions?
Semi-supervised learning

- Use unlabeled data $X_0$ to derive new features $z = \phi(x)$
- Train $y = f(\phi(x), w)$
Transfer learning

- Use one data set \((X_0, y_0)\) to train a model
- Find feature transformations \(\phi(x)\)
- Use those transformations \(\phi(x)\) on data from data set with a different label, \(y\).
Transfer learning for NNets

Data set for learning $\phi(x)$

Data set with target $y$. 

$x_0$, $y_0$

$x$, $y$
New AI can guess whether you're gay or straight from a photograph

An algorithm deduced the sexuality of people on a dating site with up to 91% accuracy, raising tricky ethical questions.
Deep neural networks are more accurate than humans at detecting sexual orientation from facial images.

Michal Kosinski & Yilun Wang

We show that faces contain much more information about sexual orientation than can be perceived and interpreted by the human brain. We used deep neural networks to extract features from 35,326 facial images. These features were entered into a logistic regression aimed at classifying sexual orientation. Given a single facial image, a classifier could correctly distinguish between gay and heterosexual men in 81% of cases, and in 74% of cases for women. Human judges achieved much lower accuracy: 61% for men and 54% for women. The accuracy of the algorithm increased to 91% and 83%, respectively, given five facial images per person. Facial features employed by the classifier included both fixed (e.g., nose shape) and transient facial features (e.g., grooming style). Consistent with the prenatal hormone theory of sexual orientation, gay men and women tended to have gender-atypical facial morphology, expression, and grooming styles. Additionally, given that companies and governments are increasingly using computer vision algorithms to detect people’s intimate traits, our findings expose a threat to the privacy and safety of gay men and women.

https://osf.io/fk3xr/ 2017
Deep learning case study

- Download images and labels from a dating site
  - where people declare their sexual orientation
- Only keep images with a single “good” face
  - Use Face++ to identify faces -- yielded 35,000 faces
- Use M-turkers to QC & restrict to Caucasians
- Use pretrained CNN to compute ~ 4,000 ‘scores’/image
  - VGG-Face was trained on 2.6 million faces
- Use logistic regression on SVD of the 4,000 scores
  - report cross-validation error predicting gay/straight
What you should know

- CNN
  - local receptive field, max pooling
- Rectified Linear Unit (ReLU)
- Dropout
- Back-propagation, momentum, Mini-batch
- At least four kinds of regularization
- Transfer learning
Questions?