Neural Networks

CSE 446: Machine Learning
Emily Fox
University of Washington
March 10, 2017

Single-layer neural network
Perceptron as a neural network

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

\[ g = \begin{cases} 
1 & \text{if } \sum_{j=0}^{d} w_j \mathbf{x}[j] > 0 \\
-1 & \text{otherwise}
\end{cases} \]

Sigmoid neuron

Just change $g$!
- Why would we want to do this?
- Notice the output range $[0,1]$. What was it before?
- Look familiar?
Perceptron, linear classification, Boolean functions:

\[ x[j] \in \{0,1\} \]

- Can learn \( x[1] \lor x[2] \)?
  - \(-0.5 + x[1] + x[2] \)
- Can learn \( x[1] \land x[2] \)?
  - \(-1.5 + x[1] + x[2] \)
- Can learn any conjunction or disjunction?
  - \(0.5 + x[1] + \ldots + x[d] \)
  - \((-d+0.5) + x[1] + \ldots + x[d] \)
- Can learn majority?
  - \((-0.5*d) + x[1] + \ldots + x[d] \)
- What are we missing? The dreaded XOR!, etc.

Introducing a hidden layer
What can’t a simple linear classifier represent?

\[
\text{XOR} = x[1] \text{ AND NOT } x[2] \quad \text{OR} \quad \text{NOT } x[1] \text{ AND } x[2]
\]

Solving the XOR problem: Going beyond linear classification by adding a layer

\[
\text{Thresholded to 0 or 1}
\]
Solving the XOR problem: Going beyond linear classification by adding a layer

\[ y = x[1] \text{ XOR } x[2] = (x[1] \land \neg x[2]) \lor (x[2] \land \neg x[1]) \]

\[ v[1] = (x[1] \land \neg x[2]) \]
\[ = -0.5 + x[1] - x[2] \]

\[ v[2] = (x[2] \land \neg x[1]) \]
\[ = -0.5 + x[2] - x[1] \]

\[ y = v[1] \lor v[2] \]
\[ = -0.5 + v[1] + v[2] \]

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!
A general neural network

- Layers and layers and layers of linear models and non-linear transformations
- Around for about 50 years
  - Fell in “disfavor” in 90s
- In last few years, big resurgence
  - Impressive accuracy on several benchmark problems
  - Powered by huge datasets, GPUs, & modeling/learning alg improvements

Learning neural networks with hidden layers
Recall: Optimizing a single-layer neuron

We train to minimize sum of squared errors:

$$\ell(w) = \frac{1}{2} \sum_i[y_i - g(w_0 + \sum_j w_j x_i[j])]^2$$

Taking gradients:

$$\frac{\partial \ell}{\partial w_j} = -\sum_i [y_i - g(w_0 + \sum_j w_j x_i[j])] \frac{\partial}{\partial w_j} g(w_0 + \sum_j w_j x_i[j])$$

$$= -\sum_i [y_i - g(w_0 + \sum_j w_j x_i[j])] x_i[j] g'(w_0 + \sum_j w_j x_i[j])$$

Solution just depends on $g'$: derivative of activation function!

Forward propagation

1-hidden layer:

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

For fixed weights, forming predictions is easy!

Compute values left to right
1. Inputs: $x[1],...,x[d]$
2. Hidden: $v[1],...,v[d]$
3. Output: $y$
Gradient descent for 1-hidden layer: 

**Output layer parameters**

\[
\ell(w) = \frac{1}{2} \sum_i [y_i - \text{out}(x_i)]^2
\]

\[
\text{out}(x) = g(\sum_{k'} w_{k'} g(\sum_{j'} w_{k'j'} x[j'])) = g(\sum_{k'} w_{k'} v_i[k'])
\]

\[
\frac{\partial \ell}{\partial w_k} = \sum_{i=1}^{N} \left[ y_i - \text{out}(x_i) \right] \frac{\partial \text{out}(x_i)}{\partial w_k}
\]

Gradient for last layer same as single node case, but with hidden nodes \(v\) as input!

Dropped \(w_0\) to make derivation simpler

---

Gradient descent for 1-hidden layer: 

**Hidden layer parameters**

\[
\ell(w) = \frac{1}{2} \sum_i [y_i - \text{out}(x_i)]^2
\]

\[
\text{out}(x) = g(\sum_{k'} w_{k'} g(\sum_{j'} w_{k'j'} x[j']))
\]

\[
\frac{\partial \ell}{\partial w_{k_j}} = \sum_{i=1}^{N} \left[ y_i - \text{out}(x_i) \right] \frac{\partial \text{out}(x_i)}{\partial w_{k_j}}
\]

For hidden layer, two parts:

\[
\frac{\partial \text{out}(x_i)}{\partial w_{k_j}} = g' \left( \sum_{k'} w_{k'} g(\sum_{j'} w_{k'j'} x[j']) \right) \frac{\partial g(\sum_{k'} w_{k'} v_i[k'])}{\partial w_{k_j}}
\]

Recursive computation of gradient on output layer 

Normal update for single neuron

Dropped \(w_0\) to make derivation simpler
Multilayer neural networks

**Inference and Learning**

- **Forward pass:** left to right, each hidden layer in turn
- **Gradient computation:** right to left, propagating gradient for each node

**Forward propagation – Prediction**

- Recursive algorithm
- Start from input layer
- Output of node $v[k]$ with parents $u[1], u[2], \ldots$:

\[
v[k] = g \left( \sum_j w^k_j u[j] \right)
\]
Back-propagation – Learning

• Just gradient descent!!!
• Recursive algorithm for computing gradient
• For each example
  – Perform forward propagation
  – Start from output layer
    • Compute gradient of node \( v[k] \) with parents \( u[1], u[2], \ldots \):
    • Update weight \( w_{ij} \)
    • Repeat (move to preceding layer)

Convergence of backprop

Perceptron leads to convex optimization
  – Gradient descent reaches global minima
Multilayer neural nets not convex
  – Gradient descent gets stuck in local minima
  – Selecting number of hidden units and layers = fuzzy process
  – NNs have made a HUGE comeback in the last few years!!!
    • Neural nets are back with a new name!!!!
      - Deep belief networks
      - Huge error reduction when trained with lots of data on GPUs
Overfitting in NNs

Are NNs likely to **overfit**?
- **Yes**, they can represent arbitrary functions!!!

Avoiding overfitting?
- More **training data**
- Fewer hidden nodes / better **topology**
- **Regularization**
- Early stopping

Neural networks can do cool things!
Object recognition

Number detection
Acoustic Modeling for Speech Recognition

Trained in <5 days on cluster of 800 machines
30% reduction in Word Error Rate for English
(“biggest single improvement in 20 years of speech research”)
Launched in 2012 at time of Jellybean release of Android

2012-era Convolutional Model for Object Recognition

Softmax to predict object class

Fully-connected layers

Convolutional layers
(same weights used at all spatial locations in layer)

Convolutional networks developed by
Yann LeCun (NYU)

Basic architecture developed by Krizhevsky, Sutskever & Hinton
(all now at Google).

Won 2012 ImageNet challenge with 16.4% top-5 error rate
2014-era Model for Object Recognition

Module with 6 separate convolutional layers

24 layers deep!

Developed by team of Google Researchers:
Won 2014 ImageNet challenge with 6.66% top-5 error rate

Good Fine-grained Classification

“hibiscus”

“dahlia”

Slides from Jeff Dean at Google
Good Generalization

Both recognized as a "meal"

Sensible Errors

"snake"  "dog"

Slides from Jeff Dean at Google
Works in practice for real users.

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

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

Slides from Jeff Dean at Google
Object detection

![Image of object detection]

Redmon et al. 2015

Neural network summary
What you need to know about neural networks

- Perceptron:
  - Relationship to general neurons
- Multilayer neural nets
  - Representation
  - Derivation of backprop
  - Learning rule
- Overfitting

Course Wrap-Up
CSE 446: Machine Learning
Emily Fox
University of Washington
March 10, 2017
What you have learned this quarter

- Learning is function approximation
- Point estimation
- Regression
- Overfitting
- Bias-Variance tradeoff
- Ridge, LASSO
- Cross validation
- Stochastic gradient descent
- Coordinate descent
- Subgradient
- Logistic regression
- Decision trees
- Boosting
- Instance-based learning
- Perceptron
- SVMs
- Kernel trick
- Dimensionality reduction, PCA
- K-means
- Mixtures of Gaussians
- EM
- Discriminative v. Generative learning
- Unsupervised v. Supervised learning
- Naïve Bayes
- Bayes nets
- Neural networks

BIG PICTURE

Improving the performance at some task through experience!!! 😊
- before you start any learning task, remember the fundamental questions:

<table>
<thead>
<tr>
<th>What is the learning problem?</th>
<th>From what experience?</th>
<th>What model?</th>
</tr>
</thead>
<tbody>
<tr>
<td>What loss function are you optimizing?</td>
<td>With what optimization algorithm?</td>
<td>How will you evaluate it?</td>
</tr>
<tr>
<td>Which learning algorithm?</td>
<td>With what guarantees?</td>
<td></td>
</tr>
</tbody>
</table>

©2017 Emily Fox

CSE 446: Machine Learning
Regression

Example: Predicting house prices

Data -> Regression -> Intelligence

$ = ??

house size

+ house features

price ($)

$600,000 and 1000 sq. ft.

Classification

Example: Sentiment analysis

Data -> Classification -> Intelligence

Score(x) < 0

Score(x) > 0

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

All reviews:

“awesome”

“awful”

Score(x) < 0

Score(x) > 0

positive

negative
Similarity/finding data
Example: Document retrieval

Clustering
Example: Document structuring for retrieval
Embedding
Example: Embedding images to visualize data

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

Images with thousands or millions of pixels

Deep Learning
Example: Visual product recommender
You have done a lot!!!

• And (hopefully) learned a lot!!!
  – Implemented
    • LASSO
    • Logistic regression
    • Perceptron
    • Clustering
    • …
  – Answered hard questions and proved many interesting results
  – Completed (I am sure) an amazing ML project
  – And did excellently on the final!

Thank You for the Hard Work!!!