CS 4803 / 7643: Deep Learning

Topics:
- Regularization
- Neural Networks

Dhruv Batra
Georgia Tech
Administrativia

• PS1/HW1 out
  – Available later today on Canvas
  – Due in 4 weeks
  – Asks about topics coming in the next couple of weeks
  – Please please please please please please start early
  – More details next class
Recap from last time
Parametric Approach: **Linear Classifier**

\[ f(x, W) = Wx + b \]

- **Array of 32x32x3 numbers** (3072 numbers total)
- **Image**
- **W** parameters or weights
- **10 numbers giving class scores**

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

Input image

Stretch pixels into column

56
231
24
2

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Example with an image with 4 pixels, and 3 classes (cat/dog/ship)
\[ W = \begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}, \quad b = \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix} \]

\[ x_i = \begin{bmatrix} 1 \end{bmatrix} \]

New, single \( W \):

\[ W' = \begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 & 1.1 \\ 1.5 & 1.3 & 2.1 & 0.0 & 3.2 \\ 0 & 0.25 & 0.2 & -0.3 & -1.2 \end{bmatrix}, \quad b' = \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \\ 1 \end{bmatrix} \]
Linear Classifier: Three Viewpoints

Algebraic Viewpoint

\[ f(x, W) = Wx \]

Visual Viewpoint

One template per class

Geometric Viewpoint

Hyperplanes cutting up space

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Recall from last time: Linear Classifier

<table>
<thead>
<tr>
<th></th>
<th>airplane</th>
<th>automobile</th>
<th>bird</th>
<th>cat</th>
<th>deer</th>
<th>dog</th>
<th>frog</th>
<th>horse</th>
<th>ship</th>
<th>truck</th>
</tr>
</thead>
<tbody>
<tr>
<td>score</td>
<td>-3.45</td>
<td>-8.87</td>
<td>0.09</td>
<td>2.9</td>
<td>4.48</td>
<td>8.02</td>
<td>3.78</td>
<td>1.06</td>
<td>-0.36</td>
<td>-0.72</td>
</tr>
<tr>
<td>score</td>
<td>-0.51</td>
<td>6.04</td>
<td>5.31</td>
<td>-4.22</td>
<td>-4.19</td>
<td>3.58</td>
<td>4.49</td>
<td>-4.37</td>
<td>-2.09</td>
<td>-2.93</td>
</tr>
<tr>
<td>score</td>
<td>3.42</td>
<td>4.64</td>
<td>2.65</td>
<td>5.1</td>
<td>2.64</td>
<td>5.55</td>
<td>-4.34</td>
<td>-1.5</td>
<td>-4.79</td>
<td>6.14</td>
</tr>
</tbody>
</table>

TODO:

1. Define a **loss function** that quantifies our unhappiness with the scores across the training data.
2. Come up with a way of efficiently finding the parameters that minimize the loss function. (optimization)
Softmax vs. SVM

\[ L_i = - \log \left( \frac{e^{sy_i}}{\sum_j e^{sj}} \right) \]

\[ L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \]
Suppose: 3 training examples, 3 classes. With some $W$ the scores $f(x, W) = Wx$ are:

<table>
<thead>
<tr>
<th>Class</th>
<th>Scores</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>3.2, 1.3, 2.2</td>
</tr>
<tr>
<td>car</td>
<td>5.1, 4.9, 2.5</td>
</tr>
<tr>
<td>frog</td>
<td>-1.7, 2.0, -3.1</td>
</tr>
</tbody>
</table>

Multiclass SVM loss:

Given an example where $x$ is the image and $y$ is the (integer) label, and using the shorthand for the scores vector:

$$ L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases} $$

$$ = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) $$

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Softmax vs. SVM

\[ L_i = - \log \left( \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \right) \]

\[ L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \]
Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{sk}}{\sum_{j} e^{sj}}$$

**Softmax Function**

<table>
<thead>
<tr>
<th>Cat</th>
<th>Frog</th>
<th>Car</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>-1.7</td>
<td>5.1</td>
</tr>
</tbody>
</table>
Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as probabilities

\[ s = f(x_i; W) \]

\[ P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \]

<table>
<thead>
<tr>
<th>Class</th>
<th>Score</th>
<th>Probabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>3.2</td>
<td>24.5</td>
</tr>
<tr>
<td>car</td>
<td>5.1</td>
<td>164.0</td>
</tr>
<tr>
<td>frog</td>
<td>-1.7</td>
<td>0.18</td>
</tr>
</tbody>
</table>

Probabilities must be >= 0

unnormalized probabilities

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as probabilities

\[
s = f(x_i; W)
\]

\[
P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}
\]

Probabilities must be \( \geq 0 \)

Probabilities must sum to 1

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>3.2</td>
<td>24.5</td>
</tr>
<tr>
<td>car</td>
<td>5.1</td>
<td>164.0</td>
</tr>
<tr>
<td>frog</td>
<td>-1.7</td>
<td>0.18</td>
</tr>
</tbody>
</table>

exp

unnormalized probabilities

normalize

probabilities
Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

\[ s = f(x_i; W) \]

\[ P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \]

**Softmax Function**

<table>
<thead>
<tr>
<th>Class</th>
<th>Score</th>
<th>Unnormalized Probabilities</th>
<th>Probabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>3.2</td>
<td>24.5</td>
<td>0.13</td>
</tr>
<tr>
<td>car</td>
<td>5.1</td>
<td>164.0</td>
<td>0.87</td>
</tr>
<tr>
<td>frog</td>
<td>-1.7</td>
<td>0.18</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Probabilities must be >= 0
Probabilities must sum to 1

Unnormalized log-probabilities / logits
Unnormalized probabilities
Probabilities

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

\[ s = f(x_i; W) \]

\[ P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \]

**Softmax Function**

Probabilities must be >= 0

Probabilities must sum to 1

\[ L_i = -\log P(Y = y_i | X = x_i) \]

\[ L_i = -\log(0.13) \]

\[ = 2.04 \]

Cat: 3.2, 24.5, 0.13

Car: 5.1, 164.0, 0.87

Frog: -1.7, 0.18, 0.00

Unnormalized log-probabilities / logits

Unnormalized probabilities

Probabilities

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

\[ s = f(x_i; W) \]

\[ P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \]

Want to interpret raw classifier scores as **probabilities**

Probabilities must be \( \geq 0 \)

Probabilities must sum to 1

\[ L_i = -\log P(Y = y_i | X = x_i) \]

**Unnormalized log-probabilities / logits**

**normalized probabilities**

\[ \exp \]

\[ \text{normalize} \]

**Maximum Likelihood Estimation**

Choose probabilities to maximize the likelihood of the observed data

\[ \log(0.13) \]

\[ = 2.04 \]

---

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Log-Likelihood / KL-Divergence / Cross-Entropy

\[ D = \{(x_i, y_i)\} \]

\[ \text{IID} \sim P_x \]

\[ \text{MLE} = \max \log P(CD1\mid w) \]

\[ \approx \max \sum_i \log P(C_y \mid x_i, w) \]
Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as probabilities

\[ s = f(x_i; W) \]

\[ P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \]

Softmax Function

**Unnormalized log-probabilities / logits**

<table>
<thead>
<tr>
<th></th>
<th>cat</th>
<th>car</th>
<th>frog</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3.2</td>
<td>5.1</td>
<td>-1.7</td>
</tr>
</tbody>
</table>

**Unnormalized probabilities**

<table>
<thead>
<tr>
<th></th>
<th>24.5</th>
<th>164.0</th>
<th>0.18</th>
</tr>
</thead>
</table>

**Probabilities**

<table>
<thead>
<tr>
<th></th>
<th>0.13</th>
<th>0.87</th>
<th>0.00</th>
</tr>
</thead>
</table>

**Correct probs**

<table>
<thead>
<tr>
<th></th>
<th>1.00</th>
<th>0.00</th>
<th>0.00</th>
</tr>
</thead>
</table>

Probabilities must be \(\geq 0\)

Probabilities must sum to 1

\[ L_i = - \log P(Y = y_i | X = x_i) \]

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
**Softmax Classifier** (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

<table>
<thead>
<tr>
<th>cat</th>
<th>3.2</th>
<th>24.5</th>
<th>0.13</th>
<th>1.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>car</td>
<td>5.1</td>
<td>164.0</td>
<td>0.87</td>
<td>0.00</td>
</tr>
<tr>
<td>frog</td>
<td>-1.7</td>
<td>0.18</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Unnormalized log-probabilities / logits

Unnormalized probabilities

Probabilities

Probabilities must be >= 0

Probabilities must sum to 1

Correct probs

$D_{KL}(P||Q) = \sum_y P(y) \log \frac{P(y)}{Q(y)}$

Kullback–Leibler divergence

$Li = -\log P(Y = y_i|X = x_i)$

Compare

Correct probs

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

\[ s = f(x_i; W) \]

\[ P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \]

Softmax Function

<table>
<thead>
<tr>
<th>cat</th>
<th>3.2</th>
<th>24.5</th>
<th>0.13</th>
<th>1.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>car</td>
<td>5.1</td>
<td>164.0</td>
<td>0.87</td>
<td>0.00</td>
</tr>
<tr>
<td>frog</td>
<td>-1.7</td>
<td>0.18</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Unnormalized log-probabilities / logits

Unnormalized probabilities / logits

Probabilities must be >= 0

Probabilities must sum to 1

\[ L_i = -\log P(Y = y_i|X = x_i) \]

Correct probs

Cross Entropy

\[ H(P, Q) = H(p) + D_{KL}(P||Q) \]
Plan for Today

• (Finish) Loss Functions
• Regularization
• Neural Networks
Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Maximize probability of correct class

$$L_i = - \log P(Y = y_i | X = x_i)$$

Putting it all together:

$$L_i = - \log(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}})$$

Q: What is the min/max possible loss $L_i$?

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
**Softmax Classifier** *(Multinomial Logistic Regression)*

Want to interpret raw classifier scores as **probabilities**

\[
s = f(x_i; W) \quad P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}
\]

Maximize probability of correct class

\[
L_i = -\log P(Y = y_i | X = x_i)
\]

Putting it all together:

\[
L_i = -\log \left( \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \right)
\]

**Q:** What is the min/max possible loss \(L_i\)?

**A:** min 0, max infinity

---

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as probabilities

\[ s = f(x_i; W) \]

\[ P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \]

Maximize probability of correct class

\[ L_i = -\log P(Y = y_i | X = x_i) \]

Putting it all together:

\[ L_i = -\log \left( \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \right) \]

Q2: At initialization all \( s \) will be approximately equal; what is the loss?

\[ \log \frac{1}{k} = \log k \]
Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

\[ s = f(x_i; W) \]

\[ P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \]

Maximize probability of correct class

Putting it all together:

\[ L_i = -\log P(Y = y_i | X = x_i) \]

\[ L_i = -\log \left( \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \right) \]

**Q2:** At initialization all \( s \) will be approximately equal; what is the loss?

**A:** \( \log(C) \), eg \( \log(10) \approx 2.3 \)

---

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Softmax vs. SVM

\[ L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right) \]

\[ L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \]
Softmax vs. SVM

Matrix multiply + bias offset

\[ W \]

\[ x_i \]

\[ b \]

\[ y_i \]

\[ 2 \]

Hinge loss (SVM)

\[
\max(0, -2.85 - 0.28 + 1) + \\
\max(0, 0.86 - 0.28 + 1)
\]

\[ = 1.58 \]

cross-entropy loss (Softmax)

\[
\frac{\exp(-2.85)}{\exp(-2.85) + \exp(0.86 + 0.28 + 1)} \]

\[
\frac{\exp(0.86)}{\exp(-2.85) + \exp(0.86 + 0.28 + 1)} \]

\[
\frac{\exp(0.28)}{\exp(-2.85) + \exp(0.86 + 0.28 + 1)} \]

\[ \Rightarrow 0.631 \]

\[ 
- \log(0.353) 
\]

\[ = 0.452 \]

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
**Softmax vs. SVM**

\[ L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right) \quad \text{vs} \quad L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \]

*assume scores:*

- \([10, -2, 3]\)
- \([10, 9, 9]\)
- \([10, -100, -100]\)

and \(y_i = 0\)

Q: Suppose I take a datapoint and I jiggle a bit (changing its score slightly). What happens to the loss in both cases?
Plan for Today

• (Finish) Loss Functions
• Regularization
• Neural Networks
Regularization

Data loss: Model predictions should match training data

\[ L(W) = \frac{1}{N} \sum_{i=1}^{N} L_i(f(x_i, W), y_i) \]
Regularization

$$L(W) = \frac{1}{N} \sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda R(W)$$

**Data loss:** Model predictions should match training data

**Regularization:** Prevent the model from doing too well on training data

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Regularization

\[
L(W) = \frac{1}{N} \sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda R(W)
\]

- **Data loss**: Model predictions should match training data
- **Regularization**: Prevent the model from doing too well on training data

\(\lambda\) = regularization strength (hyperparameter)

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Regularization Intuition in Polynomial Regression

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Polynomial Regression

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Polynomial Regression

\[ y = w_0 + w_1 x + 1 \]

\[ = w_0 + w_1 x + w_2 x^2 \]

\[ = w_0 + \cdots + w_d x^d \quad \text{d-degree} \]

\[ = \begin{bmatrix} w_0 & \cdots & w_d \end{bmatrix} \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^d \end{bmatrix} = \mathbf{w}^T \phi(x) \]
Polynomial Regression
Error Decomposition

Modeling Error

Estimation Error

Optimization Error = 0

Multi-class Logistic Regression

\[(w_0, w_1, \ldots, w_m)\]

\[(w_0, w_1, w_2)\]

Input

\[\text{C-HxWx3}\]

Softmax

Reality

\[\chi \rightarrow \gamma\]
Polynomial Regression

• Demo: [https://arachnoid.com/polysolve/](https://arachnoid.com/polysolve/)

• You are a scientist studying runners.
  - You measure average speeds of the best runners at different ages.

• Data: Age (years), Speed (mph)
  - 10 6
  - 15 9
  - 20 11
  - 25 12
  - 29 13
  - 40 11
  - 50 10
  - 60 9

(C) Dhruv Batra
Regularization

\[ L(W) = \frac{1}{N} \sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda R(W) \]

\( \lambda \) = regularization strength (hyperparameter)

**Data loss:** Model predictions should match training data

**Regularization:** Prevent the model from doing too well on training data

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Regularization

\[ L(W) = \frac{1}{N} \sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda R(W) \]

**Data loss**: Model predictions should match training data

**Regularization**: Prevent the model from doing *too* well on training data

\[ \lambda = \text{regularization strength (hyperparameter)} \]

**Simple examples**

- **L2 regularization**: 
  \[ R(W) = \sum_{k} \sum_{l} W_{k,l}^2 \]

- **L1 regularization**: 
  \[ R(W) = \sum_{k} \sum_{l} |W_{k,l}| \]

- **Elastic net (L1 + L2)**: 
  \[ R(W) = \sum_{k} \sum_{l} \beta W_{k,l}^2 + |W_{k,l}| \]

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Regularization

\[ L(W) = \frac{1}{N} \sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda R(W) \]

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing too well on training data

\( \lambda \) = regularization strength (hyperparameter)

Simple examples
- L2 regularization: \( R(W) = \sum_k \sum_l W_{k,l}^2 \)
- L1 regularization: \( R(W) = \sum_k \sum_l |W_{k,l}| \)
- Elastic net (L1 + L2): \( R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}| \)

More complex:
- Dropout
- Batch normalization
- Stochastic depth, fractional pooling, etc
Regularization

\[ L(W) = \frac{1}{N} \sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda R(W) \]

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing too well on training data

\[ \lambda = \text{regularization strength (hyperparameter)} \]

Why regularize?

- Express preferences over weights
- Make the model simple so it works on test data
- Improve optimization by adding curvature

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
- We have some dataset of \((x, y)\)
- We have a **score function**: 
  \[ s = f(x; W) = Wx \]
- We have a **loss function**:

\[
L_i = - \log\left( \frac{e^{sy_i}}{\sum_j e^{s_j}} \right)
\]

\[
L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)
\]

\[
L = \frac{1}{N} \sum_{i=1}^{N} L_i + R(W)
\]

---

**Recap**

- **Softmax**
- **SVM**
- **Full loss**

---

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
We have some dataset of \((x, y)\)
- We have a **score function**: \(s = f(x; W) = Wx\)
- We have a **loss function**:

\[
L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{sj}}\right)
\]

\[
L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)
\]

\[
L = \frac{1}{N} \sum_{i=1}^{N} L_i + R(W)
\]

**Softmax**

**SVM**

**Full loss**

**Slide Credit:** Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Next: Neural Networks
So far: Linear Classifiers
Hard cases for a linear classifier

**Class 1:**
First and third quadrants

**Class 2:**
Second and fourth quadrants

1 \leq L2 norm \leq 2

Everything else

Three modes

Everything else

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Aside: Image Features

\[ f(x) = Wx \]

Feature Representation

Class scores

Raw Input

I
Cannot separate red and blue points with linear classifier
Image Features: Motivation

Cannot separate red and blue points with linear classifier

\[ f(x, y) = (r(x, y), \theta(x, y)) \]

After applying feature transform, points can be separated by linear classifier
Example: Color Histogram
Example: Histogram of Oriented Gradients (HoG)

Divide image into 8x8 pixel regions
Within each region quantize edge direction into 9 bins

Example: 320x240 image gets divided into 40x30 bins; in each bin there are 9 numbers so feature vector has 30*40*9 = 10,800 numbers

Image features vs Neural Nets

Feature Extraction

10 numbers giving scores for classes

Training

Error Decomposition

Modeling Error

Optimization Error

Estimation Error

Reality

Multi-class Logistic Regression

Softmax

FC HxWx3

input

horse  person

(C) Dhruv Batra
Neural networks: without the brain stuff

(Before) Linear score function:

\[ f = Wx \]

\[ = W^{(3)} (W^{(2)} W^{(1)} x) \]
Neural networks: without the brain stuff

(Before) Linear score function: \( f = Wx \)

(Now) 2-layer Neural Network: \( f = W_2 \max(0, W_1x) \)
Neural networks: without the brain stuff

(Before) Linear score function: $f = Wx$

(Now) 2-layer Neural Network

$\hat{f} = W_2 \max(0, W_1 x)$
Neural networks: without the brain stuff

(Before) Linear score function:

\[ f = Wx \]

(Now) 2-layer Neural Network

\[ f = W_2 \max(0, W_1x) \]
Neural networks: without the brain stuff

(Before) Linear score function:
\[ \sigma(a) = \frac{1}{1 + e^{-a}} \]

(Now) 2-layer Neural Network or 3-layer Neural Network

\[ f = W_2 \max(0, W_1 x) \]

or

\[ f = W_3 \max(0, W_2 \max(0, W_1 x)) \]

Non-linearity

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Multilayer Networks

- Cascaded “neurons”
- The output from one layer is the input to the next
- Each layer has its own sets of weights

Mathematical notation:
$$h^0 = \sum_{i} w_{i} x_{i}$$

Image Credit: Andrej Karpathy, CS231n
Neural networks: Architectures

“2-layer Neural Net”, or “1-hidden-layer Neural Net”

“3-layer Neural Net”, or “2-hidden-layer Neural Net”

’S Fully-connected’ layers
Impulses carried toward cell body

dendrite

Impulses carried away from cell body

presynaptic terminal

axon

cell body

This image by Felipe Perucho is licensed under CC-BY 3.0

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Impulses carried toward cell body

Impulses carried away from cell body

dendrite

cell body

presynaptic terminal

axon

sigmoid activation function

\[
\frac{1}{1 + e^{-x}}
\]
Be very careful with your brain analogies!

**Biological Neurons:**
- Many different types
- Dendrites can perform complex non-linear computations
- Synapses are not a single weight but a complex non-linear dynamical system
- Rate code may not be adequate

[Dendritic Computation. London and Hausser]
Activation functions

**Sigmoid**
\[ \sigma(x) = \frac{1}{1 + e^{-x}} \]

**tanh**
\[ \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \]

**ReLU**
\[ \max(0, x) \]

**Leaky ReLU**
\[ \max(0.1x, x) \]

**Maxout**
\[ \max(w_1^T x + b_1, w_2^T x + b_2) \]

**ELU**
\[ \begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases} \]

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Activation Functions

- sigmoid vs tanh

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

\[ \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \]

\[ f(20(a)) - 1 \]

\[ a = 10^5 \]

(C) Dhruv Batra
Fig. 4. (a) Not recommended: the standard logistic function, $f(x) = 1/(1 + e^{-x})$. (b) Hyperbolic tangent, $f(x) = 1.7159 \ \tanh\left(\frac{2}{3}x\right)$. 
Rectified Linear Units (ReLU)

[Krizhevsky et al., NIPS12]
Demo Time

• https://playground.tensorflow.org