Image Classification
Announcements

• Project 4 to be released shortly

• Vote for Project 3 artifacts!
  – Deadline: midnight tonight
Today

- Image classification pipeline
- Training, validation, testing
- Nearest neighbor classification
- Linear classification
- Score function and loss function

- Building up to CNNs for learning
  – Next 2-4 lectures on deep learning
Image Classification: A core task in Computer Vision

- Assume given set of discrete labels, e.g.
  \{cat, dog, cow, apple, tomato, truck, ... \}

\[
\begin{align*}
f(\text{apple}) &= \text{“apple”} \\
f(\text{tomato}) &= \text{“tomato”} \\
f(\text{cow}) &= \text{“cow”}
\end{align*}
\]
Image classification demo

https://cloud.google.com/vision/

See also:
https://aws.amazon.com/rekognition/
https://www.clarifai.com/
https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/
Image Classification

(assume given set of discrete labels) 
{dog, cat, truck, plane, ...}

→ cat
Image Classification: Problem
Recall from last time: Challenges of recognition

Viewpoint

Illumination

Deformation

Occlusion

Clutter

Intraclass Variation
An image classifier

```python
def classify_image(image):
    # Some magic here?
    return class_label
```

Unlike e.g. sorting a list of numbers,

no obvious way to hard-code the algorithm for recognizing a cat, or other classes.
Data-driven approach

• Collect a database of images with labels
• Use ML to train an image classifier
• Evaluate the classifier on test images

Example training set
Data-driven approach

• Collect a database of images with labels
• Use ML to train an image classifier
• Evaluate the classifier on test images

```python
def train(train_images, train_labels):
    # build a model of images -> labels

def predict(image):
    # evaluate the model on the image
    return class_label
```
Classifiers

- Nearest Neighbor
- kNN ("k-Nearest Neighbors")
- Linear Classifier
- SVM (Support Vector Machine)
- ...

First: Nearest Neighbor (NN) Classifier

• Train
  – Remember all training images and their labels

• Predict
  – Find the closest (most similar) training image
  – Predict its label as the true label
Example dataset: CIFAR-10
10 labels
50,000 training images, each image is tiny: 32x32
10,000 test images.

airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
CIFAR-10 and NN results

Example dataset: CIFAR-10
10 labels
50,000 training images
10,000 test images.

For every test image (first column), examples of nearest neighbors in rows.
k-nearest neighbor

- Find the k closest points from training data
- Take **majority vote** from K closest points
What does this look like?
What does this look like?
How to find the most similar training image? What is the distance metric?

**L1 distance:**

\[
d_1(I_1, I_2) = \sum_p |I^p_1 - I^p_2|
\]

Where \( I_1 \) denotes image 1, and \( p \) denotes each pixel.

<table>
<thead>
<tr>
<th>test image</th>
<th>training image</th>
<th>pixel-wise absolute value differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>56 32 10 18</td>
<td>10 20 24 17</td>
<td>46 12 14 1</td>
</tr>
<tr>
<td>90 23 128 133</td>
<td>8 10 89 100</td>
<td>82 13 39 33</td>
</tr>
<tr>
<td>24 26 178 200</td>
<td>12 16 178 170</td>
<td>12 10 0 30</td>
</tr>
<tr>
<td>2 0 255 220</td>
<td>4 32 233 112</td>
<td>2 32 22 108</td>
</tr>
</tbody>
</table>
Choice of distance metric

- Hyperparameter

L1 (Manhattan) distance

\[ d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p| \]

L2 (Euclidean) distance

\[ d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2} \]

- Two most commonly used special cases of p-norm

\[ \|x\|_p = (|x_1|^p + \cdots + |x_n|^p)^{\frac{1}{p}} \quad p \geq 1, x \in \mathbb{R}^n \]
K-Nearest Neighbors: Distance Metric

L1 (Manhattan) distance

\[ d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p| \]

L2 (Euclidean) distance

\[ d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2} \]
K-Nearest Neighbors: Distance Metric

L1 (Manhattan) distance

\[ d_1(I_1, I_2) = \sum_p |I_{1p} - I_{2p}| \]

L2 (Euclidean) distance

\[ d_2(I_1, I_2) = \sqrt{\sum_p (I_{1p} - I_{2p})^2} \]

Visualization: L2 distance
Hyperparameters

• What is the best distance to use?
• What is the best value of k to use?

• These are hyperparameters: choices about the algorithm that we set rather than learn

• How do we set them?
  – One option: try them all and see what works best
Setting Hyperparameters

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

Your Dataset
Setting Hyperparameters

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

**BAD:** $K = 1$ always works perfectly on training data

Your Dataset
Setting Hyperparameters

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

BAD: $K = 1$ always works perfectly on training data

---

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

---

Your Dataset

train    test
Setting Hyperparameters

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

BAD: $K = 1$ always works perfectly on training data

---

**Your Dataset**

---

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

BAD: No idea how algorithm will perform on new data

---

| train | test |
Setting Hyperparameters

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

BAD: K = 1 always works perfectly on training data

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

BAD: No idea how algorithm will perform on new data

Idea #3: Split data into train, val, and test; choose hyperparameters on val and evaluate on test

Better!
Setting Hyperparameters

<table>
<thead>
<tr>
<th>fold 1</th>
<th>fold 2</th>
<th>fold 3</th>
<th>fold 4</th>
<th>fold 5</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>fold 1</td>
<td>fold 2</td>
<td>fold 3</td>
<td>fold 4</td>
<td>fold 5</td>
<td>test</td>
</tr>
<tr>
<td>fold 1</td>
<td>fold 2</td>
<td>fold 3</td>
<td>fold 4</td>
<td>fold 5</td>
<td>test</td>
</tr>
</tbody>
</table>

**Idea #4: Cross-Validation**: Split data into **folds**, try each fold as validation and average the results.

Useful for small datasets, but not used too frequently in deep learning.
Example of 5-fold cross-validation for the value of $k$.

Each point: single outcome.

The line goes through the mean, bars indicated standard deviation

(Seems that $k \sim 7$ works best for this data)
Recap: How to pick hyperparameters?

• Methodology
  – Train and test
  – Train, validate, test

• Train for original model
• Validate to find hyperparameters
• Test to understand generalizability
kNN -- Complexity and Storage

• N training images, M test images

• Training: $O(1)$
• Testing: $O(MN)$

• Hmm...
  – Normally need the opposite
  – Slow training (ok), fast testing (necessary)
k-Nearest Neighbor on images never used.

- terrible performance at test time
- distance metrics on level of whole images can be very unintuitive

(all 3 images have same L2 distance to the one on the left)
k-Nearest Neighbors: Summary

• In **image classification** we start with a **training set** of images and labels, and must predict labels on the **test set**

• The **K-Nearest Neighbors** classifier predicts labels based on nearest training examples

• Distance metric and K are **hyperparameters**

• Choose hyperparameters using the **validation set**; only run on the test set once at the very end!
Linear classifiers
Score function → class scores
Score function: $f$

Parametric approach

$[32\times32\times3]$ array of numbers 0...1 (3072 numbers total)

$\text{image} \rightarrow f(x,W) \rightarrow \text{10 numbers, indicating class scores}$
Parametric approach: Linear classifier

\[ f(x, W) = Wx \]

[32x32x3] array of numbers 0...1

parameters, or “weights”

10 numbers, indicating class scores
Parametric approach: Linear classifier

\[ f(x, W) = Wx + (b) \]

- \([32 \times 32 \times 3]\) array of numbers 0...1
- Parameters, or “weights”
Linear Classifier

define a score function

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

data (image)

class scores

“weights”

“bias vector”

“parameters”
Example with an image with 4 pixels, and 3 classes (cat/dog/ship)
Interpretation: Template matching

$$f(x_i, W, b) = W x_i + b$$
Geometric Interpretation

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

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

\[ x_i \text{ positive: } \mathbf{x}_i \cdot \mathbf{w} + b \geq 0 \]
\[ x_i \text{ negative: } \mathbf{x}_i \cdot \mathbf{w} + b < 0 \]

Which hyperplane is best? We will come back to this later
Hard cases for a linear classifier

Class 1:
First and third quadrants

Class 1:
1 <= L2 norm <= 2

Class 1:
Three modes

Class 2:
Second and fourth quadrants

Class 2:
Everything else

Class 2:
Everything else
Linear Classifier: Three Viewpoints

### Algebraic Viewpoint

\[ f(x, W) = Wx \]

### Visual Viewpoint

One template per class

### Geometric Viewpoint

Hyperplanes cutting up space
So far: Defined a (linear) score function $f(x, W) = Wx + b$.

Example class scores for 3 images for some W:

<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>score</td>
<td>-3.45</td>
<td>-6.87</td>
<td>0.09</td>
<td>2.9</td>
<td>4.48</td>
<td>8.02</td>
<td>3.78</td>
<td>1.06</td>
<td>-0.36</td>
<td>-0.72</td>
</tr>
</tbody>
</table>

How can we tell whether this W is good or bad?
Coming up:
- Loss function  
  (quantifying what it means to have a “good” $W$)
- Optimization  
  (start with random $W$ and find a $W$ that minimizes the loss)
- ConvNets!  
  (tweak the functional form of $f$)

$$f(x,W) = Wx + b$$