Lecture 11: Linear Classifiers
Administrative: HW2

- HW2 due Friday 2/26
Administrative: Well-Being Break

- Wednesday 2/24 is an official Well-Being Day
- No lecture on Thursday 2/25
- Regular office hours and discussion sections this week
Dithering Winners! 4\textsuperscript{th} Place
Dithering Winners! 4th Place
Dithering Winners! 3rd Place
Dithering Winners! 2\textsuperscript{nd} Place
Dithering Winners! 1\textsuperscript{st} Place

if you're taking computer vision now

how did you see the computer in 281
Last Time: Machine Learning

Traditional Programming

```
Input --> Algorithm --> Output
```

Machine Learning

```
Data --> Algorithm --> Model --> Output
```

Human
Last Time: Supervised Learning

1. Collect a dataset of images and labels
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on new images

```python
def train(images, labels):
    # Machine learning!
    return model

def predict(model, test_images):
    # Use model to predict labels
    return test_labels
```

Example training set

- airplane
- automobile
- bird
- cat
- deer
Last Time: Types of ML

Supervised Learning

**Data:** \((x, y)\)
- \(x\) is input / feature
- \(y\) is label / target

**Goal:** Learn a *function* to map \(x \rightarrow y\)

Unsupervised Learning

**Data:** \(x\)
- Just data, no labels!

**Goal:** Learn underlying *structure* in the data
Last Time: Least Squares

“Least squares” = Find the line that minimizes squared error

Data:
\((x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\)
\(x_i, y_i \in \mathbb{R}\)

Model:
y = mx + b
Or: \(x = (x, 1); \ w = (m, b)\)
y = \(w \cdot x\)

Training:
\(w^* = \arg \min_w \sum_{i=1}^{N} (y_i - w \cdot x_i)^2\)
Last Time: Over/Under Fitting, Regularization

L2-Regularized Least Squares

$$\arg\min_w \|y - Xw\|^2 + \lambda \|w\|^2$$

- **Fit training data**
- **Regularization Strength**
- **Penalize complexity**

---

Test data | Training data
---|---

\[\begin{array}{c|c}
\text{Temperature} & \text{Latitude} \\
90 & 10 \\
80 & 20 \\
70 & 30 \\
60 & 40 \\
\end{array}\]

Underfitting

\[\begin{array}{c|c}
\text{Temperature} & \text{Latitude} \\
80 & 10 \\
70 & 20 \\
60 & 30 \\
50 & 40 \\
\end{array}\]

Overfitting

Test data | Training data
---|---

\[\begin{array}{c|c}
\text{Temperature} & \text{Latitude} \\
80 & 10 \\
70 & 20 \\
60 & 30 \\
50 & 40 \\
\end{array}\]

\[\begin{array}{c|c}
\text{Temperature} & \text{Latitude} \\
80 & 10 \\
70 & 20 \\
60 & 30 \\
50 & 40 \\
\end{array}\]

---
Last Time: Choosing Hyperparameters

**Idea #1**: Choose hyperparameters that work best on the data

_**BAD:**_ $\lambda = 0$ always works best on training data

![Diagram showing data split into train and test](image)

**Idea #2**: Split data into _train_ and _test_, choose hyperparameters that work best on test data

_**BAD:**_ No idea how we will perform on new data

![Diagram showing data split into train and test](image)

**Idea #3**: Split data into _train_, _val_, and _test_; choose hyperparameters on _val_ and evaluate on _test_

_**Better!**_

![Diagram showing data split into train, validation, and test](image)
Today: Linear Classifiers
Image Classification: Core Vision Task

**Input:** image

**Output:** Assign image to one of a fixed set of categories

- cat
- bird
- deer
- dog
- truck

This image by Nikita is licensed under CC-BY 2.0
Classification with Least Squares

\[ x_i \in \mathbb{R}^D \text{ is image feature} \]
\[ y_i \in \mathbb{R}^C \text{ is one-hot label} \]
\[ y_{i,c} = 1 \text{ if } x_i \text{ has category } c, \ 0 \text{ otherwise} \]

Training \((x_i, y_i)\):
\[
\arg\min_{W} \sum_{i=1}^{n} \|Wx_i - y_i\|^2
\]

Inference \((x)\):
\[ Wx > t \]

Unprincipled in theory, but often effective in practice
The reverse (regression via discrete bins) is also common

Classification via Memorization

Just **memorize** (as in a Python dictionary)
Consider cat/dog/hippo classification.

If this: cat.
If this: dog.
If this: hippo.
Classification via Memorization

Where does this go wrong?

Rule: if this, then cat

Hmmm. Not quite the same.
Classification via Memorization

(1) Compute distance between feature vectors
(2) find nearest
(3) use label.

Known Images

Labels

Cat

Dog

Test Image

Cat!

\[
D(x_1, x_T) \quad x_T
\]

\[
D(x_N, x_T)
\]

\[
x_1 \quad x_N
\]
Nearest Neighbor

“Algorithm”

Training \((x_i, y_i)\):
Memorize training set

Inference \((x)\):
bestDist, prediction = Inf, None
for i in range(N):
  if dist\((x_i, x)\) < bestDist:
    bestDist = dist\((x_i, x)\)
    prediction = \(y_i\)
Nearest Neighbor

Nearest neighbors in two dimensions

Points are training examples; colors give training labels

Background colors give the category a test point would be assigned

Decision boundaries can be noisy; affected by outliers

How to smooth out decision boundaries? Use more neighbors!

Decision boundary is the boundary between two classification regions

$x_0$

$x_1$
K-Nearest Neighbors

K = 1

K = 3

Instead of copying label from nearest neighbor, take \textbf{majority vote} from K closest points
K-Nearest Neighbors

K = 1

K = 3

Using more neighbors helps smooth out rough decision boundaries
K-Nearest Neighbors

Using more neighbors helps reduce the effect of outliers
K-Nearest Neighbors

When $K > 1$ there can be ties! Need to break them somehow.
K-Nearest Neighbors: Distance Metric

**L1 (Manhattan) Distance**

\[ d(x, y) = \sum_i |x_i - y_i| \]

**L2 (Euclidean) Distance**

\[ d(x, y) = \left( \sum_i (x_i - y_i)^2 \right)^{1/2} \]
K-Nearest Neighbors: Distance Metric

L1 (Manhattan) Distance

\[ d(x, y) = \sum_i |x_i - y_i| \]

L2 (Euclidean) Distance

\[ d(x, y) = \left( \sum_i (x_i - y_i)^2 \right)^{1/2} \]
K-Nearest Neighbors

What distance? What value for K?

Training | Validation | Test

- Use these data points for lookup
- Evaluate on these points for different k, distances
K-Nearest Neighbors

• No learning going on but usually effective
• Same algorithm for every task
• As number of datapoints $\rightarrow \infty$, error rate is guaranteed to be at most $2x$ worse than optimal you could do on data
• Training is fast, but inference is slow. Opposite of what we want!
Linear Classifiers

Example Setup: 3 classes

Model – one weight per class:

\[ w_0^T x \] big if cat
\[ w_1^T x \] big if dog
\[ w_2^T x \] big if hippo

Stack together: \[ W_{3xF} \] where \( x \) is in \( R^F \)
Linear Classifiers

Weight matrix a collection of scoring functions, one per class

\[ W \]

\[ Wx_i \]

Prediction is vector where jth component is “score” for jth class.

Diagram by: Karpathy, Fei-Fei
Linear Classifiers: Geometric Intuition

What does a linear classifier look like in 2D?

Be aware: Intuition from 2D doesn’t always carry over into high-dimensional spaces. See: *On the Surprising Behavior of Distance Metrics in High Dimensional Space*. Charu, Hinneburg, Keim. ICDT 2001

Diagram credit: Karpathy & Fei-Fei
Linear Classifiers: Visual Intuition

CIFAR 10:
32x32x3 Images, 10 Classes

- Turn each image into feature by unrolling all pixels
- Train a linear model to recognize 10 classes
Linear Classifiers: Visual Intuition

Decision rule is $\mathbf{w}^T \mathbf{x}$. If $\mathbf{w}_i$ is big, then big values of $x_i$ are indicative of the class.

Deer or Plane?
Linear Classifiers: Visual Intuition

Decision rule is $\mathbf{w}^\top \mathbf{x}$. If $\mathbf{w}_i$ is big, then big values of $x_i$ are indicative of the class.

Ship or Dog?
Linear Classifiers: Visual Intuition

Decision rule is $\mathbf{w}^T \mathbf{x}$. If $\mathbf{w}_i$ is big, then big values of $x_i$ are indicative of the class.
So Far: Linear Score Function

Model – one weight per class:

\[ w_0^T x \quad \text{big if cat} \]
\[ w_1^T x \quad \text{big if dog} \]
\[ w_2^T x \quad \text{big if hippo} \]

Stack together: \[ W_{3 \times F} \] where \( x \) is in \( \mathbb{R}^F \)

How do we know which \( W \) is best?
Choosing W: Loss Function

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

Low loss = good classifier
High loss = bad classifier

(Also called: **objective function**; **cost function**)

Negative loss function sometimes called **reward function**, **profit function**, **utility function**, **fitness function**, etc.

Given a dataset

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

of images \(x_i\) and labels \(y_i\),

Loss for a single example is:

\[
L_i(f(x_i, W), y_i)
\]

Loss for the dataset is

\[
L = \frac{1}{N} \sum_{i=1}^{N} L_i(f(x_i, W), y_i)
\]
Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

Classifier scores

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

cat  \quad 3.2

car  \quad 5.1

frog  \quad -1.7
Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

Classifier scores

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

Softmax function

\[ p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)} \]

cat \hspace{1cm} 3.2

car \hspace{1cm} 5.1

frog \hspace{1cm} -1.7
Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

Classifier scores

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

Softmax function

\[ p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)} \]

Unnormalized log-probabilities / logits

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>3.2</td>
</tr>
<tr>
<td>car</td>
<td>5.1</td>
</tr>
<tr>
<td>frog</td>
<td>-1.7</td>
</tr>
</tbody>
</table>
Cross-Entropy Loss

Want to interpret raw classifier scores as probabilities

Classifier scores
\[ s = f(x_i, W) \]

Softmax function
\[ p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)} \]

Probabilities must be \( \geq 0 \)

Unnormalized log-probabilities / logits

Unnormalized probabilities
Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

Classifer scores

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

Softmax function

\[ p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)} \]

- **cat**: 3.2
  - **unnormalized log-probabilities / logits**: 3.2
  - **unnormalized probabilities**: 24.5
  - **probabilities**: 0.13

- **car**: 5.1
  - **unnormalized log-probabilities / logits**: 5.1
  - **unnormalized probabilities**: 164
  - **probabilities**: 0.87

- **frog**: -1.7
  - **unnormalized log-probabilities / logits**: -1.7
  - **unnormalized probabilities**: 0.18
  - **probabilities**: 0.00

Probabilities must be >= 0
Probabilities must sum to 1
Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

Classifier scores

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

Softmax function

\[ p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)} \]

Loss

\[ L_i = -\log(p_{yi}) \]

<table>
<thead>
<tr>
<th></th>
<th>Classifier scores ( s = f(x_i, W) )</th>
<th>Softmax function</th>
<th>Loss ( L_i = -\log(p_{yi}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>3.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>car</td>
<td>5.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>frog</td>
<td>-1.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Unnormalized log-probabilities / logits

\[ \exp \]

Unnormalized probabilities

\[ \text{normalize} \]

Probabilities

\[ \sum_j \exp(s_j) \]

Probabilities must be \( \geq 0 \)

Probabilities must sum to 1

\[ L_i = -\log(0.13) = 2.04 \]
Cross-Entropy Loss

Want to interpret raw classifier scores as probabilities

Classifier scores
\[ s = f(x_i, W) \]

Softmax function
\[ p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)} \]

Loss
\[ L_i = -\log(p_{y_i}) \]

<table>
<thead>
<tr>
<th>Class</th>
<th>Classifier scores</th>
<th>Softmax function</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>3.2</td>
<td>[ 24.5 ]</td>
<td>[ 0.13 ]</td>
</tr>
<tr>
<td>car</td>
<td>5.1</td>
<td>[ 164 ]</td>
<td>[ 0.87 ]</td>
</tr>
<tr>
<td>frog</td>
<td>-1.7</td>
<td>[ 0.18 ]</td>
<td>[ 0.00 ]</td>
</tr>
</tbody>
</table>

Maximize probabilities

Maximum Likelihood Estimation
Choose weights to maximize the likelihood of the observed data
(See EECS 445 or EECS 545)
Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

Classifier scores

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

Softmax function

\[ p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)} \]

Loss

\[ L_i = - \log(p_{y_i}) \]

Unnormalized log-probabilities / logits

<table>
<thead>
<tr>
<th>cat</th>
<th>3.2</th>
<th>24.5</th>
<th>0.13</th>
<th>Correct probs</th>
</tr>
</thead>
<tbody>
<tr>
<td>car</td>
<td>5.1</td>
<td>164</td>
<td>0.87</td>
<td>1.00</td>
</tr>
<tr>
<td>frog</td>
<td>-1.7</td>
<td>0.18</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Probabilities must be >= 0

Probabilities must sum to 1

Compare
Want to interpret raw classifier scores as **probabilities**

**Classifier scores**

$$s = f(x_i, W)$$

**Softmax function**

$$p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)}$$

**Loss**

$$L_i = -\log(p_{y_i})$$

---

<table>
<thead>
<tr>
<th></th>
<th>Classifier scores</th>
<th>Softmax function</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>3.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>car</td>
<td>5.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>frog</td>
<td>-1.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Unnormalized log-probabilities / logits**

<table>
<thead>
<tr>
<th></th>
<th>exp</th>
<th>normalize</th>
<th>Kullback-Leibler Divergence</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>24.5</td>
<td>0.13</td>
<td>[D_{KL}(P</td>
</tr>
<tr>
<td>car</td>
<td>164</td>
<td>0.87</td>
<td></td>
</tr>
<tr>
<td>frog</td>
<td>0.18</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

**Probabilities must be >= 0**

**Probabilities must sum to 1**

**Compare**

**Correct probs**

1.00

0.00
Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

Classifier scores

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

Softmax function

\[ p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)} \]

Loss

\[ L_i = -\log(p_{y_i}) \]

### Examples

<table>
<thead>
<tr>
<th>cat</th>
<th>3.2</th>
<th>24.5</th>
<th>0.13</th>
<th>1.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>car</td>
<td>5.1</td>
<td>164</td>
<td>0.87</td>
<td>0.00</td>
</tr>
<tr>
<td>frog</td>
<td>-1.7</td>
<td>0.18</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Unnormalized log-probabilities / logits

Unnormalized probabilities

Probabilities

Probabilities must be \( \geq 0 \)

Probabilities must sum to 1

Correct probs

Cross-Entropy:

\[ H(P, Q) = H(P) + D_{KL}(P \parallel Q) \]

\[ D_{KL}(P \parallel Q) = \sum_i p_i \log \left( \frac{p_i}{q_i} \right) \]
Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

Classifier scores
\[ s = f(x_i, W) \]

Softmax function
\[ p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)} \]

Loss
\[ L_i = -\log(p_{y_i}) \]

Putting it all together:
\[ L_i = -\log\left(\frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}\right) \]
Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

Classifier scores

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

Softmax function

\[ p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)} \]

Loss

\[ L_i = - \log(p_{y_i}) \]

Putting it all together:

\[ L_i = - \log\left(\frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}\right) \]

**Q:** What is the min / max possible loss \( L_i \)?

<table>
<thead>
<tr>
<th>cat</th>
<th>3.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>car</td>
<td>5.1</td>
</tr>
<tr>
<td>frog</td>
<td>-1.7</td>
</tr>
</tbody>
</table>
Cross-Entropy Loss

Want to interpret raw classifier scores as **probabilities**

Classifier scores
\[ s = f(x_i, W) \]

Softmax function
\[ p_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)} \]

Loss
\[ L_i = -\log(p_{y_i}) \]

Putting it all together:
\[ L_i = -\log \left( \frac{\exp(s_{y_i})}{\sum_j \exp(s_j)} \right) \]

**Q:** If all scores are small random values, what is the loss?

<table>
<thead>
<tr>
<th>Class</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>3.2</td>
</tr>
<tr>
<td>car</td>
<td>5.1</td>
</tr>
<tr>
<td>frog</td>
<td>-1.7</td>
</tr>
</tbody>
</table>
Multiclass SVM Loss

"The score of the correct class should be higher than all the other scores"

Given an example $(x_i, y_i)$ ($x_i$ is image, $y_i$ is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

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

Given an example \((x_i, y_i)\) (\(x_i\) is image, \(y_i\) is label)

Let \(s = f(x_i, W)\) be scores

Then the SVM loss has the form:

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

Given an example \((x_i, y_i)\) (\(x_i\) is image, \(y_i\) is label)

Let \( s = f(x_i, W) \) be scores

Then the SVM loss has the form:

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

<table>
<thead>
<tr>
<th></th>
<th>cat</th>
<th>car</th>
<th>frog</th>
</tr>
</thead>
<tbody>
<tr>
<td>label</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>score</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 \((x_i, y_i)\)
\((x_i\text{ is image}, y_i\text{ is label})\)

Let \(s = f(x_i, W)\) be scores

Then the SVM loss has the form:

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

\[
= \max(0, 5.1 - 3.2 + 1) + \max(0, -1.7 - 3.2 + 1)
\]

\[
= \max(0, 2.9) + \max(0, -3.9)
\]

\[
= 2.9 + 0
\]

\[
= 2.9
\]
## Multiclass SVM Loss

Given an example \((x_i, y_i)\) 
\((x_i\) is image, \(y_i\) is label)

Let \(s = f(x_i, W)\) be scores

Then the SVM loss has the form:

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

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

<table>
<thead>
<tr>
<th></th>
<th>cat</th>
<th>car</th>
<th>frog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss</td>
<td>2.9</td>
<td>0</td>
<td>0</td>
</tr>
<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>
Multiclass SVM Loss

Given an example \((x_i, y_i)\) (\(x_i\) is image, \(y_i\) is label)

Let \(s = f(x_i, W)\) be scores

Then the SVM loss has the form:

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

\[
= \max(0, 2.2 - (-3.1) + 1) \\
+ \max(0, 2.5 - (-3.1) + 1) \\
= \max(0, 6.3) + \max(0, 6.6) \\
= 6.3 + 6.6 \\
= 12.9
\]
Multiclass SVM Loss

Given an example \((x_i, y_i)\) (\(x_i\) is image, \(y_i\) is label)

Let \(s = f(x_i, W)\) be scores

Then the SVM loss has the form:

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

Loss over the dataset is:

\[
L = \frac{(2.9 + 0.0 + 12.9)}{3} = 5.27
\]
Multiclass SVM Loss

Given an example \((x_i, y_i)\) (\(x_i\) is image, \(y_i\) is label)

Let \(s = f(x_i, W)\) be scores

Then the SVM loss has the form:

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

Q: What happens to the loss if the scores for the car image change a bit?

<table>
<thead>
<tr>
<th></th>
<th>cat</th>
<th>car</th>
<th>frog</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scores</td>
<td>3.2</td>
<td>5.1</td>
<td>-1.7</td>
<td>2.9</td>
</tr>
<tr>
<td>Loss</td>
<td>2.2</td>
<td>2.5</td>
<td>-3.1</td>
<td>12.9</td>
</tr>
</tbody>
</table>
Multiclass SVM Loss

Given an example \((x_i, y_i)\)
\((x_i \text{ is image, } y_i \text{ is label})\)

Let \(s = f(x_i, W)\) be scores

Then the SVM loss has the form:
\[
L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)
\]

**Q:** What are the min and max possible loss?

<table>
<thead>
<tr>
<th></th>
<th>Loss</th>
<th>cat</th>
<th>car</th>
<th>frog</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>2.9</td>
<td>3.2</td>
<td>1.3</td>
<td>-1.7</td>
</tr>
<tr>
<td>car</td>
<td>5.1</td>
<td>4.9</td>
<td>2.0</td>
<td>-2.9</td>
</tr>
<tr>
<td>frog</td>
<td>-1.7</td>
<td>2.0</td>
<td>2.5</td>
<td>-3.1</td>
</tr>
<tr>
<td></td>
<td>2.9</td>
<td>0</td>
<td>12.9</td>
<td></td>
</tr>
</tbody>
</table>
Multiclass SVM Loss

Given an example $(x_i, y_i)$ ($x_i$ is image, $y_i$ is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

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

Q: If all scores were random, what loss would we expect?
Multiclass SVM Loss

Given an example \((x_i, y_i)\) 
\((x_i \text{ is image, } y_i \text{ is label})\)

Let \(s = f(x_i, W)\) be scores

Then the SVM loss has the form:
\[
L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)
\]

Q: What would happen if sum were over all classes? 
(including \(j = y_i\))

<table>
<thead>
<tr>
<th></th>
<th>cat</th>
<th>car</th>
<th>frog</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>1.3</td>
<td>2.2</td>
<td>2.9</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>4.9</td>
<td>2.5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>-1.7</td>
<td>2.0</td>
<td>-3.1</td>
<td>12.9</td>
<td></td>
</tr>
</tbody>
</table>
Multiclass SVM Loss

Given an example \((x_i, y_i)\) (\(x_i\) is image, \(y_i\) is label)

Let \(s = f(x_i, W)\) be scores

Then the SVM loss has the form:

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

Q: What if the loss used mean instead of sum?

<table>
<thead>
<tr>
<th></th>
<th>cat</th>
<th>car</th>
<th>frog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss</td>
<td>2.9</td>
<td>0</td>
<td>12.9</td>
</tr>
<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>
Multiclass SVM Loss

Given an example \((x_i, y_i)\) (\(x_i\) is image, \(y_i\) is label)

Let \(s = f(x_i, W)\) be scores

Then the SVM loss has the form:

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

**Q:** What if we used this loss instead?

\[
L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)^2
\]
Cross-Entropy vs SVM Loss

$$L_i = -\log \frac{\exp(s_{y_i})}{\sum_j \exp(s_j)}$$

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

Assume scores:

[10, -2, 3]
[10, 9, 9]
[10, -100, -100]

and $y_i = 0$

**Q:** What is cross-entropy loss? What is SVM loss?

**A:** Cross-entropy loss > 0
SVM loss = 0
Cross-Entropy vs SVM Loss

\[ L_i = -\log \frac{\exp(s_{y_i})}{\sum_j \exp(s_j)} \]

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

Assume scores:

[10, -2, 3]
[10, 9, 9]
[10, -100, -100]
and \( y_i = 0 \)

Q: What happens to each loss if I slightly change the scores of the last datapoint?

A: Cross-entropy loss will change; SVM loss will stay the same
Cross-Entropy vs SVM Loss

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

Assume scores:

[10, -2, 3]
[10, 9, 9]
[10, -100, -100]
and \( y_i = 0 \)

Q: What happens to each loss if I double the score of the correct class from 10 to 20?

A: Cross-entropy loss will decrease, SVM loss still 0
Recap

- **Image Classification** is a core computer vision task
- **K-Nearest Neighbors** is classification via memorization
- **Linear classifiers** learn one template per category to match with the input
- A **loss function** specifies your preference over different settings of weights
- **Cross-Entropy loss** maximizes probability of correct class
- **SVM Loss** wants correct score larger than other scores
Next Time: How to choose W? Optimization!