Lecture:

Introduction to Learning Based Vision

Gokberk Cinbis
The modern world of machine learning

Figures from Krizhevsky et al., Shelhamer et al, Johnson et al, van den Oord et al, Silver et al.
Google snaps up object recognition startup DNNresearch

Google has acquired a research startup founded within the University of Toronto, whose work includes object recognition.

by Josh Lowensohn / 13 March 2013, 9:22 am AEDT

Google has acquired a three-person Canadian research company that specializes in voice and image recognition.

DNNresearch, which was founded last year within the the University of Toronto's computer science department, specializes in object recognition and now belongs to Google.

From left: Ilya Sutskever, Alex Krizhevsky and University Professor Geoffrey Hinton of the University of Toronto's Department of Computer Science. (photo by John Guatto, University of Toronto)
Google snaps up object recognition startup DNNresearch

Google has acquired a research startup founded within the University of Toronto, whose work includes object recognition.

by Josh Lowensohn @Josh / 13 March 2013, 9:22 am AEDT

Google has acquired a three-person Canadian research company that specializes in voice and image recognition.

DNNresearch, which was founded last year within the University of Toronto's computer science department, specializes in object recognition and now belongs to Google.
Acquisitions

Google snaps up object recognition startup DNNresearch

Google has acquired a research startup founded within the University of Toronto, whose work includes object recognition.

by Josh Lowensohn @Josh / 13 March 2013, 9:22 am AEDT

Google has acquired a three-person Canadian research company that specializes in voice and image recognition.

DNNresearch, which was founded last year within...

---

8TH ANNUAL CRUNCHIES AWARDS Celebrate the Best of Tech in 2014 Get Your Tickets Now

---

Google Acquires Artificial Intelligence Startup DeepMind

---

Yann LeCun

December 9, 2013 · 🌐

Big news today!

Facebook has created a new research laboratory with the ambitious, long-term goal of bringing about major advances in Artificial Intelligence.
Acquisitions

Google snaps up object recognition startup DNNresearch

Google has acquired DNNresearch, a Toronto, Canada-based research company that develops deep learning software.

San Diego artificial intelligence startup acquired by leading URL application

San Diego artificial intelligence startup acquired by leading URL application.

IBM acquires deep learning startup AlchemyAPI

IBM acquires deep learning startup AlchemyAPI.
What is Machine Learning?

- “the acquisition of knowledge or skills through experience, study, or by being taught.”

- Can be (almost) mapped to reinforcement, unsupervised and supervised machine learning.
What is Machine Learning?

• [Arthur Samuel, 1959]  
  – Field of study that gives computers  
  – the ability to learn without being explicitly programmed

• [Kevin Murphy] algorithms that  
  – automatically detect patterns in data  
  – use the uncovered patterns to predict future data or other outcomes of interest

• [Tom Mitchell] algorithms that  
  – improve their performance (P)  
  – at some task (T)  
  – with experience (E)
What is Machine Learning?

Data → Machine Learning → Understanding

Slide by Dhruv Batra
ML in Nutshell

- Tens of thousands of machine learning algorithms
  - Hundreds new every year

- Decades of ML research oversimplified:
  - All of Machine Learning:
  - Learn a mapping from input to output $f: X \rightarrow Y$
    - e.g. $X$: emails, $Y$: \{spam, notspam\}
Types of Learning

• Supervised learning
  – Training data includes desired outputs

• Unsupervised learning
  – Training data does not include desired outputs

• Weakly or Semi-supervised learning
  – Training data includes a few desired outputs

• Reinforcement learning
  – Rewards from sequence of actions
Tasks

Supervised Learning

x \rightarrow \text{Classification} \rightarrow y \quad \text{Discrete}

x \rightarrow \text{Regression} \rightarrow y \quad \text{Continuous}

Unsupervised Learning

x \rightarrow \text{Clustering} \rightarrow y \quad \text{Discrete ID}

x \rightarrow \text{Dimensionality Reduction} \rightarrow y \quad \text{Continuous}

Slide by Dhruv Batra
Examples for supervised learning
Vision: Image Classification

- [http://cloudcv.org/classify/](http://cloudcv.org/classify/)

Slide by Dhruv Batra
NLP: Machine Translation

\[ x = \begin{array}{cccccc} a_1 = 2 & a_2 = 0 & a_3 = 1 & a_4 = 3 & a_5 = 4 & a_6 = 2 & a_7 = 5 \\ bringen & sie & bitte & das & auto & zurück \end{array} \]

\[ y = \begin{array}{cccc} please & return & the & car \end{array} \]
Speech: Speech2Text
Image captioning

"woman is holding bunch of bananas."

"black cat is sitting on top of suitcase."

http://cs.stanford.edu/people/karpathy/deepimagesent/
AI: Turing Test

“Can machines think”

Q: Please write me a sonnet on the subject of the Forth Bridge.
A: Count me out on this one. I never could write poetry.

Q: Add 34957 to 70764.
A: (Pause about 30 seconds and then give as answer) 105621.
AI: Visual Turing Test

Q: How many slices of pizza are there?

A: 6

Slide by Dhruv Batra

http://cloudcv.org/vqa/
Q: How many slices of pizza are there?

A: 6

Demo: http://cloudcv.org/vqa/

Intro. To Comp. Vision.
Supervised Learning

- Input: $x$ (images, text, emails…)
- Output: $y$ (spam or non-spam…)

Slide by Dhruv Batra
Supervised Learning

• Input: $x$ (images, text, emails…)
• Output: $y$ (spam or non-spam…)
• (Unknown) Target Function
  – $f: X \rightarrow Y$ (the “true” mapping / reality)
Supervised Learning

- Input: \( x \) (images, text, emails…)
- Output: \( y \) (spam or non-spam…)
- (Unknown) Target Function
  - \( f: X \rightarrow Y \) (the “true” mapping / reality)
- Data: \((x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\)
Supervised Learning

- Input: \( x \) (images, text, emails…)
- Output: \( y \) (spam or non-spam…)
- (Unknown) Target Function
  - \( f: X \rightarrow Y \) (the “true” mapping / reality)
- Data: \((x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\)
- Model / Hypothesis Class
  - \( g: X \rightarrow Y \)
  - \( y = g(x) = \text{sign}(w^T x) \)
- Learning = Search in hypothesis space
  - Find best \( g \) in model class.

Slide by Dhruv Batra
Supervised Learning

- Input: $x$ (images, text, emails...)
- Output: $y$ (spam or non-spam...)
- (Unknown) Target Function
  - $f: X \rightarrow Y$ (the “true” mapping / reality)
- Data: $(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$
- Model / Hypothesis Class
  - $g: X \rightarrow Y$
  - $y = g(x) = \text{sign}(w^T x)$
- Learning = Search in hypothesis space
  - Find best $g$ in model class.
Synonyms

• Representation Learning
• Deep (Machine) Learning
• Deep Neural Networks
• Deep Unsupervised Learning
• Simply: Deep Learning
So what is Deep (Machine) Learning?

• A few different ideas:

  • (Hierarchical) Compositionality
    – Cascade of non-linear transformations
    – Multiple layers of representations

  • End-to-End Learning
    – Learning (goal-driven) representations
    – Learning to feature extraction

  • Distributed Representations
    – No single neuron “encodes” everything
    – Groups of neurons work together
Incoming slides and lectures

- Some fundamentals of machine learning
  - A traditional image classification pipeline (Use raw pixels as features and nearest neighbor classifier)
  - Linear classification
  - Loss functions
  - Training by numerical optimization

- Then, deep learning will be a natural extension..
A simple image classification pipeline
Image Classification: a core task in Computer Vision

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

cat
The problem: **semantic gap**

Images are represented as 3D arrays of numbers, with integers between [0, 255].

E.g.
300 x 100 x 3

(3 for 3 color channels RGB)
Challenges: Viewpoint Variation

What the computer sees

Slide by Fei-Fei Li, Andrej Karpathy & Justin Johnson
Challenges: Illumination
Challenges: Deformation
Challenges: Occlusion
Challenges: Background clutter
Challenges: Intraclass variation
An image classifier

```python
def predict(image):
    # ????
    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.

Slide by Fei-Fei Li, Andrej Karpathy & Justin Johnson
Attempts have been made
Data-driven approach:
1. Collect a dataset of images and labels
2. Use Machine Learning to train an image classifier
3. Evaluate the classifier on a withheld set of test images

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

def predict(model, test_images):
    # predict test_labels using the model...
    return test_labels
```

Example training set
First classifier: **Nearest Neighbor Classifier**

Remember all training images and their labels

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

def predict(model, test_images):
    # predict test_labels using the model...
    return test_labels
```

Predict the label of the most similar training image
Example dataset: **CIFAR-10**

10 labels

**50,000** training images, each image is tiny: **32x32**

**10,000** test images.
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.
How do we compare the images? What is the **distance metric**?

**L1 distance:**

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

<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>

Slide by Fei-Fei Li, Andrej Karpathy & Justin Johnson
import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred
import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N ""
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for ""
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred

Slide by Fei-Fei Li, Andrej Karpathy & Justin Johnson
Nearest Neighbor classifier

for every test image:
- find nearest train image with L1 distance
- predict the label of nearest training image

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in range(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred

import numpy as np
import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimensional of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred

Q: how does the classification speed depend on the size of the training data?
import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N ""
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for ""
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros((num_test, dtype = self.ytr.dtype))

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred

Q: how does the classification speed depend on the size of the training data?
linearily :(

This is backwards:
- test time performance is usually much more important in practice.
- CNNs flip this: expensive training, cheap test evaluation
Aside: Approximate Nearest Neighbor
find approximate nearest neighbors quickly

- **ANN**: A Library for Approximate Nearest Neighbor Searching
  - **David M. Mount** and **Sunil Arya**
  - Version 1.1.2
  - Release Date: Jan 27, 2010

**What is ANN?**

ANN is a library written in C++, which supports data structures and algorithms for both exact and approximate nearest neighbor searching in arbitrary high dimensions.

In the nearest neighbor problem a set of data points in d-dimensional space is given. These points are preprocessed into a data structure, so that given any query point q, the nearest or generally k nearest points of P to q can be reported efficiently. The distance between two points can be defined in many ways. ANN assumes that distances are measured using any class of distance functions called Minkowski metrics. These include the well known Euclidean distance, Manhattan distance, and max distance.

Based on our own experience, ANN performs quite efficiently for point sets ranging in size from thousands to hundreds of thousands, and in dimensions as high as 20. (For applications in significantly higher dimensions, the results are rather spotty, but you might try it anyway.)

The library implements a number of different data structures, based on kd-trees and box-decomposition trees, and employs a couple of different search strategies.

The library also comes with test programs for measuring the quality of performance of ANN on any particular data sets, as well as programs for visualizing the structure of the geometric data structures.

- **FLANN - Fast Library for Approximate Nearest Neighbors**

  - What is FLANN?
  - FLANN is a library for performing fast approximate nearest neighbor searches in high dimensional spaces. It contains a collection of algorithms we found to work best for nearest neighbor search and a system for automatically choosing the best algorithm and optimum parameters depending on the dataset.
  - FLANN is written in C++ and contains bindings for the following languages: C, MATLAB and Python.

  - News
    - (14 December 2012) Version 1.8.0 is out bringing incremental addition/removal of points to/from indexes
    - (20 December 2011) Version 1.7.0 is out bringing two new index types and several other improvements.
    - You can find binary installers for FLANN on the Point Cloud Library project page. Thanks to the PCL development!
    - Mac OS X users can install from our project page.
    - New release introducing an easier way to use custom distances, kd-tree-implementation optimized for low dimensionality search and experimental MPI support
    - New release introducing new C++ templated API, thread-safe search, save/load of indexes and more.
    - The FLANN license was changed from LGPL to BSD.

  - How fast is it?
    - In our experiments we have found FLANN to be about one order of magnitude faster on many datasets (in query time), than previously available approximate nearest neighbor search software.

  - Publications
    - More information and experimental results can be found in the following papers:

Slide by Fei-Fei Li, Andrej Karpathy & Justin Johnson
The choice of distance is a hyperparameter common choices:

L1 (Manhattan) distance

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

L2 (Euclidean) distance

\[ d_2(I_1, I_2) = \sqrt{\sum_p (I^p_1 - I^p_2)^2} \]
k-Nearest Neighbor
find the k nearest images, have them vote on the label

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.
Q: what is the accuracy of the nearest neighbor classifier on the training data, when using the Euclidean distance?
Q2: what is the accuracy of the $k$-nearest neighbor classifier on the training data?
What is the best **distance** to use?
What is the best value of **k** to use?

i.e. how do we set the **hyperparameters**?
What is the best distance to use? What is the best value of $k$ to use? i.e. how do we set the hyperparameters?

Very problem-dependent. Must try them all out and see what works best.
Try out what hyperparameters work best on test set.
Trying out what hyperparameters work best on test set:
Very bad idea. The test set is a proxy for the generalization performance!
Use only VERY SPARINGLY, at the end.
Validation data use to tune hyperparameters
Cross-validation

cycle through the choice of which fold is the validation fold, average results.
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 $k \approx 7$ works best for this data)
k-Nearest Neighbor on images is **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)
Summary

- **Image Classification**: We are given a **Training Set** of labeled images, asked to predict labels on **Test Set**. Common to report the **Accuracy** of predictions (fraction of correctly predicted images)

- We introduced the **k-Nearest Neighbor Classifier**, which predicts the labels based on nearest images in the training set

- We saw that the choice of distance and the value of k are **hyperparameters** that are tuned using a **validation set**, or through **cross-validation** if the size of the data is small.

- Once the best set of hyperparameters is chosen, the classifier is evaluated once on the test set, and reported as the performance of kNN on that data.