CS4501: Introduction to Computer Vision

Softmax Classifier

Various slides from previous courses by:

d'A. Forsyth (Berkeley / UIUC), I. Kokkinos (Ecole Centrale / UCL). S. Lazebnik (UNC / UIUC), S. Seitz (MSR / Facebook), J. Hays (Brown / Georgia Tech), A. Berg (Stony Brook / UNC), D. Samaras (Stony Brook) . J. M. Frahm (UNC), V. Ordonez (UVA), Steve Seitz (UW).
Previous Class

- Introduction to Machine Learning
- Ethics in Machine Learning: Choosing Problems in ML
- Supervised vs Unsupervised ML
- K-Nearest Neighbors Classification
- K-Means Clustering
Today’s Class

• Softmax Classifier
  • Inference vs Training
  • Gradient Descent (GD)
  • Stochastic Gradient Descent (SGD)
  • mini-batch Stochastic Gradient Descent (SGD)

• Regularization

• Max-Margin Classifier
Supervised Learning - Classification

Training Data

Cat
Dog
Bear

Test Data

Cat
Dog
Bear
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
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

<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 ) ( \hat{y}_1 = 1 )</td>
<td></td>
</tr>
<tr>
<td>( x_2 = [x_{21} \ x_{22} \ x_{23} \ x_{24}] )</td>
<td>( y_2 = 2 ) ( \hat{y}_2 = 2 )</td>
<td></td>
</tr>
<tr>
<td>( x_3 = [x_{31} \ x_{32} \ x_{33} \ x_{34}] )</td>
<td>( y_3 = 1 ) ( \hat{y}_3 = 2 )</td>
<td></td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots ) ( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( x_n = [x_{n1} \ x_{n2} \ x_{n3} \ x_{n4}] )</td>
<td>( y_n = 3 ) ( \hat{y}_n = 1 )</td>
<td></td>
</tr>
</tbody>
</table>

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:

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

Training Data

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Targets / Labels / Ground Truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 = [x_{11} \ x_{12} \ x_{13} \ x_{14}] )</td>
<td>( y_1 = 1 )</td>
</tr>
<tr>
<td>( x_2 = [x_{21} \ x_{22} \ x_{23} \ x_{24}] )</td>
<td>( y_2 = 2 )</td>
</tr>
<tr>
<td>( x_3 = [x_{31} \ x_{32} \ x_{33} \ x_{34}] )</td>
<td>( y_3 = 1 )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( x_n = [x_{n1} \ x_{n2} \ x_{n3} \ x_{n4}] )</td>
<td>( y_n = 3 )</td>
</tr>
</tbody>
</table>
## Supervised Learning – Linear Softmax

### Training Data

<table>
<thead>
<tr>
<th>Inputs</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>[ \vdots ]</td>
<td>[ \vdots ]</td>
<td>[ \vdots ]</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

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

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

\[
\begin{align*}
f_c &= e^{g_c} / (e^{g_c} + e^{g_d} + e^{g_b}) \\
f_d &= e^{g_d} / (e^{g_c} + e^{g_d} + e^{g_b}) \\
f_b &= e^{g_b} / (e^{g_c} + e^{g_d} + e^{g_b})
\end{align*}
\]
How do we find a good $w$ and $b$?

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

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

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

Why?
Gradient Descent (GD)

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

\[ \lambda = 0.01 \]

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

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

Compute: \( dL(w, b)/dw \) 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.
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:
   \[
   w = w - \text{lambda} \times \left( \frac{dL}{dw} \right)
   \]
2. Compute the gradient (derivative) of \( L(w) \) at point \( w = 12 \). (e.g. \( dL/dw = 6 \))

3. Recompute \( w \) as:

\[
w = w - \lambda \cdot \left( \frac{dL}{dw} \right)
\]
Gradient Descent (GD) (idea)

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

3. Recompute $w$ as:

$$w = w - \lambda \cdot \left( \frac{dL}{dw} \right)$$
Our function $L(w)$

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

\[ L(w) = 3 + (1 - w)^2 \]

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

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

\[ \lambda = 0.01 \]

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

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

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.

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

\[ \lambda = 0.01 \]

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

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

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

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

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

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

\( \lambda = 0.01 \)

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

\begin{align*}
\text{for } e = 0, \text{ num\_epochs } & \text{ do} \\
\text{for } b = 0, \text{ num\_batches } & \text{ do} \\
\quad & \text{Compute: } \frac{dl(w, b)}{dw} \text{ and } \frac{dl(w, b)}{db} \text{ for } |B| = 1 \\
\quad & \text{Update } w: \quad w = w - \lambda \frac{dl(w, b)}{dw} \\
\quad & \text{Update } b: \quad b = b - \lambda \frac{dl(w, b)}{db} \\
\quad & \text{Print: } l(w, b) \quad /\!/ \text{ Useful to see if this is becoming smaller or not.}
\end{align*}

\text{end}
\text{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:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Let’s say: \( \text{label} = 1 \)

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

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

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

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

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

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

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

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

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

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

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

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

\[
\frac{\partial \ell}{\partial a_i} = \hat{y}_i - y_i
\]
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
  \[ f_c = \frac{e^{g_c}}{(e^{g_c} + e^{g_d} + e^{g_b})} \]
  \[ f_d = \frac{e^{g_d}}{(e^{g_c} + e^{g_d} + e^{g_b})} \]
  \[ f_b = \frac{e^{g_b}}{(e^{g_c} + e^{g_d} + e^{g_b})} \]
### Linear Max Margin Classifier

<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>$\hat{y}_1 = [4.3 \ -1.3 \ 1.1]$</td>
</tr>
<tr>
<td>$x_2 = [x_{21} \ x_{22} \ x_{23} \ x_{24}]$</td>
<td>$y_2 = [0 \ 1 \ 0]$</td>
<td>$\hat{y}_2 = [0.5 \ 5.6 \ -4.2]$</td>
</tr>
<tr>
<td>$x_3 = [x_{31} \ x_{32} \ x_{33} \ x_{34}]$</td>
<td>$y_3 = [1 \ 0 \ 0]$</td>
<td>$\hat{y}_3 = [3.3 \ 3.5 \ 1.1]$</td>
</tr>
<tr>
<td>\hspace{1cm}</td>
<td>\hspace{1cm}</td>
<td>\hspace{1cm}</td>
</tr>
<tr>
<td>$x_n = [x_{n1} \ x_{n2} \ x_{n3} \ x_{n4}]$</td>
<td>$y_n = [0 \ 0 \ 1]$</td>
<td>$\hat{y}_n = [1.1 \ -5.3 \ -9.4]$</td>
</tr>
</tbody>
</table>

**Inputs:** $x_i \\
**Targets / Labels / Ground Truth:** $y_i \\
**Predictions:** $\hat{y}_i$
Linear – Max Margin Classifier - Inference

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

\[ f_c = w_{c1}x_{i1} + w_{c2}x_{i2} + w_{c3}x_{i3} + w_{c4}x_{i4} + b_c \]
\[ f_d = w_{d1}x_{i1} + w_{d2}x_{i2} + w_{d3}x_{i3} + w_{d4}x_{i4} + b_d \]
\[ f_b = w_{b1}x_{i1} + w_{b2}x_{i2} + w_{b3}x_{i3} + w_{b4}x_{i4} + b_b \]
Training: 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:

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

Why this might be good compared to softmax?
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

Taken from Christopher Bishop’s Machine Learning and Pattern Recognition Book.
Detecting Overfitting

• Look at the values of the weights in the polynomial

<table>
<thead>
<tr>
<th></th>
<th>$M = 0$</th>
<th>$M = 1$</th>
<th>$M = 6$</th>
<th>$M = 9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_0^*$</td>
<td>0.19</td>
<td>0.82</td>
<td>0.31</td>
<td>0.35</td>
</tr>
<tr>
<td>$w_1^*$</td>
<td>-1.27</td>
<td>7.99</td>
<td>232.37</td>
<td></td>
</tr>
<tr>
<td>$w_2^*$</td>
<td></td>
<td>-25.43</td>
<td>-5321.83</td>
<td></td>
</tr>
<tr>
<td>$w_3^*$</td>
<td></td>
<td>17.37</td>
<td>48568.31</td>
<td></td>
</tr>
<tr>
<td>$w_4^*$</td>
<td></td>
<td></td>
<td>-231639.30</td>
<td></td>
</tr>
<tr>
<td>$w_5^*$</td>
<td></td>
<td></td>
<td>640042.26</td>
<td></td>
</tr>
<tr>
<td>$w_6^*$</td>
<td></td>
<td></td>
<td>-1061800.52</td>
<td></td>
</tr>
<tr>
<td>$w_7^*$</td>
<td></td>
<td></td>
<td>1042400.18</td>
<td></td>
</tr>
<tr>
<td>$w_8^*$</td>
<td></td>
<td></td>
<td>-557682.99</td>
<td></td>
</tr>
<tr>
<td>$w_9^*$</td>
<td></td>
<td></td>
<td>125201.43</td>
<td></td>
</tr>
</tbody>
</table>
Recommended Reading


Print and Read Chapter 1 (at minimum)
More ...

• Regularization
• Momentum updates
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:

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

\[
\text{minimize } L(w, b) + \alpha \sum_{i} |w_i|^2
\]

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

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

\[ \lambda = 0.01 \]

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

Initialize w and b randomly

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

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

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

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

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

\[ \lambda = 0.01 \]

Initialize w and b randomly

**for** e = 0, num_epochs **do**

**for** b = 0, num_batches **do**

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

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

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

\( \lambda = 0.01 \)

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

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

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

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

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

Keep track of previous gradients in an accumulator variable! and use a weighted average with current gradient.
Solution: Momentum Updates

\( \lambda = 0.01 \quad \tau = 0.9 \)

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

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

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

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

Keep track of previous gradients in an accumulator variable! and use a weighted average with current gradient.
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/
Questions?