CS4501: Introduction to Computer Vision
Max-Margin Classifier, Regularization, Generalization, Momentum, Regression, Multi-label Classification / Tagging
Previous Class

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

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

• Generalization
• Regularization / Momentum
• Max-Margin Classifier
• Regression / Tagging
(mini-batch) Stochastic Gradient Descent (SGD)

\[ \lambda = 0.01 \]

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

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

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

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

Update \( w \):
\[
\begin{align*}
w &= w - \lambda \frac{dl(w, b)}{dw}
\end{align*}
\]

Update \( b \):
\[
\begin{align*}
b &= b - \lambda \frac{dl(w, b)}{db}
\end{align*}
\]

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

end

end

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

For Softmax Classifier
Supervised Learning – Softmax Classifier

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

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

\[ f_c = \frac{e^{g_c}}{(e^{g_c} + e^{g_d} + e^{g_b})} \]
\[ f_d = \frac{e^{g_d}}{(e^{g_c} + e^{g_d} + e^{g_b})} \]
\[ f_b = \frac{e^{g_b}}{(e^{g_c} + e^{g_d} + e^{g_b})} \]

\[ \hat{y}_i = [f_c \ f_d \ f_b] \]
## Linear Max Margin Classifier

**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>$\hat{y}_1 = [4.3 \ -1.3 \ 1.1]$</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>...</td>
<td>...</td>
<td>...</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**
- **targets / labels / ground truth**
- **predictions**
Linear – Max Margin Classifier - Inference

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

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

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

Why this might be good compared to softmax?
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
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: \( L(w, b) = \sum_{i=1}^{i=8} (\hat{y}_i - y_i)^2 \)
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

\[ f \text{ is linear} \]
\[ f \text{ is cubic} \]
\[ f \text{ is a polynomial of degree 9} \]

Loss(\(w\)) is high
Loss(\(w\)) is low
Loss(\(w\)) is zero!

Underfitting  
High Bias  
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>-25.43</td>
<td>-5321.83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( w_3^* )</td>
<td>17.37</td>
<td>48568.31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( w_4^* )</td>
<td></td>
<td>-231639.30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( w_5^* )</td>
<td></td>
<td>640042.26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( w_6^* )</td>
<td></td>
<td>-1061800.52</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( w_7^* )</td>
<td></td>
<td>1042400.18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( w_8^* )</td>
<td></td>
<td>-557682.99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( w_9^* )</td>
<td></td>
<td>125201.43</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Recommended Reading

• [http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop%20-Pattern%20Recognition%20And%20Machine%20Learning%20-Springer%202006.pdf](http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop%20-Pattern%20Recognition%20And%20Machine%20Learning%20-Springer%202006.pdf)

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:

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

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

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

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

\textbf{for} b = 0, num\_batches \textbf{do}

\hspace{1cm} Compute: \( \frac{dl(w, b)}{dw} \) and \( \frac{dl(w, b)}{db} \)

\hspace{1cm} Update \( w \): \( w = w - \lambda \frac{dl(w, b)}{dw} - \lambda aw \)

\hspace{1cm} Update \( b \): \( b = b - \lambda \frac{dl(w, b)}{db} - \lambda aw \)

\hspace{1cm} Print: \( l(w, b) \) // Useful to see if this is becoming smaller or not.

end

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

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

**for** $b = 0$, 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

These are only approximations to the true gradient with respect to $L(w, b)$.
\[ \lambda = 0.01 \]

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

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

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

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

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

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

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.

\[ \text{end} \]

\[ \text{end} \]

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

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

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

global $v$

for $e = 0$, num_epochs do

for $b = 0$, 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

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

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