Deep Learning: Searching for Images

Retrieving similar images

Input Image

Nearest neighbors
Neural networks

↓

Learning *very* non-linear features

Perceptron as a neural network

This is one neuron:
- Input edges $x[1], \ldots, x[d]$, along with intercept $x[0] = 1$
- Sum passed through an activation function $g$
Composing individual neurons

Hidden layer

Single unit:

$$out(x) = g(w_0 + \sum_j w_j x[j])$$

1-hidden layer:

$$out(x) = g(w_0 + \sum_k w_k g(w_0^k + \sum_j w_j^k x[j]))$$

No longer convex function!
Some choices of activation function $g$

- **Sigmoid**
  - Historically popular, but (mostly) fallen out of favor
  - Neuron’s activation saturates
    - Weights get very large $\rightarrow$ gradients get small
  - Not zero-centered $\rightarrow$ other issues in the gradient steps
  - When put on the output layer, called “softmax” because interpreted as class probability (soft assignment)

- **Hyperbolic tangent** $g(x) = \tanh(x)$
  - Saturates like sigmoid unit, but zero-centered

- **Rectified linear unit (ReLU)** $g(x) = x^+ = \max(0,x)$
  - Most popular choice these days
  - Fragile during training and neurons can “die off”...
  - Be careful about learning rates
  - “Noisy” or “leaky” variants

- **Softplus** $g(x) = \log(1+\exp(x))$
  - Smooth approximation to rectifier activation

---

Application of deep learning to computer vision
Standard image classification approach

Input → Extract features → Use simple classifier

- Hand-created features
- e.g., logistic regression, SVMs

Face?

Deep learning: *implicitly learns features*

Layer 1 → Layer 2 → Layer 3 → Prediction

Example detectors learned
Example interest points detected

[Zeiler & Fergus '13]
The challenge of applying regular neural networks (multilayer perceptrons) to images

- Images are **high-dimensional** inputs!
  - CIFAR-10 images are **small**: $32 \times 32 \times 3$ (pixel x pixel x 3 color channels)
    - # of 1st layer weights:

- If $200 \times 200 \times 3$:

- Images are **structured** inputs...leverage this!

Convolution networks (ConvNets)

Arrange neurons in 3D

- Not fully connected!
  - Each neuron is connected to a small region of previous layer

Vector of class scores
ConvNet components: Convolutional layers

Apply learned filters over input volume:

Parameter sharing!
(same weights applied to all x,y of a depth slice)

ConvNet components: Convolutional layers

For one filter:

Stride and zero-padding:

gifs from:
http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html
ConvNet components: 
**Pooling layers**

- Downsampling the spatial dims
- Common to insert between successive conv layers
- Typically max pooling of size 2x2 with stride 2
  - Applied separately to each depth slice
  - Tends to work better than average pooling

ConvNet components: 
**Fully connected layers**

Same as the neural networks presented earlier
- Used, e.g., to compute the class scores (output)
Returning to our example...
“Detectors” are the learned filters

Example detectors learned

Example interest points detected

Layer 1  Layer 2  Layer 3  Prediction

Image classification

Input: \( \mathbf{x} \)  Image pixels

Output: \( \mathbf{y} \)  Predicted object

Top Predictions
- Labrador retriever
- golden retriever
- redbone
- bloodhound
- Rhodesian ridgeback

Zeiler & Fergus '13
Scene parsing with deep learning

[Farabet et al. '13]

Object detection

Redmon et al. 2015
Retrieving similar images

Input Image  Nearest neighbors

Deep learning performance
Sample results using deep neural networks

German traffic sign recognition benchmark
- 99.5% accuracy (IDSIA team)

House number recognition
- 97.8% accuracy per character [Goodfellow et al. ‘13]

ImageNet 2012 competition:
1.2M training images, 1000 categories

Top 3 teams

Exploited hand-coded features like SIFT
ImageNet 2012 competition: 1.2M training images, 1000 categories

Winning entry: SuperVision
8 layers, 60M parameters [Krizhevsky et al. '12]

Achieving these amazing results required:
• New learning algorithms
• GPU implementation

Going even deeper...

Won 2014 ImageNet challenge with 6.66% top-5 error rate

Huge CNN depth has proven helpful in recognition systems... Maybe because images contain hierarchical structure (faces contain eyes contain edges, etc.)
Challenges of deep learning

Deep learning score card

**Pros**

- Enables learning of features rather than hand tuning
- Impressive performance gains
  - Computer vision
  - Speech recognition
  - Some text analysis
- Potential for more impact
Deep learning workflow

Lots of labeled data → Training set → Learn deep neural net → Validate → Validation set → Adjust parameters, network architecture,...

Many tricks needed to work well...

Different types of layers, connections,... needed for high accuracy

[Krizhevsky et al. ’12]
## Deep learning score card

**Pros**
- Enables learning of features rather than hand tuning
- Impressive performance gains
  - Computer vision
  - Speech recognition
  - Some text analysis
- Potential for more impact

**Cons**
- Requires a lot of data for high accuracy
- Computationally really expensive
- Extremely hard to tune
  - Choice of architecture
  - Parameter types
  - Hyperparameters
  - Learning algorithm

\[
\text{Computational cost + so many choices} = \text{incredibly hard to tune}
\]

---

**Deep features:**

Deep learning  
+  
Transfer learning
Standard image classification approach

Input

Extract features

Can we learn features from data, even when we don’t have data or time?

Use simple classifier
e.g., logistic regression, SVMs

Face?

Transfer learning: Use data from one task to help learn on another
Old idea, explored for deep learning by Donahue et al. ’14 & others

Lots of data:

Learn neural net

Great accuracy on cat v. dog

Some data:

Neural net as feature extractor + Simple classifier

Great accuracy on 101 categories
What’s learned in a neural net

Neural net trained for Task 1: cat vs. dog

More generic
Can be used as feature extractor

Very specific
to Task 1
Should be ignored for other tasks

Transfer learning in more detail...

For Task 2, predicting 101 categories,
learn only end part of neural net

Neural net trained for Task 1: cat vs. dog

Keep weights fixed!

More generic
Can be used as feature extractor

Very specific
to Task 1
Should be ignored for other tasks

Use simple classifier
e.g., logistic regression,
SVMs, nearest neighbor,...
Careful where you cut: *latter layers may be too task specific*

Example detectors learned

Example interest points detected

[Zeiler & Fergus '13]

Transfer learning with deep features workflow

Some labeled data

Extract features with neural net trained on different task

Training set

Learn simple classifier

Validation set

Validate
How general are deep features?

compology

Summary of deep learning
What you can do now...

- Describe multilayer neural network models
- Motivate why 2-layer neural networks can be universal function approximators
- Build convolutional neural networks using various components, including convolutional layers, pooling, and fully connected layers
- Compare and contrast ConvNets with standard multilayer neural networks
- Interpret the role of features as local detectors in computer vision
- Relate neural networks to hand-crafted image features
- Describe some settings where deep learning achieves significant performance boosts
- State the pros & cons of deep learning model
- Apply the notion of transfer learning
- Use neural network models trained in one domain as features for building a model in another domain
- Build an image retrieval tool using deep features
Course Wrap-Up

STAT/CSE 416: Intro to Machine Learning
Emily Fox
University of Washington
May 31, 2018

What you have learned this quarter

- Regression
- Overfitting
- Training, test, and generalization error
- Bias-Variance tradeoff
- Ridge, LASSO
- Cross validation
- Gradient descent
- Classification
- Logistic regression
- Decision trees
- Boosting
- Precision and recall
- Nearest-neighbor retrieval, regression, and classification
- Kernel regression
- Locality sensitive hashing
- Dimensionality reduction, PCA
- k-means clustering
- Hierarchical clustering
- Unsupervised v. supervised learning
- Recommender systems
- Matrix factorization
- Coordinate descent
- Neural networks
- Convolutional neural networks
- Transfer learning for deep learning
Case Study 1: Predicting house prices

Regression
Case study: Predicting house prices

Models
- Linear regression
- Regularization: Ridge (L2), Lasso (L1)

Including many features:
- Square feet
- # bathrooms
- # bedrooms
- Lot size
- Year built
- …
Regression
Case study: Predicting house prices

Algorithms
• Gradient descent

\[
\text{RSS}(w_0, w_1) = \sum \left( y - (w_0 + w_1 \times \text{sq.ft.}) \right)^2
\]

Regression
Case study: Predicting house prices

Concepts
• Loss functions, bias-variance tradeoff, cross-validation, sparsity, overfitting, model selection

1. Noise
2. Bias
3. Variance
Case Study 2:
Sentiment analysis

Sushi was awesome, the food was awesome, but the service was awful.

All reviews:

- Score(x) > 0
- Score(x) < 0

Classification
Case study: Analyzing sentiment

Models
- Linear classifiers (logistic regression)
- Multiclass classifiers
- Decision trees
- Boosted decision trees and random forests
3. Classification

**Case study: Analyzing sentiment**

### Algorithms
- Boosting
- Learning from weighted data

### Concepts
- Decision boundaries, maximum likelihood estimation, ensemble methods, random forests
- Precision and recall

---

**Weighted error**
- weighted_error = 0.2
- weighted_error = 0.35
- weighted_error = 0.3
- weighted_error = 0.4

---

**Classification**

**Case study: Analyzing sentiment**

Squeezing last bit of accuracy by blending models

---

**Netflix Prize**

Leaderboard: 10.05%

Classifier A
- Best classifier
- Classifier B
Case Study 3:
Document retrieval

Data → Nearest neighbor → Intelligence

Case Study 3+:
Document structuring for retrieval

Data → Clustering → Intelligence

Data types: SPORTS, WORLD NEWS, ENTERTAINMENT, SCIENCE
Case Study 3++: Dimensionality reduction

Images with thousands or millions of pixels

Can we give each image a coordinate, such that similar images are near each other?

Clustering & Retrieval

Case study: Finding documents

Models
- Nearest neighbors
- Clustering
- Hierarchical clustering
Clustering & Retrieval

Case study: Finding documents

- k-means
- Locality-sensitive hashing (LSH)
- NN regression and classification
- Kernel regression
- Agglomerative and divisive clustering
- PCA

Concepts

- Distance metrics, kernels, approximation algorithms, dimensionality reduction

Algorithms

- Distance metrics, kernels, approximation algorithms, dimensionality reduction

Principal components:

Reconstructing:
Case Study 4: 
Product recommendation

Your past purchases: 
+ purchase histories of all customers

Recommended items:

Recommender Systems & Matrix Factorization 
Case study: Recommending Products

Models
- Collaborative filtering
- Matrix factorization

Rating = \approx X_{ij} 
known for black cells
unknown for white cells

Rows index movies
Columns index users

Parameters of model
Recommender Systems & Matrix Factorization

Case study: Recommending Products

Algorithms
- Coordinate descent

$$\begin{align*}
\text{Rating} &= \begin{bmatrix}
L & R' \\
\end{bmatrix}
\end{align*}$$

Form estimates $$\hat{L}_u$$ and $$\hat{R}_v$$

=None

Recommender Systems & Matrix Factorization

Case study: Recommending Products

Concepts
- Matrix completion, cold-start problem

Customers
- Products

Customers
- Products
Case Study 5:
Visual product recommender

Deep Learning
Case study: Visual product recommender

Models
- Perceptron
- General neural network
- Convolutional neural network
Deep Learning

Case study: Visual product recommender

Algorithms
- Convolutions
- Backpropagation (high level only)

Concepts
- Activation functions, hidden layers, architecture choices

Forward
Gradient

Convolutions
Backpropagation

Activation functions:
- Sigmoid
- Hyperbolic tangent
- ReLU
Improving the performance at some task though experience!
– before you start any learning task, remember the fundamental questions:

What is the learning problem?  From what experience?  What model?

What loss function are you optimizing?  With what optimization algorithm?

With what guarantees?  How will you evaluate it?
Thank you for the hard work!!!