Introduction to semantic vision
Course announcements

- Homework 5 will be posted tonight.
- Yannis has extra office hours today, 4-8 pm.
- Yannis’ office hours on Friday will be covered by Neeraj.
- How many of you went to Matthias Niessner’s talk today?
Overview of today’s lecture

Leftover from last lecture: radiometric calibration.

New in this lecture:

• Introduction to semantic vision.
• Image classification.
• Bag-of-words.
• K-means clustering.
• Classification.
• K nearest neighbors.
• Naïve Bayes.
• Support vector machine.
Slide credits

Most of these slides were adapted from:

• Kris Kitani (16-385, Spring 2017).
• Noah Snavely (Cornell University).
• Fei-Fei Li (Stanford University).
Course overview

1. Image processing. 
   Lectures 1 – 7
   See also 18-793: Image and Video Processing

2. Geometry-based vision. 
   Lectures 7 – 12
   See also 16-822: Geometry-based Methods in Vision

   Lectures 13 – 16
   See also 16-823: Physics-based Methods in Vision
   See also 15-463: Computational Photography

   We are starting this part now

5. Dealing with motion.
What do we mean by ‘semantic vision’?
Is this a street light? (Verification / classification)
Where are the people? (Detection)
Is that Potala palace? (Identification)
What's in the scene?
(semantic segmentation)
Object categorization

- mountain
- building
- tree
- banner
- street lamp
- vendor
- people
What type of scene is it? (Scene categorization)

Outdoor
Marketplace
City
Activity / Event Recognition

what are these people doing?
Object recognition
Is it really so hard?

Find the chair in this image

Output of normalized correlation

This is a chair
Object recognition
Is it really so hard?

Find the chair in this image

Pretty much garbage
Simple template matching is not going to make it

A “popular method is that of template matching, by point to point correlation of a model pattern with the image pattern. These techniques are inadequate for three-dimensional scene analysis for many reasons, such as occlusion, changes in viewing angle, and articulation of parts.” Nivatia & Binford, 1977.
And it can get a lot harder

How do humans do recognition?

• We don’t completely know yet
• But we have some experimental observations.
Observation 1

• We can recognize familiar faces even in low-resolution images
Observation 2:

- High frequency information is not enough

Jim Carrey

Kevin Costner
What is the single most important facial features for recognition?
What is the single most important facial features for recognition?
Observation 4:

- Image Warping is OK
Spatial configuration matters too
Spatial configuration matters too
The list goes on

Face Recognition by Humans: Nineteen Results All Computer Vision Researchers Should Know About

Variability: Camera position
            Illumination
            Shape parameters

Why is this hard?
How many object categories are there?

~10,000 to 30,000
Challenge: variable viewpoint

Michelangelo 1475-1564
Challenge: variable illumination

image credit: J. Koenderink
and small things from Apple.
(Actual size)

Challenge: scale
Challenge: deformation
Deformation
Challenge: Occlusion

Magritte, 1957
Challenge: background clutter

Kilmeny Niland. 1995
Challenge: Background clutter
Challenge: intra-class variations
Common approaches
Common approaches: object recognition

Feature Matching
Spatial reasoning
Window classification
Feature matching
What object do these parts belong to?
Some local features are very informative.

An object as a collection of local features (bag-of-features)

- deals well with occlusion
- scale invariant
- rotation invariant

Are the positions of the parts important?
Why not use SIFT matching for everything?

- Works well for object *instances*

- Not great for generic object *categories*
Pros

- Simple
- Efficient algorithms
- Robust to deformations

Cons

- No spatial reasoning
Common approaches: object recognition

- Feature Matching
- Spatial reasoning
- Window classification
Spatial reasoning
The position of every part depends on the positions of all the other parts.

Many parts, many dependencies!
1. Extract features
2. Match features
3. Spatial verification
1. Extract features  
2. Match features  
3. Spatial verification
1. Extract features
2. Match features
3. Spatial verification

an old idea...
Fu and Booth. Grammatical Inference. 1975

Scene

Structural (grammatical) description
Coded Chromosome

\[ v_T = \{ \begin{array}{c} \text{a}, \\
\text{b}, \\
\text{c}, \\
\text{d} \end{array} \} \]

\[ x = \text{cdabbdbbabbbcbbabbbbdbbabbb} \]

Substructures of Coded Chromosome

\[ S_1 = \{ [b[[[a]b]b]b]; [b[b[b[a]]b]b]; [b[b[[[a]b]b]b]b]; [b[b[a]]b] \} \]
The Representation and Matching of Pictorial Structures

MARTIN A. FISCHLER and ROBERT A. ELSCLAGER

Abstract—The primary problem dealt with in this paper is the following. Given some description of a visual object, find that object in an actual photograph. Part of the solution to this problem is the specification of a descriptive scheme, and a metric on which to base the decision of "goodness" of matching or detection.

We offer a combined descriptive scheme and decision metric which is general, intuitively satisfying, and which has led to promising experimental results. We also present an algorithm which takes the above descriptions, together with a matrix representing the intensities of the actual photograph, and then finds the described object in the matrix. The algorithm uses a procedure similar to dynamic programming in order to cut down on the vast amount of computation otherwise necessary.

One desirable feature of the approach is its generality. A new programming system does not need to be written for every new description; instead, one just specifies descriptions in terms of a certain set of primitives and parameters.

1972

Description for left edge of face

\[
\text{VALUE}(X) = (E + F + G + H) - (A + B + C + D)
\]

Note: VALUE(X) is the value assigned to the \( L(\text{EV}) \) corresponding to the location X as a function of the intensities of locations A through H in the sensed scene.
A more probabilistic approach…

think of locations as random variables (RV)

vector of RVs: set of part locations

\[ L = \{ L_1, L_2, \ldots, L_M \} \]
A more modern probabilistic approach…

think of locations as random variables (RV)

\[
L = \{L_1, L_2, \ldots, L_M\}
\]

What are the dimensions of R.V. \(L\)?

How many possible combinations of part locations?
A more modern probabilistic approach…

think of locations as random variables (RV)

vector of RVs: set of part locations

$\mathbf{L} = \{L_1, L_2, \ldots, L_M\}$

image (N pixels)

What are the dimensions of R.V. $L$?

$L_m = [x \ y]$  

How many possible combinations of part locations?
A more modern probabilistic approach…

think of locations as random variables (RV)

vector of RVs: set of part locations

\[ L = \{L_1, L_2, \ldots, L_M\} \]

What are the dimensions of R.V. \( L \)?

\[ L_m = [x \ y] \]

How many possible combinations of part locations?

\[ N^M \]
Most likely set of locations $L$ is found by maximizing:

$$p(L|I) \propto p(I|L)p(L)$$

What kind of prior can we formulate?
Given any collection of selfie images, where would you expect the nose to be?

What would be an appropriate prior?

\[ P(L_{\text{nose}}) = ? \]
A simple factorized model

\[ p(L) = \prod_{m} p(L_m) \]

Break up the joint probability into smaller (independent) terms
Independent locations

\[ p(L) = \prod_{m} p(L_m) \]

Each feature is allowed to move independently

Does not model the relative location of parts at all
Tree structure
(star model)

Represent the location of all the parts relative to a single reference part

Assumes that one reference part is defined (who will decide this?)

\[ p(L) = p(L_{\text{root}}) \prod_{m=1}^{M-1} p(L_m|L_{\text{root}}) \]
Fully connected (constellation model)

\[ p(L) = p(l_1, \ldots, l_N) \]

Explicitly represents the joint distribution of locations

Good model: Models relative location of parts
BUT Intractable for moderate number of parts
Pros

- Retains spatial constraints
- Robust to deformations

Cons

- Computationally expensive
- Generalization to large inter-class variation (e.g., modeling chairs)
Feature Matching

Spatial reasoning

Window classification
Window-based
Template Matching

1. get image window
2. extract features
3. classify

*When does this work and when does it fail?*

*How many templates do you need?*
find the ‘nearest’ exemplar, inherit its label
Template Matching

1. get image window (or region proposals)
2. extract features
3. compare to template

Do this part with one big classifier ‘end to end learning’
Convolutional Neural Networks

Convolution

Image patch (raw pixels values)

response of one ‘filter’

A 96 x 96 image convolved with 400 filters (features) of size 8 x 8 generates about 3 million values ($89^2 \times 400$)

Pooling

Image patch (raw pixels values)

response of one ‘filter’

Pooling aggregates statistics and lowers the dimension of convolution

max/min response over a region
630 million connections
60 millions parameters to learn

Krizhevsky, A., Sutskever, I. and Hinton, G. E.
ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012.
Pros

• Retains spatial constraints

• Efficient test time performance

Cons

• Many many possible windows to evaluate

• Requires large amounts of data

• Sometimes (very) slow to train
History of ideas in recognition

- 1960s – early 1990s: the geometric era
- 1990s: appearance-based models
- Mid-1990s: sliding window approaches
- Late 1990s: local features
- Early 2000s: parts-and-shape models
- Mid-2000s: bags of features
- Present trends: data-driven methods, deep learning
What Matters in Recognition?

• Learning Techniques
  – E.g. choice of classifier or inference method

• Representation
  – Low level: SIFT, HoG, GIST, edges
  – Mid level: Bag of words, sliding window, deformable model
  – High level: Contextual dependence
  – Deep features

• Data
  – More is always better
  – Annotation is the hard part
Types of Recognition

• Instance recognition
  • Recognizing a known object but in a new viewpoint, with clutter and occlusion
  • Location/Landmark Recognition
    • Recognize Paris, Rome, ... in photographs
    • Ideas from information retrieval

• Category recognition
  • Harder problem, even for humans
  • Bag of words, part-based, recognition and segmentation
Simultaneous recognition and detection
PASCAL VOC 2005-2012

20 object classes

Classification: person, motorcycle

Detection

Person

Motorcycle

Segmentation

Action: riding bicycle

The PASCAL Visual Object Classes Challenge 2009 (VOC2009)

- 20 object categories (aeroplane to TV/monitor)

- Three (+2) challenges:
  - Classification challenge (is there an X in this image?)
  - Detection challenge (draw a box around every X)
  - Segmentation challenge (which class is each pixel?)
Examples

Aeroplane

Bicycle

Bird

Boat

Bottle

Bus

Car

Cat

Chair

Cow
Classification Challenge

- Predict whether at least one object of a given class is present in an image

is there a cat?
Pascal VOC 2007 Average Precision
Pascal VOC 2012 Average Precision
Detection Challenge

- Predict the bounding boxes of all objects of a given class in an image (if any)
True Positives - Person

UoCTTI_LSVM-MDPM

MIZZOU_DEF-HOG-LBP

NECUIUC_CLS-DTCT
False Positives - Person

UoCTTI_L SVM-MDPM

MIZZOU_DEF-HOG-LBP

NECU1UC_CLS-DTCT
“Near Misses” - Person

UoCTTI_LSV-MDPM

MIZZOU_DEF-HOG-LBP

NECUIUC_CLS-DTCT
True Positives - Bicycle

UoCTTI_LSVM-MDPM

OXFORD_MKL

NECUIUC_CLS-DTCT
False Positives - Bicycle

UoCTTI_LSVM-MDPM

OXFORD_MKL

NECUUIUC_CLS-DTCT
Where to from here?

• Scene Understanding
  • Big data – lots of images
  • Crowd-sourcing – lots of people
  • Deep Learning – lots of compute
installation by Erik Kessels

24 Hrs in Photos

http://www.kesselskramer.com/exhibitions/24-hrs-of-photos
Daily Number of Photos Uploaded & Shared on Select Platforms, 2005 – 2014YTD

- Flickr
- Snapchat
- Instagram
- Facebook
- WhatsApp (2013, 2014 only)

Data Sets

- ImageNet
  - Huge, Crowdsourced, Hierarchical, *Iconic* objects
- PASCAL VOC
  - *Not* Crowdsourced, bounding boxes, 20 categories
- SUN Scene Database, Places
  - *Not* Crowdsourced, 397 (or 720) scene categories
- LabelMe (Overlaps with SUN)
  - Sort of Crowdsourced, Segmentations, Open ended
- SUN *Attribute* database (Overlaps with SUN)
  - Crowdsourced, 102 attributes for every scene
- OpenSurfaces
  - Crowdsourced, materials
- Microsoft COCO
  - Crowdsourced, large-scale objects
Large Scale Visual Recognition Challenge (ILSVRC) 2010-2012

20 object classes 22,591 images
1000 object classes 1,431,167 images

Variety of object classes in ILSVRC

**PASCAL**
- bird
- bottle
- car

**ILSVRC**
- flamingo
- cock
- ruffed grouse
- quail
- partridge
- pill bottle
- beer bottle
- wine bottle
- water bottle
- pop bottle
- race car
- wagon
- minivan
- jeep
- cab
Variety of object classes in ILSVRC

Amount of Texture

Color Distinctiveness

Shape Distinctiveness

Real-world Size
Deep Learning or CNNs

- Since 2012, huge impact..., best results
- Can soak up all the data for better prediction
**IMAGENET Large Scale Visual Recognition Challenge**

**Year 2010**
- NEC-UIUC
  - Dense grid descriptor: HOG, LBP
  - Coding: local coordinate, super-vector
  - Pooling, SPM
  - Linear SVM

[Lin CVPR 2011]

**Year 2012**
- SuperVision

[Krizhevsky NIPS 2012]

**Year 2014**
- GoogLeNet
  - VGG
  - MSRA

[Simonyan arxiv 2014] [He arxiv 2014]
Image classification
Image Classification

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

→ cat
Image Classification: Problem

What the computer sees:

- 82% cat
- 15% dog
- 2% hat
- 1% mug

_image classification_
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
Bag of words
What object do these parts belong to?
Some local features are very informative.

An object as a collection of local features (bag-of-features)

- deals well with occlusion
- scale invariant
- rotation invariant
(not so) crazy assumption

spatial information of local features can be ignored for object recognition (i.e., verification)
CalTech6 dataset

<table>
<thead>
<tr>
<th>class</th>
<th><strong>bag of features</strong></th>
<th><strong>bag of features</strong></th>
<th><strong>Parts-and-shape model</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>airplanes</td>
<td>98.8</td>
<td>97.1</td>
<td>90.2</td>
</tr>
<tr>
<td>cars (rear)</td>
<td>98.3</td>
<td><strong>98.6</strong></td>
<td>90.3</td>
</tr>
<tr>
<td>cars (side)</td>
<td><strong>95.0</strong></td>
<td>87.3</td>
<td>88.5</td>
</tr>
<tr>
<td>faces</td>
<td>100</td>
<td>99.3</td>
<td>96.4</td>
</tr>
<tr>
<td>motorbikes</td>
<td>98.5</td>
<td>98.0</td>
<td>92.5</td>
</tr>
<tr>
<td>spotted cats</td>
<td>97.0</td>
<td>—</td>
<td>90.0</td>
</tr>
</tbody>
</table>

Works pretty well for image-level classification
Bag-of-features

represent a data item (document, texture, image) as a histogram over features

an old idea
(e.g., texture recognition and information retrieval)
Texture recognition

Universal texton dictionary

histogram
Vector Space Model

G. Salton. 'Mathematics and Information Retrieval' Journal of Documentation, 1979

http://www.fidey.com/generators/newspaper/snippet.asp
A document (datapoint) is a vector of counts over each word (feature)

\[ \mathbf{v}_d = [n(w_1,d) \ n(w_2,d) \ \cdots \ n(w_T,d)] \]

\( n(\cdot) \) counts the number of occurrences

What is the similarity between two documents?
A document (datapoint) is a vector of counts over each word (feature)

\[ \mathbf{v}_d = [n(w_{1,d}) \ n(w_{2,d}) \ \cdots \ n(w_{T,d})] \]

\( n(\cdot) \) counts the number of occurrences

What is the similarity between two documents?

Use any distance you want but the cosine distance is fast.

\[ d(\mathbf{v}_i, \mathbf{v}_j) = \cos \theta \]

\[ = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|} \]
but not all words are created equal
**TF-IDF**

**Term Frequency Inverse Document Frequency**

\[ \mathbf{v}_d = [n(w_1,d) \quad n(w_2,d) \quad \cdots \quad n(w_T,d)] \]

weigh each word by a heuristic

\[ \mathbf{v}_d = [n(w_1,d)\alpha_1 \quad n(w_2,d)\alpha_2 \quad \cdots \quad n(w_T,d)\alpha_T] \]

\[ n(w_i,d)\alpha_i = n(w_i,d) \log \left\{ \frac{D}{\sum_{d'} 1[w_i \in d']} \right\} \]

(down-weights common terms)
Standard BOW pipeline
(for image classification)
Dictionary Learning:
Learn Visual Words using clustering

Encode:
build Bags-of-Words (BOW) vectors for each image

Classify:
Train and test data using BOWs
Dictionary Learning:
Learn Visual Words using clustering

1. extract features (e.g., SIFT) from images
Dictionary Learning:
Learn Visual Words using clustering

2. Learn visual dictionary (e.g., K-means clustering)
What kinds of features can we extract?
• Regular grid
  • Vogel & Schiele, 2003
  • Fei-Fei & Perona, 2005

• Interest point detector
  • Csurka et al. 2004
  • Fei-Fei & Perona, 2005
  • Sivic et al. 2005

• Other methods
  • Random sampling (Vidal-Naquet & Ullman, 2002)
  • Segmentation-based patches (Barnard et al. 2003)
Normalize patch

Detect patches

[Mikojaczyk and Schmid ’02]
[Mata, Chum, Urban & Pajdla, ’02]
[Sivic & Zisserman, ’03]

Compute SIFT descriptor

[Lowe’99]
How do we learn the dictionary?
K-means clustering
1. Select initial centroids at random
1. Select initial centroids at random

2. Assign each object to the cluster with the nearest centroid.
1. Select initial centroids at random

2. Assign each object to the cluster with the nearest centroid.

3. Compute each centroid as the mean of the objects assigned to it (go to 2)
1. Select initial centroids at random.

2. Assign each object to the cluster with the nearest centroid.

3. Compute each centroid as the mean of the objects assigned to it (go to 2).

2. Assign each object to the cluster with the nearest centroid.
1. Select initial centroids at random

2. Assign each object to the cluster with the nearest centroid.

3. Compute each centroid as the mean of the objects assigned to it (go to 2)

Repeat previous 2 steps until no change
K-means Clustering

Given k:
1. Select initial centroids at random.
2. Assign each object to the cluster with the nearest centroid.
3. Compute each centroid as the mean of the objects assigned to it.
4. Repeat previous 2 steps until no change.
From what **data** should I learn the dictionary?
From what **data** should I learn the dictionary?

- Dictionary can be learned on separate training set
- Provided the training set is sufficiently representative, the dictionary will be “universal”
Example visual dictionary
Example dictionary

Source: B. Leibe
Another dictionary

Source: B. Leibe
Dictionary Learning:
Learn Visual Words using clustering

Encode:
build Bags-of-Words (BOW) vectors for each image

Classify:
Train and test data using BOWs
1. Quantization: image features gets associated to a visual word (nearest cluster center)

**Encode:**
build Bags-of-Words (BOW) vectors for each image
Encode:
build Bags-of-Words (BOW) vectors for each image

2. Histogram: count the number of visual word occurrences
Dictionary Learning:
Learn Visual Words using clustering

Encode:
build Bags-of-Words (BOW) vectors for each image

Classify:
Train and test data using BOWs
K nearest neighbors

Naïve Bayes

Support Vector Machine
K nearest neighbors
Distribution of data from two classes
Distribution of data from two classes

Which class does q belong too?
Distribution of data from two classes

Look at the neighbors

$q$
K-Nearest Neighbor (KNN) Classifier

Non-parametric pattern classification approach

Consider a two class problem where each sample consists of two measurements \((x,y)\).

For a given query point \(q\), assign the class of the nearest neighbor

Compute the \(k\) nearest neighbors and assign the class by majority vote.
Nearest Neighbor is competitive

<table>
<thead>
<tr>
<th>Test Error Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear classifier (1-layer NN)</td>
</tr>
<tr>
<td>K-nearest-neighbors, Euclidean</td>
</tr>
<tr>
<td>K-nearest-neighbors, Euclidean, deskewed</td>
</tr>
<tr>
<td>K-NN, Tangent Distance, 16x16</td>
</tr>
<tr>
<td>K-NN, shape context matching</td>
</tr>
<tr>
<td>1000 RBF + linear classifier</td>
</tr>
</tbody>
</table>

**MNIST Digit Recognition**
- Handwritten digits
- 28x28 pixel images: \( d = 784 \)
- 60,000 training samples
- 10,000 test samples

Yann LeCunn
What is the best distance metric between data points?

- Typically Euclidean distance

- Locality sensitive distance metrics

- Important to normalize. Dimensions have different scales

How many K?

- Typically k=1 is good

- Cross-validation (try different k!)
Distance metrics

\[ D(x, y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_N - y_N)^2} \quad \text{Euclidean} \]

\[ D(x, y) = \frac{x \cdot y}{\|x\| \|y\|} = \frac{x_1 y_1 + \cdots + x_N y_N}{\sqrt{\sum_n x_n^2} \sqrt{\sum_n y_n^2}} \quad \text{Cosine} \]

\[ D(x, y) = \frac{1}{2} \sum_n \frac{(x_n - y_n)^2}{x_n + y_n} \quad \text{Chi-squared} \]
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

\[ \|x\|_p = (|x_1|^p + \cdots + |x_n|^p)^{\frac{1}{p}} \quad p \geq 1, x \in \mathbb{R}^n \]
Visualization: L2 distance
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
- Labels of the $k$ points “vote” to classify
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 and see what works best
Try out what hyperparameters work best on the test set.

train data

---

test data
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
evaluate on test set ONCE at the end
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)
How to pick hyperparameters?

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

• Train for original model
• Validate to find hyperparameters
• Test to understand generalizability
Pros
  – simple yet effective

Cons
  – search is expensive (can be sped-up)
  – storage requirements
  – difficulties with high-dimensional data
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)
Naïve Bayes
Which class does $q$ belong to?
Distribution of data from two classes

- Learn parametric model for each class
- Compute probability of query
This is called the posterior.
The probability of a class $z$, given the observed features $X$:

$$p(z|X)$$

For classification, $z$ is a discrete random variable (e.g., car, person, building)

$X$ is a set of observed features (e.g., features from a single image)

(it’s a function that returns a single probability value)
This is called the posterior: the probability of a class $z$, given the observed features $X$

$$p(z|x_1, \ldots, x_N)$$

For classification, $z$ is a discrete random variable (e.g., car, person, building)

Each $x$ is an observed feature (e.g., visual words)

(it’s a function that returns a single probability value)
Recall:

The posterior can be decomposed according to Bayes’ Rule

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

In our context…

$$p(z|x_1, \ldots, x_N) = \frac{p(x_1, \ldots, x_N|z)p(z)}{p(x_1, \ldots, x_N)}$$
The naive Bayes’ classifier is solving this optimization

$$\hat{z} = \arg\max_{z \in \mathcal{Z}} p(z|X)$$

MAP (maximum a posteriori) estimate

$$\hat{z} = \arg\max_{z \in \mathcal{Z}} \frac{p(X|z)p(z)}{p(X)}$$  \hspace{1cm} \text{Bayes’ Rule}

$$\hat{z} = \arg\max_{z \in \mathcal{Z}} p(X|z)p(z)$$  \hspace{1cm} \text{Remove constants}

To optimize this…we need to compute this

Compute the likelihood…
A naive Bayes’ classifier assumes all features are \textit{conditionally independent}

\[ p(x_1, \ldots, x_N | z) = p(x_1 | z)p(x_2, \ldots, x_N | z) \]
\[ = p(x_1 | z)p(x_2 | z)p(x_3, \ldots, x_N | z) \]
\[ = p(x_1 | z)p(x_2 | z) \cdots p(x_N | z) \]

**Recall:**

\[ p(x, y) = p(x | y)p(y) \]
\[ p(x, y) = p(x)p(y) \]
To compute the MAP estimate

Given (1) a set of known parameters \( p(z) \) \( p(x|z) \) \( \{x_1, x_2, \ldots, x_N\} \)

(2) observations

\[
\hat{z} = \arg \max_{z \in Z} p(z) \prod_{n} p(x_n | z)
\]
DARPA Selects Carnegie Mellon University’s National Robotics Engineering Center ranked third among teams competing in the Defense Advanced Research Projects Agency (DARPA) Robotics Challenge Trials this weekend in Homestead, Fla., and was selected by the agency as one of eight teams eligible for DARPA funding to prepare for next year’s final. The full team presented CMU’s arm-robot CMU Highly Intelligent Mobile Platform, or CHIMP robot. The team scored 18 out of 100 possible points, placing 22 percent above the 32 percent during the six month period that the CMU team was preparing for the challenge.

<table>
<thead>
<tr>
<th>count</th>
<th>1</th>
<th>6</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>word</td>
<td>Tartan</td>
<td>robot</td>
<td>CHIMP</td>
<td>CMU</td>
<td>bio</td>
<td>soft</td>
<td>ankle</td>
<td>sensor</td>
<td></td>
</tr>
<tr>
<td>p(x</td>
<td>z)</td>
<td>0.09</td>
<td>0.55</td>
<td>0.18</td>
<td>0.09</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.09</td>
</tr>
</tbody>
</table>

\[
p(X|z) = \prod_{v} p(x_v|z)^{c(w_v)}
\]

\[
= (0.09)^1 (0.55)^6 \cdots (0.09)^1
\]

Numbers get really small so use log probabilities

\[
\log p(X|z = \text{‘grandchallenge’}) = -2.42 - 3.68 - 3.43 - 2.42 - 0.07 - 0.07 - 0.07 - 2.42 = -14.58
\]

\[
\log p(X|z = \text{‘softrobot’}) = -7.63 - 9.37 - 15.18 - 2.97 - 0.02 - 0.01 - 0.02 - 2.27 = -37.48
\]

* typically add pseudo-counts (0.001)

** this is an example for computing the likelihood, need to multiply times prior to get posterior
log \( p(X|z=\text{grand challenge}) \) = - 14.58

log \( p(X|z=\text{bio inspired}) \) = - 37.48

log \( p(X|z=\text{grand challenge}) \) = - 94.06

log \( p(X|z=\text{bio inspired}) \) = - 32.41

* typically add pseudo-counts (0.001)

** this is an example for computing the likelihood, need to multiply times prior to get posterior
Support Vector Machine
Image Classification

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

→ cat
Score function

class scores
Linear Classifier

define a score function

data (histogram)

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

class scores

“weights”

“bias vector”

“parameters”
Example with an image with 4 pixels, and 3 classes (cat/dog/ship).
Which class does $q$ belong too?
Distribution of data from two classes

Learn the decision boundary
First we need to understand hyperplanes...
Hyperplanes (lines) in 2D

\[ w_1 x_1 + w_2 x_2 + b = 0 \]

A line can be written as the dot product plus a bias:

\[ \mathbf{w} \cdot \mathbf{x} + b = 0 \]

Another version, add a weight 1 and push the bias inside:

\[ \mathbf{w} \cdot \mathbf{x} = 0 \]

\[ \mathbf{w} \in \mathbb{R}^3 \]
Hyperplanes (lines) in 2D

\[ \mathbf{w} \cdot \mathbf{x} + b = 0 \quad \text{(offset/bias outside)} \]
\[ \mathbf{w} \cdot \mathbf{x} = 0 \quad \text{(offset/bias inside)} \]

\[ w_1 x_1 + w_2 x_2 + b = 0 \]
Hyperplanes (lines) in 2D

\[ \mathbf{w} \cdot \mathbf{x} + b = 0 \quad \text{(offset/bias outside)} \quad \mathbf{w} \cdot \mathbf{x} = 0 \quad \text{(offset/bias inside)} \]

\[ w_1 x_1 + w_2 x_2 + b = 0 \]

**Important property:**
*Free to choose any normalization of \( w \)*

The line

\[ w_1 x_1 + w_2 x_2 + b = 0 \]

and the line

\[ \lambda (w_1 x_1 + w_2 x_2 + b) = 0 \]

define the same line
What is the distance to origin?

(hint: use normal form)
You get the normal form

\[ \frac{b}{\|w\|} \]

distance to origin

\[ w \cdot x + b = 0 \]

scale \( w \cdot x + b = 0 \) by \( \frac{1}{\|w\|} \)

you get the normal form

\[ x \cos \theta + y \sin \theta = \rho \]
What is the distance between two parallel lines? (hint: use distance to origin)
distance between two parallel lines $\frac{1}{\|w\|}$

$w \cdot x + b = -1$

$w \cdot x + b = 0$

Difference of distance to origin

$$\frac{b + 1}{\|w\|} - \frac{b}{\|w\|} = \frac{1}{\|w\|}$$
Hyperplanes (planes) in 3D

Now we can go to 3D …

What happens if you change $b$?

What are the dimensions of this vector?

$w \cdot x + b = 0$
Hyperplanes (planes) in 3D

\[ \frac{b + 1}{\|w\|} \]

\[ w \cdot x + b = -1 \]
What's the distance between these parallel planes?

Hyperplanes (planes) in 3D

\[ \mathbf{w} \cdot \mathbf{x} + b = -1 \]
\[ \mathbf{w} \cdot \mathbf{x} + b = 0 \]
\[ \mathbf{w} \cdot \mathbf{x} + b = 1 \]
Hyperplanes (planes) in 3D

\[
\frac{2}{\| \mathbf{w} \|}
\]

\[
\mathbf{w} \cdot \mathbf{x} + b = -1
\]

\[
\mathbf{w} \cdot \mathbf{x} + b = 0
\]

\[
\mathbf{w} \cdot \mathbf{x} + b = 1
\]
What’s the best $w$?
What’s the best $w$?
What’s the best $w$?
What’s the best $\mathbf{w}$?
What’s the best $w$?
What’s the best \( \mathbf{w} \)?

Intuitively, the line that is the farthest from all interior points.
What’s the best \( \mathbf{w} \)?

**Maximum Margin solution:**
most stable to perturbations of data
What’s the best $w$?

Want a hyperplane that is far away from ‘inner points’
Find hyperplane $\mathbf{w}$ such that …

the gap between parallel hyperplanes $\frac{2}{\|\mathbf{w}\|}$ is maximized.
Can be formulated as a maximization problem

\[
\max_{\mathbf{w}} \frac{2}{\|\mathbf{w}\|}
\]

subject to \( \mathbf{w} \cdot \mathbf{x}_i + b \geq +1 \) if \( y_i = +1 \)

\( \leq -1 \) if \( y_i = -1 \)

for \( i = 1, \ldots, N \)

What does this constraint mean?

Why is it +1 and -1?

label of the data point
Can be formulated as a maximization problem

$$\max_{\mathbf{w}} \frac{2}{\|\mathbf{w}\|}$$

subject to $\mathbf{w} \cdot \mathbf{x}_i + b \geq +1$ if $y_i = +1$

subject to $\mathbf{w} \cdot \mathbf{x}_i + b \leq -1$ if $y_i = -1$

for $i = 1, \ldots, N$

Equivalently,

$$\min_{\mathbf{w}} \|\mathbf{w}\|$$

subject to $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$ for $i = 1, \ldots, N$
‘Primal formulation’ of a linear SVM

\[
\min_w \|w\|
\]

Objective Function

Subject to \( y_i (w \cdot x_i + b) \geq 1 \) for \( i = 1, \ldots, N \)

Constraints

This is a convex quadratic programming (QP) problem
(a unique solution exists)
‘soft’ margin
What’s the best \textbf{w}?
What’s the best $w$?
Separating cats and dogs

Very narrow margin
What’s the best $\mathbf{w}$?

Intuitively, we should allow for some misclassification if we can get more robust classification.
What’s the best \( w \)?

Trade-off between the MARGIN and the MISTAKES
(might be a better solution)
Adding slack variables $\xi_i \geq 0$

misclassified point

$\frac{\xi_i}{\|w\|} > \frac{2}{\|w\|}$
‘soft’ margin

objective

\[
\min_{\mathbf{w}, \xi} \|\mathbf{w}\|^2 + C \sum_i \xi_i
\]

subject to

\[
y_i (\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \\
\text{for } i = 1, \ldots, N
\]
The slack variable allows for mistakes, as long as the inverse margin is minimized.
'soft' margin

Objective

\[
\min_{\mathbf{w}, \xi} \| \mathbf{w} \|^2 + C \sum_i \xi_i
\]

Subject to

\[
y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \\
\text{for } i = 1, \ldots, N
\]

- Every constraint can be satisfied if slack is large
- C is a regularization parameter
  - Small C: ignore constraints (larger margin)
  - Big C: constraints (small margin)
- Still QP problem (unique solution)
C = Infinity  hard margin

SVM (L1) by Sequential Minimal Optimizer
Kernel: linear (-), C: Inf
Kernel evaluations: 971
Number of Support Vectors: 3
Margin: 0.0966
Training error: 0.00%
C = 10  soft margin

Comment Window
SVM (L1) by Sequential Minimal Optimizer
Kernel: linear (-), C: 10.0000
Kernel evaluations: 2645
Number of Support Vectors: 4
Margin: 0.2265
Training error: 3.70%
References

Basic reading:
• Szeliski, Chapter 14.