CS6501: Deep Learning for Visual Recognition

Softmax Classifier + SGD
Today’s Class

Intro to Machine Learning

What is Machine Learning?
Supervised Learning: Classification with K-nearest neighbors
Unsupervised Learning: Clustering with K-means clustering

Softmax Classifier

Stochastic Gradient Descent

Regularization
Teaching Assistants

Ziyan Yang
(tw8cb@virginia.edu)
Office Hours: Thursdays 3 to 5pm (Rice 442)

Paola Cascante-Bonilla
(pc9za@virginia.edu)
Hours: Fridays 2 to 4pm (Rice 442)
Also...

• Assignment 2 will be released between today and tomorrow.

• Subscribe and check Piazza regularly, important information about assignments will go there. Please use Piazza.
Machine Learning

• **Machine learning** is the subfield of computer science that gives "computers the ability to learn without being explicitly programmed.”
  - term coined by Arthur Samuel 1959 while at IBM

• The study of algorithms that can learn from data.

• In contrast to previous Artificial Intelligence systems based on Logic, e.g. ”Expert Systems”
Supervised Learning vs Unsupervised Learning

\[ x \rightarrow y \]
Supervised Learning vs Unsupervised Learning

\[ x \rightarrow y \]
Supervised Learning vs Unsupervised Learning

$X \rightarrow Y$

Classification

Clustering
Supervised Learning Examples

Classification

Face Detection

Language Parsing

Structured Prediction

The screen was a sea of red
Supervised Learning Examples

\[
cat = f( )
\]

\[
= f( )
\]

\[
= f( \text{The screen was a sea of red} )
\]
Supervised Learning – k-Nearest Neighbors

- cat
- dog
- bear

$k = 3$

cat, cat, dog
Supervised Learning – k-Nearest Neighbors

k=3

cat, dog, dog

cat

dog

bear, dog, dog

k=3
Supervised Learning – k-Nearest Neighbors

• How do we choose the right K?
• How do we choose the right features?
• How do we choose the right distance metric?
Supervised Learning – k-Nearest Neighbors

• How do we choose the right K?
• How do we choose the right features?
• How do we choose the right distance metric?

**Answer:** Just choose the one combination that works best! **BUT** not on the test data.

Instead split the training data into a ”Training set” and a ”Validation set” (also called ”Development set”)

Training, Validation (Dev), Test Sets
Training, Validation (Dev), Test Sets

- Training Set
- Validation Set
- Testing Set

Used during development
Training, Validation (Dev), Test Sets

Only to be used for evaluating the model at the very end of development and any changes to the model after running it on the test set, could be influenced by what you saw happened on the test set, which would invalidate any future evaluation.
Unsupervised Learning – k-means clustering

k = 3
1. Initially assign all images to a random cluster
Unsupervised Learning – k-means clustering

2. Compute the mean image (in feature space) for each cluster
Unsupervised Learning – k-means clustering

k = 3
3. Reassign images to clusters based on similarity to cluster means
Unsupervised Learning – k-means clustering

$k = 3$

4. Keep repeating this process until convergence
Unsupervised Learning – k-means clustering

4. Keep repeating this process until convergence

k = 3
Unsupervised Learning – k-means clustering

$k = 3$

4. Keep repeating this process until convergence
Unsupervised Learning – k-means clustering

• How do we choose the right K?
• How do we choose the right features?
• How do we choose the right distance metric?
• How sensitive is this method with respect to the random assignment of clusters?

**Answer:** Just choose the one combination that works best! **BUT** not on the test data.

Instead split the training data into a ”Training set” and a ”Validation set” (also called ”Development set”)

## Supervised Learning - Classification

<table>
<thead>
<tr>
<th>Training Data</th>
<th>Test Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Cat" /></td>
<td><img src="image2" alt="Cat" /></td>
</tr>
<tr>
<td><img src="image3" alt="Dog" /></td>
<td><img src="image4" alt="Dog" /></td>
</tr>
<tr>
<td><img src="image5" alt="Cat" /></td>
<td><img src="image6" alt="Cat" /></td>
</tr>
<tr>
<td><img src="image7" alt="Bear" /></td>
<td><img src="image8" alt="Bear" /></td>
</tr>
</tbody>
</table>

### Training Data:
- Cat
- Dog
- Cat
- Bear

### Test Data:
- Cat
- Dog
- Cat
- Bear
Supervised Learning - Classification

Training Data

\[ x_1 = [ \text{cat} ] \quad y_1 = [ \text{cat} ] \]

\[ x_2 = [ \text{dog} ] \quad y_2 = [ \text{dog} ] \]

\[ x_3 = [ \text{cat} ] \quad y_3 = [ \text{cat} ] \]

\[ \ldots \]

\[ x_n = [ \text{bear} ] \quad y_n = [ \text{bear} ] \]
Supervised Learning - Classification

We need to find a function that maps $x$ and $y$ for any of them.

$$\hat{y}_i = f(x_i; \theta)$$

How do we "learn" the parameters of this function?

We choose ones that makes the following quantity small:

$$\sum_{i=1}^{n} \text{Cost}(\hat{y}_i, y_i)$$
Supervised Learning – Linear Softmax

Training Data

inputs

targets / labels / ground truth

\[ x_1 = [x_{11} \ x_{12} \ x_{13} \ x_{14}] \quad y_1 = 1 \]

\[ x_2 = [x_{21} \ x_{22} \ x_{23} \ x_{24}] \quad y_2 = 2 \]

\[ x_3 = [x_{31} \ x_{32} \ x_{33} \ x_{34}] \quad y_3 = 1 \]

\[ \cdots \]

\[ x_n = [x_{n1} \ x_{n2} \ x_{n3} \ x_{n4}] \quad y_n = 3 \]
Supervised Learning – Linear Softmax

<table>
<thead>
<tr>
<th>Training Data</th>
<th>targets / labels / ground truth</th>
<th>predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 = [x_{11} \ x_{12} \ x_{13} \ x_{14}] )</td>
<td>( y_1 = [1 \ 0 \ 0] )</td>
<td>( \hat{y}_1 = [0.85 \ 0.10 \ 0.05] )</td>
</tr>
<tr>
<td>( x_2 = [x_{21} \ x_{22} \ x_{23} \ x_{24}] )</td>
<td>( y_2 = [0 \ 1 \ 0] )</td>
<td>( \hat{y}_2 = [0.20 \ 0.70 \ 0.10] )</td>
</tr>
<tr>
<td>( x_3 = [x_{31} \ x_{32} \ x_{33} \ x_{34}] )</td>
<td>( y_3 = [1 \ 0 \ 0] )</td>
<td>( \hat{y}_3 = [0.40 \ 0.45 \ 0.15] )</td>
</tr>
<tr>
<td>( \cdots )</td>
<td>( \cdots )</td>
<td>( \cdots )</td>
</tr>
<tr>
<td>( x_n = [x_{n1} \ x_{n2} \ x_{n3} \ x_{n4}] )</td>
<td>( y_n = [0 \ 0 \ 1] )</td>
<td>( \hat{y}_n = [0.40 \ 0.25 \ 0.35] )</td>
</tr>
</tbody>
</table>
Supervised Learning – Linear Softmax

\[ x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}] \quad y_i = [1 \ 0 \ 0] \quad \hat{y}_i = [f_c \ f_d \ f_b] \]

\[ g_c = w_{c1}x_{i1} + w_{c2}x_{i2} + w_{c3}x_{i3} + w_{c4}x_{i4} + b_c \]
\[ g_d = w_{d1}x_{i1} + w_{d2}x_{i2} + w_{d3}x_{i3} + w_{d4}x_{i4} + b_d \]
\[ g_b = w_{b1}x_{i1} + w_{b2}x_{i2} + w_{b3}x_{i3} + w_{b4}x_{i4} + b_b \]

\[ f_c = \frac{e^{g_c}}{e^{g_c} + e^{g_d} + e^{g_b}} \]
\[ f_d = \frac{e^{g_d}}{e^{g_c} + e^{g_d} + e^{g_b}} \]
\[ f_b = \frac{e^{g_b}}{e^{g_c} + e^{g_d} + e^{g_b}} \]
How do we find a good w and b?

\[ x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}] \quad y_i = [1 \ 0 \ 0] \quad \hat{y}_i = [f_c(w, b) \ f_d(w, b) \ f_b(w, b)] \]

We need to find w, and b that minimize the following:

\[
L(w, b) = \sum_{i=1}^{n} \sum_{j=1}^{3} -y_{i,j} \log(\hat{y}_{i,j}) = \sum_{i=1}^{n} -\log(\hat{y}_{i,label}) = \sum_{i=1}^{n} -\log f_{i,label}(w, b)
\]

Why?
Gradient Descent (GD)

\[ L(w, b) = \sum_{i=1}^{n} -\log f_{i,\text{label}}(w, b) \]

\[ \lambda = 0.01 \]

Initialize \( w \) and \( b \) randomly

\textbf{for} \ e = 0, \ \text{num\_epochs} \ \textbf{do}

- Compute: \( dL(w, b)/dw \) and \( dL(w, b)/db \)
- Update \( w \): \( w = w - \lambda \frac{dL(w, b)}{dw} \)
- Update \( b \): \( b = b - \lambda \frac{dL(w, b)}{db} \)

Print: \( L(w, b) \) \ // Useful to see if this is becoming smaller or not.

\textbf{end}
Gradient Descent (GD) (idea)

1. Start with a random value of $w$ (e.g. $w = 12$)

2. Compute the gradient (derivative) of $L(w)$ at point $w = 12$. (e.g. $\frac{dL}{dw} = 6$)

3. Recompute $w$ as:

   $$w = w - \lambda \cdot \frac{dL}{dw}$$
2. Compute the gradient (derivative) of $L(w)$ at point $w = 12$. (e.g. $dL/dw = 6$)

3. Recompute $w$ as:

$$w = w - \lambda \cdot (dL / dw)$$
Gradient Descent (GD) (idea)

2. Compute the gradient (derivative) of $L(w)$ at point $w = 12$. (e.g. $dL/dw = 6$)

3. Recompute $w$ as:

$$w = w - \lambda \cdot (dL / dw)$$
Our function $L(w)$

$L(w) = 3 + (1 - w)^2$
Our function $L(w)$

$$L(w) = 3 + (1 - w)^2$$

$$L(W, b) = \sum_{i=1}^{n} -\log f_{i, label}(W, b)$$
Our function $L(w)$

$$L(w) = 3 + (1 - w)^2$$

$$L(w_1, w_2, \ldots, w_{12}) = -\log_{\text{softmax}}(g(w_1, w_2, \ldots, w_{12}, x_1)_{\text{label}_1})$$

$$-\log_{\text{softmax}}(g(w_1, w_2, \ldots, w_{12}, x_2)_{\text{label}_2})$$

$$\ldots$$

$$-\log_{\text{softmax}}(g(w_1, w_2, \ldots, w_{12}, x_n)_{\text{label}_n})$$
Gradient Descent (GD)

\[ L(w, b) = \sum_{i=1}^{n} -\log f_{i,\text{label}}(w, b) \]

\[ \lambda = 0.01 \]

Initialize \( w \) and \( b \) randomly

**for** \( e = 0, \text{num\_epochs} \) **do**

Compute: \( \frac{dL(w, b)}{dw} \) and \( \frac{dL(w, b)}{db} \)

Update \( w \): \( w = w - \lambda \frac{dL(w, b)}{dw} \)

Update \( b \): \( b = b - \lambda \frac{dL(w, b)}{db} \)

Print: \( L(w, b) \)  // Useful to see if this is becoming smaller or not.

**end**
(mini-batch) Stochastic Gradient Descent (SGD)

\[ l(w, b) = \sum_{i \in B} -\log f_{i,\text{label}}(w, b) \]

\[ \lambda = 0.01 \]

Initialize \( w \) and \( b \) randomly

\[ \text{for} \ e = 0, \ \text{num\_epochs} \ \text{do} \]

\[ \text{for} \ b = 0, \ \text{num\_batches} \ \text{do} \]

\[ \text{Compute:} \quad dl(w, b)/dw \quad \text{and} \quad dl(w, b)/db \]

\[ \text{Update} \ w: \quad w = w - \lambda \ dl(w, b)/dw \]

\[ \text{Update} \ b: \quad b = b - \lambda \ dl(w, b)/db \]

\[ \text{Print:} \quad l(w, b) \quad \// \text{Useful to see if this is becoming smaller or not.} \]

\[ \text{end} \]

\[ \text{end} \]
(mini-batch) Stochastic Gradient Descent (SGD)

\[ l(w, b) = \sum_{i \in B} -\log f_{i,\text{label}}(w, b) \]

\( \lambda = 0.01 \)

Initialize w and b randomly

\[ \text{for } e = 0, \text{num\_epochs} \text{ do} \]

\[ \text{for } b = 0, \text{num\_batches} \text{ do} \]

\[ \text{Compute: } \frac{dl(w, b)}{dw} \text{ and } \frac{dl(w, b)}{db} \text{ for } |B| = 1 \]

Update w:
\[ w = w - \lambda \frac{dl(w, b)}{dw} \]

Update b:
\[ b = b - \lambda \frac{dl(w, b)}{db} \]

Print:
\[ l(w, b) \quad // \text{Useful to see if this is becoming smaller or not.} \]

end
end
Computing Analytic Gradients

This is what we have:

\[ \ell(W, b) = -\log(\hat{y}_{label}(W, b)) = -\log\left( \frac{\exp(a_{label}(W, b))}{\sum_{k=1}^{10} \exp(a_k(W, b))} \right) \]
Computing Analytic Gradients

This is what we have:

\[ \mathcal{L}(W, b) = -\log(\hat{y}_{\text{label}}(W, b)) = -\log\left( \frac{\exp(a_{\text{label}}(W, b))}{\sum_{k=1}^{10} \exp(a_k(W, b))} \right) \]

\[ \mathcal{L} = -\log\left( \frac{\exp(a_{\text{label}})}{\sum_{k=1}^{10} \exp(a_k)} \right) \]

Reminder: \[ a_i = (w_{i,1}x_1 + w_{i,2} + w_{i,3} + w_{i,4}) + b_i \]
Computing Analytic Gradients

This is what we have:

\[ \ell = -\log \left( \frac{\exp(a_{label})}{\sum_{k=1}^{10} \exp(a_k)} \right) \]
Computing Analytic Gradients

This is what we have:

\[ \ell = -\log \left( \frac{\exp(a_{\text{label}})}{\sum_{k=1}^{10} \exp(a_k)} \right) \]

This is what we need:

\[ \frac{\partial \ell}{\partial w_{ij}} \quad \text{for each} \quad w_{ij} \]

\[ \frac{\partial \ell}{\partial b_i} \quad \text{for each} \quad b_i \]
Computing Analytic Gradients

This is what we have:

\[ \ell = -\log\left(\frac{\exp(a_{\text{label}})}{\sum_{k=1}^{10} \exp(a_k)}\right) \]

Step 1: Chain Rule of Calculus

\[ \frac{\partial \ell}{\partial w_{ij}} = \frac{\partial \ell}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}} \]

\[ \frac{\partial \ell}{\partial b_i} = \frac{\partial \ell}{\partial a_i} \frac{\partial a_i}{\partial b_i} \]
Computing Analytic Gradients

This is what we have:

\[ \ell = -\log \left( \frac{\exp(a_{\text{label}})}{\sum_{k=1}^{10} \exp(a_k)} \right) \]

Step 1: Chain Rule of Calculus

Let’s do these first

\[ \frac{\partial \ell}{\partial w_{ij}} = \frac{\partial \ell}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}} \]

\[ \frac{\partial \ell}{\partial b_i} = \frac{\partial \ell}{\partial a_i} \frac{\partial a_i}{\partial b_i} \]
Computing Analytic Gradients

\[ \frac{\partial a_i}{\partial w_{i,j}} = \text{value} \quad \frac{\partial a_i}{\partial b_i} = \text{value} \]

\[ a_i = (w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i \]

\[ \frac{\partial a_i}{\partial w_{i,3}} = \frac{\partial}{\partial w_{i,3}} (w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i \]

\[ \frac{\partial a_i}{\partial w_{i,3}} = x_3 \]

\[ \frac{\partial a_i}{\partial w_{i,j}} = x_j \]
Computing Analytic Gradients

\[
\frac{\partial a_i}{\partial w_{i,j}} = x_j
\]

\[
\frac{\partial a_i}{\partial b_i} = 1
\]

\[
a_i = (w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i
\]

\[
\frac{\partial a_i}{\partial b_i} = \frac{\partial}{\partial b_i} (w_{i,1}x_1 + w_{i,2}x_2 + w_{i,3}x_3 + w_{i,4}x_4) + b_i
\]
Computing Analytic Gradients

\[
\frac{\partial a_i}{\partial w_{i,j}} = x_j
\]

\[
\frac{\partial a_i}{\partial b_i} = 1
\]
Computing Analytic Gradients

This is what we have:

\[ \ell = -\log \left( \frac{\exp(a_{label})}{\sum_{k=1}^{10} \exp(a_k)} \right) \]

Step 1: Chain Rule of Calculus

Now let’s do this one (same for both!)

\[ \frac{\partial \ell}{\partial w_{ij}} = \frac{\partial \ell}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}} \quad \text{and} \quad \frac{\partial \ell}{\partial b_i} = \frac{\partial \ell}{\partial a_i} \frac{\partial a_i}{\partial b_i} \]
Computing Analytic Gradients

\[
\frac{\partial \mathcal{L}}{\partial a_i} = \frac{\partial}{\partial a_i} \left[ -\log \left( \frac{\exp(a_{\text{label}})}{\sum_{k=1}^{10} \exp(a_k)} \right) \right]
\]

\[
= \frac{\partial}{\partial a_i} \left[ \log \left( \sum_{k=1}^{10} \exp(a_k) \right) - a_{\text{label}} \right]
\]

In our cat, dog, bear classification example: \( i = \{0, 1, 2\} \)
Computing Analytic Gradients

\[
\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i} \left[ -\log\left( \frac{\exp(a_{label})}{\sum_{k=1}^{10} \exp(a_k)} \right) \right] \\
= \frac{\partial}{\partial a_i} \left[ \log\left( \sum_{k=1}^{10} \exp(a_k) \right) - a_{label} \right]
\]

In our cat, dog, bear classification example: \( i = \{0, 1, 2\} \)

Let’s say: label = 1  

We need: \( \frac{\partial \ell}{\partial a_0} \quad \frac{\partial \ell}{\partial a_1} \quad \frac{\partial \ell}{\partial a_2} \)
Computing Analytic Gradients

\[
\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i} \left[ \log\left( \sum_{k=1}^{10} \exp(a_k) \right) - a_{label} \right]
\]

when \( i \neq label \):

\[
\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i} \left[ \log\left( \sum_{k=1}^{10} \exp(a_k) \right) - a_{label} \right]
\]

\[
\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i} \log\left( \sum_{k=1}^{10} \exp(a_k) \right)
\]

\[
\frac{\partial \ell}{\partial a_i} = \left( \frac{1}{\sum_{k=1}^{10} \exp(a_k)} \right) \left( \frac{\partial}{\partial a_i} \sum_{k=1}^{10} \exp(a_k) \right)
\]

\[
\frac{\partial \ell}{\partial a_i} = \frac{\exp(a_i)}{\sum_{k=1}^{10} \exp(a_k)} = \hat{y}_i
\]
Remember this slide?

$$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}] \quad y_i = [1 \ 0 \ 0] \quad \hat{y}_i = [f_c \ f_d \ f_b]$$

$$g_c = w_{c1}x_{i1} + w_{c2}x_{i2} + w_{c3}x_{i3} + w_{c4}x_{i4} + b_c$$
$$g_d = w_{d1}x_{i1} + w_{d2}x_{i2} + w_{d3}x_{i3} + w_{d4}x_{i4} + b_d$$
$$g_b = w_{b1}x_{i1} + w_{b2}x_{i2} + w_{b3}x_{i3} + w_{b4}x_{i4} + b_b$$

$$f_c = e^{g_c}/(e^{g_c}+e^{g_d}+e^{g_b})$$
$$f_d = e^{g_d}/(e^{g_c}+e^{g_d}+e^{g_b})$$
$$f_b = e^{g_b}/(e^{g_c}+e^{g_d}+e^{g_b})$$
Computing Analytic Gradients

\[
\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i} \left[ \log \left( \sum_{k=1}^{10} \exp(a_k) \right) - a_{label} \right]
\]

when \(i \neq \text{label} \):

\[
\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i} \left[ \log \left( \sum_{k=1}^{10} \exp(a_k) \right) - a_{label} \right]
\]

\[
\frac{\partial \ell}{\partial a_i} = \frac{\partial}{\partial a_i} \log \left( \sum_{k=1}^{10} \exp(a_k) \right)
\]

\[
\frac{\partial \ell}{\partial a_i} = \left( \frac{1}{\sum_{k=1}^{10} \exp(a_k)} \right) \left( \frac{\partial}{\partial a_i} \sum_{k=1}^{10} \exp(a_k) \right)
\]

\[
\frac{\partial \ell}{\partial a_i} = \frac{\exp(a_i)}{\sum_{k=1}^{10} \exp(a_k)} = \hat{y}_i
\]
Computing Analytic Gradients

\[ \ell = \frac{\partial}{\partial a_i} \left[ \log \left( \sum_{k=1}^{10} \exp(a_k) \right) - a_{\text{label}} \right] \]

when \( i = \text{label} \):

\[ \frac{\partial \ell}{\partial a_{\text{label}}} = \frac{\partial}{\partial a_{\text{label}}} \left[ \log \left( \sum_{k=1}^{10} \exp(a_k) \right) - a_{\text{label}} \right] \]

\[ \frac{\partial \ell}{\partial a_{\text{label}}} = \frac{\partial}{\partial a_{\text{label}}} \log(\sum_{k=1}^{10} \exp(a_k)) - 1 \]

\[ \frac{\partial \ell}{\partial a_{\text{label}}} = \left( \frac{1}{\sum_{k=1}^{10} \exp(a_k)} \right) \left( \frac{\partial}{\partial a_{\text{label}}} \sum_{k=1}^{10} \exp(a_k) \right) - 1 \]

\[ \frac{\partial \ell}{\partial a_{\text{label}}} = \frac{\exp(a_{\text{label}})}{\sum_{k=1}^{10} \exp(a_k)} - 1 = \hat{y}_i - 1 \]
Computing Analytic Gradients

\[ \frac{\partial \ell}{\partial a_0} = \hat{y}_0 \]
\[ \frac{\partial \ell}{\partial a_1} = \hat{y}_1 - 1 \]
\[ \frac{\partial \ell}{\partial a_1} = \hat{y}_2 \]

\[ \frac{\partial \ell}{\partial a} = \begin{bmatrix} \frac{\partial \ell}{\partial a_0} \\ \frac{\partial \ell}{\partial a_1} \\ \frac{\partial \ell}{\partial a_2} \end{bmatrix} = \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 - 1 \\ \hat{y}_2 \end{bmatrix} = \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \\ \hat{y}_2 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \hat{y} - y \]

\[ \frac{\partial \ell}{\partial a_i} = \hat{y}_i - y_i \]
Computing Analytic Gradients

\[ \frac{\partial \ell}{\partial w_{ij}} = \frac{\partial \ell}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}} \]

\[ \frac{\partial a_i}{\partial w_{i,j}} = x_j \]

\[ \frac{\partial a_i}{\partial b_i} = 1 \]

\[ \frac{\partial \ell}{\partial a_i} = \hat{y}_i - y_i \]

\[ \frac{\partial \ell}{\partial w_{i,j}} = (\hat{y}_i - y_i)x_j \]

\[ \frac{\partial \ell}{\partial b_i} = (\hat{y}_i - y_i) \]
Supervised Learning – Softmax Classifier

\[
\hat{y}_i = [f_c \ f_d \ f_b]
\]

Extract features
\[
x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}]
\]

Run features through classifier
\[
g_c = w_{c1}x_{i1} + w_{c2}x_{i2} + w_{c3}x_{i3} + w_{c4}x_{i4} + b_c
\]
\[
g_d = w_{d1}x_{i1} + w_{d2}x_{i2} + w_{d3}x_{i3} + w_{d4}x_{i4} + b_d
\]
\[
g_b = w_{b1}x_{i1} + w_{b2}x_{i2} + w_{b3}x_{i3} + w_{b4}x_{i4} + b_b
\]

Get predictions
\[
f_c = \frac{e^{g_c}}{e^{g_c} + e^{g_d} + e^{g_b}}
\]
\[
f_d = \frac{e^{g_d}}{e^{g_c} + e^{g_d} + e^{g_b}}
\]
\[
f_b = \frac{e^{g_b}}{e^{g_c} + e^{g_d} + e^{g_b}}
\]
More ...

• Regularization
• Momentum updates
• Hinge Loss, Least Squares Loss, Logistic Regression Loss
Assignment 2 – Linear Margin-Classifier

<table>
<thead>
<tr>
<th>Training Data</th>
<th>targets / labels / ground truth</th>
<th>predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>inputs</strong></td>
<td><strong>labels / ground truth</strong></td>
<td><strong>predictions</strong></td>
</tr>
<tr>
<td>$x_1 = \begin{bmatrix} x_{11} &amp; x_{12} &amp; x_{13} &amp; x_{14} \end{bmatrix}$</td>
<td>$y_1 = [1 \ 0 \ 0]$</td>
<td>$\hat{y}_1 = [4.3 \ -1.3 \ 1.1]$</td>
</tr>
<tr>
<td>$x_2 = \begin{bmatrix} x_{21} &amp; x_{22} &amp; x_{23} &amp; x_{24} \end{bmatrix}$</td>
<td>$y_2 = [0 \ 1 \ 0]$</td>
<td>$\hat{y}_2 = [0.5 \ 5.6 \ -4.2]$</td>
</tr>
<tr>
<td>$x_3 = \begin{bmatrix} x_{31} &amp; x_{32} &amp; x_{33} &amp; x_{34} \end{bmatrix}$</td>
<td>$y_3 = [1 \ 0 \ 0]$</td>
<td>$\hat{y}_3 = [3.3 \ 3.5 \ 1.1]$</td>
</tr>
<tr>
<td>\hspace{2cm} . &amp; \hspace{2cm} .</td>
<td>\hspace{2cm} .</td>
<td></td>
</tr>
<tr>
<td>$x_n = \begin{bmatrix} x_{n1} &amp; x_{n2} &amp; x_{n3} &amp; x_{n4} \end{bmatrix}$</td>
<td>$y_n = [0 \ 0 \ 1]$</td>
<td>$\hat{y}_n = [1.1 \ -5.3 \ -9.4]$</td>
</tr>
</tbody>
</table>
Supervised Learning – Linear Softmax

\[ x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}] \quad y_i = [1 \ 0 \ 0] \quad \hat{y}_i = [f_c \ f_d \ f_b] \]

\[ f_c = w_{c1}x_{i1} + w_{c2}x_{i2} + w_{c3}x_{i3} + w_{c4}x_{i4} + b_c \]

\[ f_d = w_{d1}x_{i1} + w_{d2}x_{i2} + w_{d3}x_{i3} + w_{d4}x_{i4} + b_d \]

\[ f_b = w_{b1}x_{i1} + w_{b2}x_{i2} + w_{b3}x_{i3} + w_{b4}x_{i4} + b_b \]
How do we find a good $w$ and $b$?

$x_i = [x_{i1} \ x_{i2} \ x_{i3} \ x_{i4}] \quad y_i = [1 \ 0 \ 0] \quad \hat{y}_i = [f_c(w, b) \ f_d(w, b) \ f_b(w, b)]$

We need to find $w$, and $b$ that minimize the following:

$$L(w, b) = \sum_{i=1}^{n} \sum_{j \neq \text{label}} \max(0, \hat{y}_{ij} - \hat{y}_{i,\text{label}} + \Delta)$$

Why?
Regression vs Classification

Regression
• Labels are continuous variables – e.g. distance.
• Losses: Distance-based losses, e.g. sum of distances to true values.
• Evaluation: Mean distances, correlation coefficients, etc.

Classification
• Labels are discrete variables (1 out of K categories)
• Losses: Cross-entropy loss, margin losses, logistic regression (binary cross entropy)
• Evaluation: Classification accuracy, etc.
Linear Regression – 1 output, 1 input

\[
(y, x) = \left( x_1, y_1 \right), \left( x_2, y_2 \right), \left( x_3, y_3 \right), \left( x_4, y_4 \right), \left( x_5, y_5 \right), \left( x_6, y_6 \right), \left( x_7, y_7 \right), \left( x_8, y_8 \right)
\]
Linear Regression – 1 output, 1 input

Model: $\hat{y} = wx + b$
Linear Regression – 1 output, 1 input

Model: $\hat{y} = wx + b$
Linear Regression – 1 output, 1 input

Model: \( \hat{y} = wx + b \)

Loss: \( L(w, b) = \sum_{i=1}^{8} (\hat{y}_i - y_i)^2 \)
Quadratic Regression

Model: $\hat{y} = w_1 x^2 + w_2 x + b$

Loss: $L(w, b) = \sum_{i=1}^{8} (\hat{y}_i - y_i)^2$
n-polynomial Regression

Model: \( \hat{y} = w_n x^n + \cdots + w_1 x + b \)

Loss: \( L(w, b) = \sum_{i=1}^{8} (\hat{y}_i - y_i)^2 \)
Overfitting

\[ f \text{ is linear} \]
\[ \text{Loss}(w) \text{ is high} \]
Underfitting
High Bias

\[ f \text{ is cubic} \]
\[ \text{Loss}(w) \text{ is low} \]

\[ f \text{ is a polynomial of degree 9} \]
\[ \text{Loss}(w) \text{ is zero!} \]
Overfitting
High Variance

Christopher M. Bishop – Pattern Recognition and Machine Learning
Regularization

- Large weights lead to large variance. i.e. model fits to the training data too strongly.

- Solution: Minimize the loss but also try to keep the weight values small by doing the following:

  \[
  \text{minimize} \quad L(w, b) + \sum_i |w_i|^2
  \]
Regularization

- Large weights lead to large variance. i.e. model fits to the training data too strongly.

- Solution: Minimize the loss but also try to keep the weight values small by doing the following:

\[
\text{minimize} \quad L(w, b) + \alpha \sum_i |w_i|^2 \quad \text{Regularizer term}
\]

E.g. L2-regularizer
SGD with Regularization (L-2)

\[ l(w, b) = l(w, b) + \alpha \sum_i |w_i|^2 \]

\[ \lambda = 0.01 \]

Initialize \( w \) and \( b \) randomly

\[ \text{for } e = 0, \text{ num\_epochs do} \]

\[ \text{for } b = 0, \text{ num\_batches do} \]

Compute: \[ dl(w, b)/dw \] and \[ dl(w, b)/db \]

Update \( w \): \[ w = w - \lambda \frac{dl(w, b)}{dw} - \lambda aw \]

Update \( b \): \[ b = b - \lambda \frac{dl(w, b)}{db} - \lambda aw \]

Print: \[ l(w, b) \] // Useful to see if this is becoming smaller or not.
Revisiting Another Problem with SGD

$\lambda = 0.01$

Initialize $w$ and $b$ randomly

for $e = 0, \text{num\_epochs}$ do
  for $b = 0, \text{num\_batches}$ do
    Compute: $dl(w, b)/dw$ and $dl(w, b)/db$
    Update $w$: $w = w - \lambda \frac{dl(w, b)}{dw} - \lambda \alpha w$
    Update $b$: $b = b - \lambda \frac{dl(w, b)}{db} - \lambda \alpha w$
    Print: $l(w, b)$  // Useful to see if this is becoming smaller or not.
  end
end

$l(w, b) = l(w, b) + \alpha \sum_i |w_i|^2$

These are only approximations to the true gradient with respect to $L(w, b)$
Revisiting Another Problem with SGD

\[
l(w, b) = l(w, b) + \alpha \sum |w_i|^2
\]

\[
\lambda = 0.01
\]

Initialize \( w \) and \( b \) randomly

\begin{verbatim}
for e = 0, num_epochs do
  for b = 0, num_batches do
    Compute: \( \frac{dl(w, b)}{dw} \) and \( \frac{dl(w, b)}{db} \)
    Update w: \( w = w - \lambda \frac{dl(w, b)}{dw} - \lambda \alpha w \)
    Update b: \( b = b - \lambda \frac{dl(w, b)}{db} - \lambda \alpha w \)
    Print: \( l(w, b) \)  // Useful to see if this is becoming smaller or not.
  end
end
\end{verbatim}

This could lead to “un-learning” what has been learned in some previous steps of training.
Solution: Momentum Updates

\[ \lambda = 0.01 \]

Initialize \( w \) and \( b \) randomly

For \( e = 0, \text{num\_epochs} \)

For \( b = 0, \text{num\_batches} \)

Compute: \( dl(w, b)/dw \) and \( dl(w, b)/db \)

Update \( w \) :
\[ w = w - \lambda \frac{dl(w, b)}{dw} - \lambda \alpha w \]

Update \( b \) :
\[ b = b - \lambda \frac{dl(w, b)}{db} - \lambda \alpha w \]

Print: \( l(w, b) \) // Useful to see if this is becoming smaller or not.

End

End

\[ l(w, b) = l(w, b) + \alpha \sum_i |w_i|^2 \]
Solution: Momentum Updates

\[ \lambda = 0.01 \quad \tau = 0.9 \]

Initialize w and b randomly

global \( v \)

\[ \text{for } e = 0, \text{ num\_epochs } \text{ do} \]

\[ \text{for } b = 0, \text{ num\_batches } \text{ do} \]

Compute: \( dl(w, b)/dw \)

Compute: \( v = \tau v + dl(w, b)/dw + \alpha w \)

Update w: \( w = w - \lambda v \)

Print: \( l(w, b) \)  // Useful to see if this is becoming smaller or not.

end

end
More on Momentum

We often think of Momentum as a means of dampening oscillations and speeding up the iterations, leading to faster convergence. But it has other interesting behavior. It allows a larger range of step-sizes to be used, and creates its own oscillations. What is going on?

https://distill.pub/2017/momentum/
Image Features: HoG


Image Features: HoG


Scikit-image implementation

We will aggregate gradient magnitude and directions in 8x8 pixel regions.


Scikit-image implementation

Image Features: HoG

Compute a histogram with 9 bins for angles from 0 to 180


Scikit-image implementation

Image Features: HoG


Scikit-image implementation


Normalize histograms with respect to histograms of adjacent neighbors.
Image Features: HoG

Image (or image region) represented by a vector containing all the histograms.

In this case how long is that vector?


Scikit-image implementation

Image Features: HoG

+ Block Normalization

Figure from Zhuolin Jiang, Zhe Lin, Larry S. Davis, ICCV 2009 for human action recognition.
Extract SIFT Feature Descriptors

Compute Histograms of Features
Summary: Image Features

• Largely replaced by Neural networks
• Still useful to study for inspiration in designing neural networks that compute features.

• Many other features proposed
  • LBP: Local Binary Patterns: Useful for recognizing faces.
  • Dense SIFT: SIFT features computed on a grid similar to the HOG features.
  • etc.
Questions?