Loss functions for image classification

Slides adapted from Fei-Fei Li, Justin Johnson, Serena Yeung
http://vision.stanford.edu/teaching/cs231n/
Readings

• Image classification:
  – http://cs231n.github.io/classification/

• Linear classification and loss functions:

• Optimization
  – http://cs231n.github.io/optimization-1/
  – http://cs231n.github.io/optimization-2/
Announcements

• Project 4 (Stereo) is out, due Thursday, April 26, 2018, by 11:59pm
  – To be done in groups of two

• Project 3 voting results
Third Place
Second Place
First Place
Sarah Le Cam and Yunie Mao
Recap

- Image classification

- Difficult to hand-code, so we learn from data

- Train on training data, set hyperparameters using validation data, (at the end) test on test data

\[
\begin{array}{ll}
\text{ llama} & 0.93 \\
\text{ car} & 0.01 \\
\end{array}
\]
Recap

• Learning methods
  – k-Nearest Neighbors
  – Linear classification

• Classifier outputs a score function giving a score to each class

• Today: how do we define how good a classifier is based on the training data?
# Linear classification

**Output scores**

<table>
<thead>
<tr>
<th></th>
<th>airplane</th>
<th>automobile</th>
<th>bird</th>
<th>cat</th>
<th>deer</th>
<th>dog</th>
<th>frog</th>
<th>horse</th>
<th>ship</th>
<th>truck</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-3.45</td>
<td>-0.51</td>
<td>6.04</td>
<td>4.64</td>
<td>5.1</td>
<td>5.55</td>
<td>-4.34</td>
<td>-1.5</td>
<td>-4.79</td>
<td>6.14</td>
</tr>
</tbody>
</table>

**TODO:**

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

2. Come up with a way of efficiently finding the parameters that minimize the loss function. *(optimization)*
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>1.3</td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td>5.1</td>
<td>4.9</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>-1.7</td>
<td>2.0</td>
<td>-3.1</td>
</tr>
</tbody>
</table>

A **loss function** tells how good our current classifier is.

Given a dataset of examples

$$\{(x_i, y_i)\}_{i=1}^{N}$$

Where $x_i$ is image and $y_i$ is (integer) label.

Loss over the dataset is a sum of loss over examples:

$$L = \frac{1}{N} \sum_{i} L_i(f(x_i, W), y_i)$$
Loss function, cost/objective function

• Given ground truth labels \((y_i)\), scores \(f(x_i, \mathbf{W})\)
  – how unhappy are we with the scores?

• Loss function or objective/cost function measures unhappiness

• During training, want to find the parameters \(\mathbf{W}\) that minimizes the loss function
Simpler example: binary classification

• Two classes (e.g., “cat” and “not cat”)
  – AKA “positive” and “negative” classes

![Cat](image1.png) ![Not Cat](image2.png)

![Cat](image3.png) ![Not Cat](image4.png)
Linear classifiers

• Find linear function (*hyperplane*) to separate positive and negative examples

\[ x_i \text{ positive: } x_i \cdot w + b \geq 0 \]

\[ x_i \text{ negative: } x_i \cdot w + b < 0 \]

Which hyperplane is best?

We need a **loss function** to decide
What is a good loss function?

• One possibility
  – Number of misclassified examples
    – Problems: discrete, can’t break ties
    – We want the loss to lead to good generalization
Idea: support vector machines (SVMs)

• Find hyperplane that maximizes the margin between the positive and negative examples

Support vector machines

• Find hyperplane that maximizes the margin between the positive and negative examples

\[
\begin{align*}
\text{x}_i \text{ positive}(y_i = 1): & \quad \text{x}_i \cdot \text{w} + b \geq 1 \\
\text{x}_i \text{ negative}(y_i = -1): & \quad \text{x}_i \cdot \text{w} + b \leq -1 \\
\text{For support vectors,} & \quad \text{x}_i \cdot \text{w} + b = \pm 1 \\
\text{Distance between point} & \quad \frac{|\text{x}_i \cdot \text{w} + b|}{\|\text{w}\|} \\
\text{and hyperplane:} & \\
\text{Therefore, the margin is} & \quad 2 / \|\text{w}\| 
\end{align*}
\]
From now on, we will use $W$ to mean both weights and bias.
What if classes aren’t linearly separable?

• Margin doesn’t exist, so we can’t maximize it!
• Alternative: *hinge loss*

\[ L_i = \max (0, 1 - y_i \cdot (x_i \cdot w)) \]

Recall: \( y_i = 1 \) for positive examples
\( y_i = -1 \) for negative examples

• Total loss: sum of hinge losses over training set
Loss function for more than 2 classes

• Given ground truth labels \((y_i)\) and scores \(f(x_i, W)\) – how unhappy are you with the scores?

\[
f(x_i, W) = [13, -7, 11]
\]
\[
y_i = 0
\]
Intuition: generalization of hinge loss
Multi-class SVM loss

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>cat</td>
<td>3.2</td>
<td>1.3</td>
<td>2.2</td>
</tr>
<tr>
<td>car</td>
<td>5.1</td>
<td>4.9</td>
<td>2.5</td>
</tr>
<tr>
<td>frog</td>
<td>-1.7</td>
<td>2.0</td>
<td>-3.1</td>
</tr>
</tbody>
</table>
Multi-class SVM loss

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>1.3</td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td>5.1</td>
<td>4.9</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>-1.7</td>
<td>2.0</td>
<td>-3.1</td>
</tr>
</tbody>
</table>

**Multiclass SVM loss:**

Given an example $(x_i, y_i)$ where $x_i$ is the image and where $y_i$ is the (integer) label, and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

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

$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$
Multi-class SVM loss

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>score</td>
<td>3.2</td>
<td>1.3</td>
<td>2.2</td>
</tr>
<tr>
<td>score</td>
<td>5.1</td>
<td>4.9</td>
<td>2.5</td>
</tr>
<tr>
<td>score</td>
<td>-1.7</td>
<td>2.0</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} 
```

“Hinge loss”
Multi-class SVM loss

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

**Multi-class SVM loss:**

Given an example $(x_i, y_i)$ where $x_i$ is the image and $y_i$ is the (integer) label, and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$
Loss function: interpretation

\[ L_i = \sum_{j \neq y_i} \max(0, f(x_i, W)_j - f(x_i, W)_{y_i}) + \Delta \]

- Loss due to example \( i \)
- Sum over all incorrect labels
- Difference between the correct class score and incorrect class score
Multi-class SVM loss

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

**Multiclass SVM loss:**

Given an example $(x_i, y_i)$ where $x_i$ is the image and $y_i$ is the (integer) label, and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

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

$$= \max(0, 5.1 - 3.2 + 1)$$

$$+ \max(0, -1.7 - 3.2 + 1)$$

$$= \max(0, 2.9) + \max(0, -3.9)$$

$$= 2.9 + 0$$

$$= 2.9$$

**Losses:** 2.9
Multi-class SVM loss

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>Losses:</td>
<td>2.9</td>
<td></td>
<td></td>
</tr>
<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:

Given an example $(x_i, y_i)$ where $x_i$ is the image and $y_i$ is the (integer) label, and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

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

$$= \max(0, 1.3 - 4.9 + 1)$$
$$+ \max(0, 2.0 - 4.9 + 1)$$
$$= \max(0, -2.6) + \max(0, -1.9)$$
$$= 0 + 0$$
$$= 0$$
Multi-class SVM loss

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>Scores</td>
<td>3.2</td>
<td>1.3</td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td>5.1</td>
<td>4.9</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>-1.7</td>
<td>2.0</td>
<td>-3.1</td>
</tr>
</tbody>
</table>

| Losses| 2.9  | 0    | 12.9 |

Multiclass SVM loss:

Given an example $(x_i, y_i)$ where $x_i$ is the image and $y_i$ is the (integer) label, and using the shorthand for the scores vector: $s = f(x_i, W)$, the SVM loss has the form:

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

$$= \max(0, 2.2 - (-3.1) + 1)$$
$$+ \max(0, 2.5 - (-3.1) + 1)$$
$$= \max(0, 6.3) + \max(0, 6.6)$$
$$= 6.3 + 6.6$$
$$= 12.9$$
Multi-class SVM loss

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>Loss</td>
<td>3.2</td>
<td>1.3</td>
<td>2.2</td>
</tr>
<tr>
<td>cat</td>
<td>5.1</td>
<td>4.9</td>
<td>2.5</td>
</tr>
<tr>
<td>car</td>
<td>-1.7</td>
<td>2.0</td>
<td>-3.1</td>
</tr>
<tr>
<td>frog</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Multiclass SVM loss:**

Given an example $(x_i, y_i)$
where $x_i$ is the image and
where $y_i$ is the (integer) label,
and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

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

Loss over full dataset is average:

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i$$

$$L = \frac{(2.9 + 0 + 12.9)}{3} = 5.27$$
Multi-class SVM loss

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></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3.2</td>
<td>1.3</td>
<td>2.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.1</td>
<td>4.9</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-1.7</td>
<td>2.0</td>
<td>-3.1</td>
<td></td>
</tr>
<tr>
<td>Losses:</td>
<td>2.9</td>
<td>0</td>
<td>12.9</td>
<td></td>
</tr>
</tbody>
</table>
**Multi-class SVM loss**

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>Losses:</td>
<td>2.9</td>
<td>0</td>
<td>12.9</td>
</tr>
</tbody>
</table>

**Multiclass SVM loss:**

Given an example $(x_i, y_i)$ where $x_i$ is the image and $y_i$ is the (integer) label, and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

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

Q2: what is the min/max possible loss?
Multi-class SVM loss

Suppose: 3 training examples, 3 classes.
With some $W$ the scores $f(x, W) = Wx$ are:

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

The SVM loss has the form:

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

Q3: At initialization $W$ is small so all $s \approx 0$. What is the loss?
Multiclass SVM Loss: Example code

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

```python
def L_i_vectorized(x, y, W):
    scores = W.dot(x)
    margins = np.maximum(0, scores - scores[y] + 1)
    margins[y] = 0
    loss_i = np.sum(margins)
    return loss_i
```
\[ f(x, W) = Wx \]

\[ L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) \]

E.g. Suppose that we found a \( W \) such that \( L = 0 \). Is this \( W \) unique?
\[ f(x, W) = Wx \]

\[ L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, f(x; W)_j - f(x; W)_{y_i} + 1) \]

E.g. Suppose that we found a \( W \) such that \( L = 0 \). Is this \( W \) unique?

No! \( 2W \) is also has \( L = 0 \)!
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><strong>3.2</strong></td>
<td><strong>1.3</strong></td>
<td><strong>2.2</strong></td>
<td></td>
</tr>
<tr>
<td><strong>5.1</strong></td>
<td><strong>4.9</strong></td>
<td><strong>2.5</strong></td>
<td></td>
</tr>
<tr>
<td><strong>-1.7</strong></td>
<td><strong>2.0</strong></td>
<td><strong>-3.1</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Losses:** **2.9** **0** **-2.2**

**Expression:** $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$

**Before:**

$= \max(0, 1.3 - 4.9 + 1)$
$+ \max(0, 2.0 - 4.9 + 1)$
$= \max(0, -2.6) + \max(0, -1.9)$
$= 0 + 0$
$= 0$

**With $W$ twice as large:**

$= \max(0, 2.6 - 9.8 + 1)$
$+ \max(0, 4.0 - 9.8 + 1)$
$= \max(0, -6.2) + \max(0, -4.8)$
$= 0 + 0$
$= 0$
\[ f(x, W) = Wx \]

\[ L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) \]

E.g. Suppose that we found a \( W \) such that \( L = 0 \). Is this \( W \) unique?

No! \( 2W \) is also has \( L = 0 \! \)!
How do we choose between \( W \) and \( 2W \)?
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
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

$$\lambda = \text{regularization strength (hyperparameter)}$$
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

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

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

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

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

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

**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
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

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

\[ x = [1, 1, 1, 1] \]
\[ w_1 = [1, 0, 0, 0] \]
\[ w_2 = [0.25, 0.25, 0.25, 0.25] \]

\[ w_1^T x = w_2^T x = 1 \]

L2 Regularization

\[ R(W) = \sum_k \sum_l W^2_{k,l} \]
Regularization: Expressing Preferences

\[ x = [1, 1, 1, 1] \]
\[ w_1 = [1, 0, 0, 0] \]
\[ w_2 = [0.25, 0.25, 0.25, 0.25] \]

L2 Regularization

\[ R(W) = \sum_k \sum_l W_{k,l}^2 \]

L2 regularization likes to “spread out” the weights

\[ w_1^T x = w_2^T x = 1 \]
Regularization: Prefer Simpler Models
Regularization: Prefer Simpler Models
Regularization: Prefer Simpler Models

Regularization pushes against fitting the data too well so we don’t fit noise in the data.
Summary

• Have score function and loss function
  – Will generalize the score function

• Find W and b to minimize loss
  – Minimize loss using gradient descent

• Next: CNNs
Summary

1. Score function

\[ f(x_i, W, b) = W x_i + b \]

2. Loss function

\[ L = \frac{1}{N} \sum_i \sum_{j \neq y_i} \left[ \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta) \right] + \lambda R(W) \]
Other loss functions

• Scores are not very intuitive

• Softmax classifier
  – Score function is same
  – Intuitive output: normalized class probabilities
  – Extension of logistic regression to multiple classes
Softmax classifier

\[ f(x_i, W) = Wx_i \]

score function is the same

softmax function

\[
\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}
\]

\[[1, -2, 0] \rightarrow [e^1, e^{-2}, e^0] = [2.71, 0.14, 1] \rightarrow [0.7, 0.04, 0.26]\]

Interpretation: squashes values into range 0 to 1

\[ P(y_i \mid x_i; W) \]
Cross-entropy loss

score function is the same

\[ f(x_i, W) = Wx_i \]

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

\[ L_i = -f_{y_i} + \log \sum_j e^{f_j} \]

\[ P(y_i \mid x_i; W) \]

i.e. we’re minimizing the negative log likelihood.
Aside: Loss function interpretation

• Probability
  – Maximum Likelihood Estimation (MLE)
  – Regularization is Maximum a posteriori (MAP) estimation

• Cross-entropy $H$
  – $p$ is true distribution (1 for the correct class), $q$ is estimated
  – Softmax classifier minimizes cross-entropy
  – Minimizes the KL divergence (Kullback-Leibler) between the distribution: distance between $p$ and $q$

\[ H(p, q) = - \sum_x p(x) \log q(x) \]
Example of the difference between the SVM and Softmax classifiers for one datapoint. In both cases we compute the same score vector $f$ (e.g. by matrix multiplication in this section). The difference is in the interpretation of the scores in $f$: The SVM interprets these as class scores and its loss function encourages the correct class (class 2, in blue) to have a score higher by a margin than the other class scores. The Softmax classifier instead interprets the scores as (unnormalized) log probabilities for each class and then encourages the (normalized) log probability of the correct class to be high (equivalently the negative of it to be low). The final loss for this example is 1.58 for the SVM and 1.04 for the Softmax classifier, but note that these numbers are not comparable; They are only meaningful in relation to loss computed within the same classifier and with the same data.
Summary

• Have score function and loss function
  – Will generalize the score function

• Find W and b to minimize loss
  – SVM vs. Softmax
    • Comparable in performance
    • SVM satisfies margins, softmax optimizes probabilities

\[ L = \frac{1}{N} \sum_i \sum_{j \neq y_i} \left[ \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta) \right] + \lambda \sum_k \sum_l W_{k,l}^2 \]

\[ L = \frac{1}{N} \sum_i -\log \left( \frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \right) + \lambda \sum_k \sum_l W_{k,l}^2 \]
Next: Gradient Descent