Midterm Review
CS230 Fall 2018
Broadcasting
Calculating Means

How would you calculate the means across the rows of the following matrix? How about the columns?

\[ M = \begin{bmatrix} 13 & 9 & 29 \\ 5 & 11 & 1 \end{bmatrix} \]
Calculating Means

How would you calculate the means across the rows of the following matrix? How about the columns?

\[ M = \begin{bmatrix} 13 & 9 & 29 \\ 5 & 11 & 1 \end{bmatrix} \]

**Rows:** \[ \text{row\_mu} = \text{np.sum}(M, \text{axis}=1) / M.\text{shape}[1] \]
Calculating Means

How would you calculate the means across the rows of the following matrix? How about the columns?

\[ M = \begin{bmatrix} 13 & 9 & 29 \\ 5 & 11 & 1 \end{bmatrix} \]

**Rows:** \( \text{row}_\mu = \frac{\text{np.sum}(M, \text{axis}=1)}{M.\text{shape}[1]} \)

**Cols:** \( \text{col}_\mu = \frac{\text{np.sum}(M, \text{axis}=0)}{M.\text{shape}[0]} \)
Computing Softmax

How would you compute the softmax across the columns of the following matrix?

\[
M = \begin{bmatrix}
20 & 1 \\
3 & 24 \\
31 & 13
\end{bmatrix}
\]
Computing Softmax

How would you compute the softmax across the columns of the following matrix?

\[ M = \begin{bmatrix} 20 & 1 \\ 3 & 24 \\ 31 & 13 \end{bmatrix} \]

\[
\text{exp} = \text{np.exp}(M)
\]
Computing Softmax

How would you compute the softmax across the columns of the following matrix?

\[
M = \begin{bmatrix}
20 & 1 \\
3 & 24 \\
31 & 13
\end{bmatrix}
\]

```python
exp = np.exp(M)
smx = exp / np.sum(exp, axis=0)
```
Computing Distances

How would you compute the closest column in the matrix $X$ to the vector $V$ (in terms of Euclidean distance)?

$$X = \begin{bmatrix} 3 & 15 & 9 \\ 12 & 34 & 20 \\ 22 & 1 & 18 \end{bmatrix}, \quad V = \begin{bmatrix} 6 \\ 11 \\ 20 \end{bmatrix}$$
Computing Distances

How would you compute the closest column in the matrix $X$ to the vector $V$ (in terms of Euclidean distance)?

$$X = \begin{bmatrix} 3 & 15 & 9 \\ 12 & 34 & 20 \\ 22 & 1 & 18 \end{bmatrix}, V = \begin{bmatrix} 6 \\ 11 \\ 20 \end{bmatrix}$$
Computing Distances

How would you compute the closest column in the matrix $X$ to the vector $V$ (in terms of Euclidean distance)?

$$X = \begin{bmatrix} 3 & 15 & 9 \\ 12 & 34 & 20 \\ 22 & 1 & 18 \end{bmatrix}, V = \begin{bmatrix} 6 \\ 11 \\ 20 \end{bmatrix}$$

$$\text{sq\_diff} = \text{np.square}(X-V)$$
Computing Distances

How would you compute the closest column in the matrix $X$ to the vector $V$ (in terms of Euclidean distance)?

$$X = \begin{bmatrix} 3 & 15 & 9 \\ 12 & 34 & 20 \\ 22 & 1 & 18 \end{bmatrix}, V = \begin{bmatrix} 6 \\ 11 \\ 20 \end{bmatrix}$$

```python
sq_diff = np.square(X-V)
dists = np.sqrt(np.sum(sq_diff, axis=0))
```
How would you compute the closest column in the matrix $X$ to the vector $V$ (in terms of Euclidean distance)?

$$X = \begin{bmatrix} 3 & 15 & 9 \\ 12 & 34 & 20 \\ 22 & 1 & 18 \end{bmatrix}, V = \begin{bmatrix} 6 \\ 11 \\ 20 \end{bmatrix}$$

```python
sq_diff = np.square(X-V)
dists = np.sqrt(np.sum(sq_diff, axis=0))
nearest = np.argmin(dists)
```
L1/L2 Regularization
Logistic Regression and Separable Data

What’s the issue with training a logistic regression model on the following data?
Logistic Regression and Separable Data

What’s the issue with training a logistic regression model on the following data?
Logistic Regression and Separable Data

What’s the issue with training a logistic regression model on the following data?

The parameters will tend to plus/minus infinity! So, it will never converge.
Solving the Exploding Weights Issue

What modification of the loss function can you implement so solve this issue? Write out the new loss function.
Solving the Exploding Weights Issue

What modification of the loss function can you implement so solve this issue? Write out the new loss function.

**Add L2 Regularization**

\[
L(y, \hat{y}, w) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}) + \frac{\lambda}{2} \sum_{i=1}^{n} w_i^2
\]

This new loss function will keep the magnitude of the weights from exploding!
Gradient of the New Loss

Compute the gradient of the weight vector with respect to this new loss function.
Gradient of the New Loss

Compute the gradient of the new loss function with respect to the weight vector.

\[
\frac{dL}{dw} = (\hat{y} - y)x + \lambda w
\]
Another Solution...

What is another, similar modification to the loss function that could help with this issue? Compute its gradient?
Another Solution...

What is another, similar modification to the loss function that could help with this issue? Compute its gradient?

Add L1 Regularization:

\[
L(y, \hat{y}, w) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}) + \sum_{i=1}^{n} |w_i|
\]

\[
\frac{dL}{dw} = (\hat{y} - y)x + \text{sign}(w)
\]
Backprop
\[ L = (y_{\text{pred}} - y)^{3/2} \]

where \( y_{\text{pred}} = \tanh(x) \cdot W \)

Assume that \( y_{\text{pred}} \in \mathbb{R}^{1 \times n} \) and \( x \in \mathbb{R}^{1 \times m} \)

Using backpropagation, obtain \( \frac{\partial L}{\partial x} \)
\[ L = (y_{\text{pred}} - y)^{\frac{1}{2}} \]

\[ y_{\text{pred}} = \tanh(x). W \]

\[ \frac{\partial L}{\partial \mathbf{W}} = ? \]

\[ \frac{\partial L}{\partial \mathbf{W}} = \frac{\partial L}{\partial y_{\text{pred}}} \cdot \frac{\partial y_{\text{pred}}}{\partial \mathbf{W}} = \frac{3}{2} \left( y_{\text{pred}} - y \right)^{\frac{1}{2}} \cdot \mathbf{W} \cdot (1 - \tanh^2(x)) \]

\[ y_{\text{pred}} = \tanh(x). \mathbf{W} \]

\[ = \frac{3}{2} \left( y_{\text{pred}} - y \right)^{\frac{1}{2}} \cdot \mathbf{W}^T \odot (1 - \tanh^2(x)) \]
CNN Input/Output Sizes
Basic (no padding, stride 1)
Basic

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Basic
Basic
Basic
Basic
Basic

Input

Filter

Conv Output
Basic

Input

Filter

Conv Output

Shape = n - f + 1
## Padding

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input

Filter
# Padding

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

- **Input**
- **Filter**
- **Conv Output**
Padding

Input

Filter

Conv Output

Shape = n + 2p - f + 1
Valid and Same Convolutions

- **Valid**
  - No padding
  - Output shape -> $n - f + 1$

- **Same**
  - Pad so that input is same as output size
  - Output shape -> $n + 2p - f + 1$
Stride

Input

Filter
Stride
Stride
Stride
Stride
Basic

Input

Filter

Conv Output

Shape = (n - f)/s + 1
With Stride

\( n \times n \) image

\( f \times f \) filter

\( p \) padding

\( s \) stride

Output size \( \rightarrow \frac{(n + 2p - f)}{s} + 1 \)
Maxpool
Forward prop

2 x 2 Pooling layer with stride 2

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>9</td>
</tr>
</tbody>
</table>

Input
### Forward prop

**2 x 2 Pooling layer with stride 2**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>10</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

**Input**

**Output**

Size of output

\[
(n-f)/s + 1
\]
Forward prop

2 x 2 Pooling layer with stride 2

Input

Output
Forward prop

2 x 2 Pooling layer with stride 2

Input

<table>
<thead>
<tr>
<th>1</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>10</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>9</td>
</tr>
</tbody>
</table>

Output

| 10 | 5 |
Forward prop

2 x 2 Pooling layer with stride 2

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>9</td>
</tr>
</tbody>
</table>

**Input**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

**Output**
Forward prop

2 x 2 Pooling layer with stride 2

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>9</td>
</tr>
</tbody>
</table>

Input

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
</tr>
</tbody>
</table>

Output
Forward prop

2 x 2 Pooling layer with stride 2

Input

Output
## Backprop

<table>
<thead>
<tr>
<th>1</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>10</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>9</td>
</tr>
</tbody>
</table>

*Input to maxpool layer*

<table>
<thead>
<tr>
<th>10</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>9</td>
</tr>
</tbody>
</table>

*Output of maxpool layer*

<table>
<thead>
<tr>
<th>-4</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-6</td>
</tr>
</tbody>
</table>

*Gradient w.r.t output*
### Backprop

#### Gradient w.r.t input

<table>
<thead>
<tr>
<th>?</th>
<th>?</th>
<th>?</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

#### Gradient w.r.t output

<table>
<thead>
<tr>
<th>-4</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-6</td>
</tr>
</tbody>
</table>
ReLU

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

\[ ReLU(x) = \begin{cases} 0, & \text{if } x < 0, \\ x, & \text{otherwise.} \end{cases} \]

\[ \frac{d}{dx} ReLU(x) = \begin{cases} 0, & \text{if } x < 0, \\ 1, & \text{otherwise.} \end{cases} \]
Maxpool

\[
\text{Maxpool}(m_{ij}) = \begin{cases} 
0, & \text{if } m_{ij} \neq \max(m) \\
x, & \text{if } m_{ij} = \max(m)
\end{cases}
\]

\[
\frac{d}{dx} \text{Maxpool}(m_{ij}) = \begin{cases} 
0, & \text{if } m_{ij} \neq \max(m) \\
1, & \text{if } m_{ij} = \max(m)
\end{cases}
\]
Backprop

Keep track of where the maximum value is

\begin{align*}
\text{Input} & : & 1 & 3 & 2 & 1 \\
& & 4 & 10 & 5 & 1 \\
& & 1 & 6 & 6 & 5 \\
& & 2 & 4 & 2 & 9 \\
\text{Output} & : & \begin{array}{cc}
10 & 5 \\
6 & 9 \\
\end{array} \\
\text{Mask} & : & \begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
\end{array}
\end{align*}
Backprop

Mask

Gradient w.r.t output

Gradient w.r.t input
Error Analysis
Dog Classifier

Trying to predict dog vs not dog.
Improving performance

Two kinds of errors - misclassification on **muffins** and **fried chicken**

https://medium.freecodecamp.org/chihuahua-or-muffin-my-search-for-the-best-computer-vision-api-cbda4d6b425d

https://barkpost.com/doodle-or-fried-chicken-twitter/
## Error analysis

- Get 100 examples on dev set

<table>
<thead>
<tr>
<th>Image number</th>
<th>Classified as muffin</th>
<th>Classified as chicken</th>
<th>...</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Y</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>-</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5%</td>
<td>50%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Error Analysis
Strategic Data Acquisition
Trigger Word Detection

Positive word → Negative words → Background noise

000000..000001..10000..000
Classification

Everyone's photos

View all 1,749,921
Dropout

Dimensions of $W^{[l]}$

$$z^{[l]} = W^{[l]} a^{[l]} + b^{[l]}$$

$$a^{[l]} = g(z^{[l]})$$

**keep probability ($p$)**

$$a = \begin{bmatrix} 0.3 \\ 0.6 \\ 0.1 \\ 0.8 \end{bmatrix}$$

$$a^{[l]} = \text{np.random.randn}(a^{[l]}, \text{shape}) < p$$

$$a^{[l]}_* = a^{[l]}$$

new $a^{[l]}$ after dropout = $\begin{bmatrix} 4 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

Done?
Note:

Expected value of $a_i^{(j)}$
\[ = p \cdot a_i^{(j)} + (1-p) \cdot 0 \]
\[ = p \cdot a_i^{(j)} \]

At test time \( \Rightarrow \) keep all on, i.e. \( p = 1 \)
\[ \therefore \text{Expected value} = a_i^{(j)} \]

\[ \therefore \text{During training, } \frac{1}{p} \text{ to get same expected value} \]
Batchnorm
\[ Z = \{z^{(1)}, \ldots, z^{(m)}\} \]

\[
\mu_B = \frac{1}{m} \sum_{i=1}^{m} z^{(i)}
\]

\[
\sigma_B^2 = \frac{1}{m} \sum_{i=1}^{m} (z^{(i)} - \mu_B)^2
\]

\[
z_{\text{norm}}^{(i)} = \frac{z^{(i)} - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}
\]

\[
\tilde{z}^{(i)} = \gamma z_{\text{norm}}^{(i)} + \beta
\]