CS 4803 / 7643: Deep Learning

Topics:
- Regularization
- Neural Networks
- Optimization
- Computing Gradients

Zsolt Kira
Georgia Tech
Recap from last time
Parametric Approach: Linear Classifier

Image

Array of $32 \times 32 \times 3$ numbers
(3072 numbers total)

$W$
parameters
or weights

$f(x, W) = Wx + b$

$10 \times 1$
$10 \times 3072$
$3072 \times 1$

10 numbers giving class scores

$W$

$f(x, W)$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$

$10 \times 1$
Error Decomposition

Realization

Input

Softmax

FC HxWx3

Multi-class Logistic Regression

Modeling Error

Optimization Error

Estimation Error

Model class

horse

person
Example with an image with 4 pixels, and 3 classes (cat/dog/ship)

Input image

Stretch pixels into column

<table>
<thead>
<tr>
<th>56</th>
<th>231</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>56</th>
</tr>
</thead>
<tbody>
<tr>
<td>231</td>
</tr>
<tr>
<td>24</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>
Example with an image with 4 pixels, and 3 classes (cat/dog/ship)
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

1. Define a loss function that quantifies our unhappiness with the scores across the training data.

1. Come up with a way of efficiently finding the parameters that minimize the loss function. (optimization)

TODO:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>airplane</td>
<td>-3.45</td>
<td>-0.51</td>
</tr>
<tr>
<td>automobile</td>
<td>-8.87</td>
<td>6.04</td>
</tr>
<tr>
<td>bird</td>
<td>0.09</td>
<td>5.31</td>
</tr>
<tr>
<td>cat</td>
<td>2.9</td>
<td>-4.22</td>
</tr>
<tr>
<td>deer</td>
<td>4.48</td>
<td>-4.19</td>
</tr>
<tr>
<td>dog</td>
<td>8.02</td>
<td>3.58</td>
</tr>
<tr>
<td>frog</td>
<td>3.78</td>
<td>4.49</td>
</tr>
<tr>
<td>horse</td>
<td>1.06</td>
<td>-4.37</td>
</tr>
<tr>
<td>ship</td>
<td>-0.36</td>
<td>-2.09</td>
</tr>
<tr>
<td>truck</td>
<td>-0.72</td>
<td>-2.93</td>
</tr>
</tbody>
</table>
Softmax vs. SVM

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

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

<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>
<tr>
<td></td>
<td>1.3</td>
<td>4.9</td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td>2.2</td>
<td>2.5</td>
<td>-3.1</td>
</tr>
</tbody>
</table>

Multiclass SVM loss:

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

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

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

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

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

“Hinge loss”

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**

$$s = f(x_i; W) \quad P(Y = k | X = x_i) = \frac{e^{sk}}{\sum_j e^{sj}}$$

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

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

Proportions must be \( \geq 0 \)

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

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 >= 0

Probabilities must sum to 1

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

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

<table>
<thead>
<tr>
<th></th>
<th>Unnormalized log-probabilities / logits</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 \( \geq 0 \)

Probabilities must sum to 1

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

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

<table>
<thead>
<tr>
<th>Category</th>
<th>Score</th>
<th>Unnormalized Probabilities</th>
<th>Normalized 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>

Unnormalized log-probabilities / logits

normalization

exp

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>
</tr>
</thead>
<tbody>
<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

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

\[
L_i = -\log(0.13) = 2.04
\]

Maximum Likelihood Estimation  
Choose probabilities to maximize the likelihood of the observed data

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

**Probabilities must be >= 0**

**Probabilities must sum to 1**

**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**

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

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

Probabilities

- **must be >= 0**
- **must sum to 1**

Classify:

- exp to get **unnormalized log-probabilities / logits**
- normalize to get **unnormalized probabilities**
- compare using **Kullback–Leibler divergence**

Correct probs:

- **0.13**
- **0.87**
- **0.00**

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

Unnormalized log-probabilities / logits

<table>
<thead>
<tr>
<th>Class</th>
<th>Score</th>
<th>Unnormalized Probabilities</th>
<th>Probabilities</th>
<th>Correct Probs</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>3.2</td>
<td>exp</td>
<td>0.13</td>
<td>1.00</td>
</tr>
<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>

Probabilities must be >= 0

Probabilities must sum to 1

Cross Entropy

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
matrix multiply + bias offset

\[
\begin{bmatrix}
0.01 & -0.05 & 0.1 & 0.05 \\
0.7 & 0.2 & 0.05 & 0.16 \\
0.0 & -0.45 & -0.2 & 0.03 \\
\end{bmatrix}
\begin{bmatrix}
x_i \\
\end{bmatrix}
+ 
\begin{bmatrix}
-15 \\
22 \\
-44 \\
\end{bmatrix} 
+ 
\begin{bmatrix}
0.0 \\
0.2 \\
-0.3 \\
\end{bmatrix} 
\]

hinge loss (SVM)

\[
\max(0, -2.85 - 0.28 + 1) + 
\max(0, 0.86 - 0.28 + 1) = 1.58
\]

cross-entropy loss (Softmax)

\[
\begin{align*}
\text{exp} & : 0.058 \\
\text{normalize} & : 0.631 \\
\text{to sum to one} & : 0.353 \\
- \log(0.353) & : 0.452
\end{align*}
\]
Plan for Today

• Regularization
• Neural Networks
• Optimization
• Computing Gradients
Regularization

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

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

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

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

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

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

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

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

• Data:
  – 10 6
  – 15 9
  – 20 11
  – 25 12
  – 29 13
  – 40 11
  – 50 10
  – 60 9
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

\[ \lambda \text{ = regularization strength} \]

\[ \text{(hyperparameter)} \]

**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

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

**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): \( R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}| \)

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

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

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

$$\lambda$$ = regularization strength (hyperparameter)
- We have some dataset of $(x,y)$
- We have a **score function**: $s = f(x; W) = Wx$
- We have a **loss function**:

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

**Softmax**

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

**SVM**

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

**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**: 
  \[ s = f(x; W) = Wx \]
  e.g. Softmax

- We have a **loss function**:

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

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

\[
L = \frac{1}{N} \sum_{i=1}^{N} L_i + R(W) \quad \text{Full loss}
\]
Error Decomposition

Reality

Multi-class Logistic Regression

Softmax
FC HxWx3
Input

model class

Modeling Error

Optimization Error = 0

Estimation Error
Next: Neural Networks
Neural networks: without the brain stuff

(Before) Linear score function: \[ f = WX \]
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

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
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
or 3-layer Neural Network

$f = W_2 \max(0, W_1x)$

$f = W_3 \max(0, W_2 \max(0, W_1x))$
Full implementation of training a 2-layer Neural Network needs ~20 lines:

```python
import numpy as np
from numpy.random import randn

N, D_in, H, D_out = 64, 1000, 100, 10
x, y = randn(N, D_in), randn(N, D_out)
w1, w2 = randn(D_in, H), randn(H, D_out)

for t in range(2000):
    h = 1 / (1 + np.exp(-x.dot(w1)))
y_pred = h.dot(w2)
loss = np.square(y_pred - y).sum()
print(t, loss)

grad_y_pred = 2.0 * (y_pred - y)
gard_w2 = h.T.dot(grad_y_pred)
gard_h = grad_y_pred.dot(w2.T)
gard_w1 = x.T.dot(grad_h * h * (1 - h))
w1 -= 1e-4 * grad_w1
w2 -= 1e-4 * grad_w2
```

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

dendrite

Impulses carried away from cell body

axon

presynaptic terminal

cell body

The 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

presynaptic terminal

axon

cell body

sigmoid activation function

\[ \frac{1}{1 + e^{-x}} \]
Impulses carried toward cell body

Impulses carried away from cell body

cell body
dendrite
apxon
presynaptic terminal

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

cell body

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
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**
\[ \tanh(x) \]

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

• sigmoid vs tanh
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)

[Graph showing the ReLU and Logistic functions with a decrease in training error rate over epochs.]

[Krizhevsky et al., NIPS12]
Limitation

• A single “neuron” is still a linear decision boundary

• What to do?

• Idea: Stack a bunch of them together!
Multilayer Networks

- Cascade Neurons together
- The output from one layer is the input to the next
- Each Layer has its own sets of weights
Neural networks: Architectures

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

“Fully-connected” layers

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

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

- https://playground.tensorflow.org
Optimization
Strategy: **Follow the slope**

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Strategy: **Follow the slope**

In 1-dimension, the derivative of a function:

\[
\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h}
\]

In multiple dimensions, the **gradient** is the vector of (partial derivatives) along each dimension.

The slope in any direction is the **dot product** of the direction with the gradient. The direction of steepest descent is the **negative gradient**.

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

```python
# Vanilla Gradient Descent

while True:
    weights_grad = evaluate_gradient(loss_fun, data, weights)
    weights += - step_size * weights_grad  # perform parameter update
```
negative gradient direction