Today

Machine Learning for Visual Tasks
Learning Pipelines
Nearest Neighbor Classifiers
Linear Classifiers
Generalization
Recognition

Image Understanding
Recognition | Tasks

**Classification:** Is this a street?
Recognition | Tasks

**Detection**: Is there a bus?
Recognition | Tasks

Segmentation: Where (exactly) is the bus?
Recognition | Tasks

Verification: Is this thing a billboard?
Recognition | Tasks

Identification: Is this Causeway Bay in Hong Kong?
Recognition | Tasks

Categorization
Recognition | Tasks

Scene Categorization

- Outside
- Street
- Day
- Crosswalk
- Junction
- ...
Challenges | Scale

~100,000 nouns in the (spoken) English language
~21,000 object categories in ImageNet
Estimated 30,000-40,000 object types
Challenges | Viewpoint
Challenges | Occlusion

Magritte, 1964
Challenges | Intra-Class Variation

Car

Car

Car
Algorithms that improve their prediction performance at some task with experience (or data).

Machine Learning for Visual Tasks

Data (experience) \[ \rightarrow \] \text{Learning Algorithm (task)} \[ \rightarrow \] Understanding (performance)

- Outside
- Street
- Day
- Crosswalk
- Junction
- ...

[Orabona]
Computer vision models

- Observe measured data, $x$
- Draw inferences from it about state of world, $w$

Examples:
- Observe adjacent frames in video sequence
- Infer camera motion
- Observe image of face
- Infer identity
- Observe images from two displaced cameras
- Infer 3d structure of scene

Regression vs. Classification

- Observe measured data, $x$
- Draw inferences from it about world, $w$

When the world state $w$ is continuous we’ll call this regression

When the world state $w$ is discrete we call this classification
Detection By Classification

If we had a function that can tell us if an image contains an object...

\[ f(\text{person}) = \checkmark \]
\[ f(\text{background}) = \times \]

We can detect all objects in the scene.

We can just slide it over all pixels in the image:

Downside: slow
We seek to find a predictor function $f$ that given an input image, or its features vector, $x$ will output a class assignment $y$ (classify it).

$$f(x) = y$$

**Training**: Find the parameters for $f$ that minimize an error between $y_{\text{predicted}}$ and $y_{\text{groundtruth}}$ on a set of samples.

**Testing**: Evaluate $f$ on unused (held-out) samples.

The decisions $f$ makes divides the input space with *decision boundaries* into *decision regions*. 
A simple pipeline

Training

Training Images

Image Features

Training Labels

Learned Classifier

Testing

Test Image

Image Features

Learned Classifier

Prediction

“Offline Learning”

Deploy the model

Dataset: ETH-80, by B. Leibe

Slide credit: D. Holec, L. Lazebnik

Fei-Fei Li

Lecture 16
Traditional ML Pipeline

- Features are not learned
- Trainable classifier is often generic (e.g. SVM)

SIFT, Bag-of-Words, Histograms

[LaZebnik]
Nearest Neighbor Classifier

The simplest idea: Given an input image, find the closest prior (training) sample and return its label.

Distance metric can be chosen: $L_2$, $L_1$, K-L, cross-correlation, etc.

$k$-Nearest Neighbor: Find the closest $k$ neighbors and make a decision (the mean, median, etc.)
Nearest Neighbor Classifier

Pros:
- Simple
- Flexible decision boundaries (explicit)

Cons:
- Slow. Must check all training samples... (can be made more efficient)
- Doesn’t “learn” anything about the data, hard to generalize.
- Has a parameter $k$, which is difficult to choose

3-Class classification ($k = 10$, weights = 'distance')
Nearest Neighbor Classifier | Choose $k$

3-Class classification ($k = 15$, weights = 'distance')

3-Class classification ($k = 1$, weights = 'distance')

Too big: high FP rate

Too small: noisy
Linear Classifier

k-NN classifiers:
- Must "remember" all the data to make a decision
- Prediction is expensive since need to go through all of "memory"

Linear classification - A more powerful idea:
- A simple, linear scoring function for data samples
- A decision function over the score
- An optimization scheme to find the best score and decision functions, given a loss function
- Throw away training data when done!

[Karpathy]
Linear Classifier

Simple image parametrization: Each pixel is a number in a very long vector.

Linear score function: \( f(x_i, W, b) = Wx_i + b \)

Need to learn: \( W \) and \( b \)
Linear Classifier

Simple image parametrization: Each pixel is a number in a very long vector.

Linear score function: \( f(x_i, W, b) = Wx_i + b \)

Need to learn: \( W \) and \( b \)

This operation reminds you of something? Hint: Think geometrically...

[Image of a cat, a table, and a calculation]
Similar to Homogeneous Coordinates: Add 1 in the end of the vector to allow for translation.
To find $W, b$ we need to know how to select the best ones.

We define a loss function:

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

AKA the **Support Vector** or “max-margin” loss.

Penalize incorrect class assignments ($j \neq y_i$) by taking the difference in score with the ground truth (annotated data) score ($y_i$).

$\Delta$ is the margin.

**Example:**

$$s = [13, -7, 11] \quad y_i = 0$$

$$L_i = \max(0, -7 - 13 + 10) + \max(0, 11 - 13 + 10) = 8$$
Questions?
Support Vector Machines

- When the data is linearly separable, there may be more than one separator (hyperplane)

Which separator is best?

[LaZebnik]
Support Vector Machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples

\[ x_i, \text{positive} \ (y_i = 1) : \quad x_i \cdot w + b \geq 1 \]
\[ x_i, \text{negative} \ (y_i = -1) : \quad x_i \cdot w + b \leq -1 \]

For support vectors, \( x_i \cdot w + b = \pm 1 \)

Distance between point and hyperplane:
\[ \frac{|x_i \cdot w + b|}{\|w\|} \]

Therefore, the margin is \( \frac{2}{\|w\|} \)


[Lazebnik]
Support Vector Machines | Max Margin

1. Maximize margin $2 / \|w\|$
2. Correctly classify all training data:
   - $x_i$ positive ($y_i = 1$): $x_i \cdot w + b \geq 1$
   - $x_i$ negative ($y_i = -1$): $x_i \cdot w + b \leq -1$

**Quadratic optimization problem:**

$$\min_{w,b} \frac{1}{2} \|w\|^2 \quad \text{subject to} \quad y_i (w \cdot x_i + b) \geq 1$$

Support Vector Machines | Loss Model

- Separable data:
  \[
  \min_{w,b} \frac{1}{2} \|w\|^2 \quad \text{subject to} \quad y_i (w \cdot x_i + b) \geq 1
  \]
  - Maximize margin
  - Classify training data correctly

- Non-separable data:
  \[
  \min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i (w \cdot x_i + b))
  \]
  - Maximize margin
  - Minimize classification mistakes

[Lazebnik]
Support Vector Machines | Loss Model

\[ \min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \max(0,1 - y_i(w \cdot x_i + b)) \]

Demo: [http://cs.stanford.edu/people/karpathy/hsvmjs/demo](http://cs.stanford.edu/people/karpathy/hsvmjs/demo)
Nonlinear SVMs

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable.
Nonlinear SVMs

- Linearly separable dataset in 1D:
- Non-separable dataset in 1D:
- We can map the data to a higher-dimensional space:
Nonlinear SVMs

- The kernel trick: instead of explicitly computing the lifting transformation $\varphi(x)$, define a kernel function $K$ such that

$$K(x, y) = \varphi(x) \cdot \varphi(y)$$

$${\mathbf{w}} \cdot {\mathbf x} + b = \sum_i \alpha_i y_i {\mathbf x}_i \cdot {\mathbf x} + b \quad \Rightarrow \quad \sum_i \alpha_i y_i \varphi({\mathbf x}_i) \cdot \varphi({\mathbf x}) + b = \sum_i \alpha_i y_i K({\mathbf x}_i, {\mathbf x}) + b$$

A nonlinear decision boundary in the data space.
Still linear in the parameter space.

[Lazebnik]
Nonlinear SVMs

Polynomial kernel: \( K(x, y) = (c + x \cdot y)^d \)
Nonlinear SVMs

Polynomial kernel: \( K(x, y) = (c + x \cdot y)^d \)
Nonlinear SVMs

Gaussian kernel

- Also known as the radial basis function (RBF) kernel:

\[ K(x, y) = \exp\left( -\frac{1}{\sigma^2} \|x - y\|^2 \right) \]
Linear Classifiers | Regularization

What if we found $W$ s.t. all images are correctly classified, i.e $L_i = 0$ for each $i$.

Remember: $L_i = Wx_i = 0$

Have we seen this before? (yes :)

What can happen?

How can we deal with it?
**Linear Classifiers | Regularization**

What if we found $W$ s.t. all images are correctly classified, i.e $L_i = 0$ for each $i$.

$W$ is not necessarily unique! It has **scale ambiguity** since: $L_i = Wx_i = 0$ can be also $2Wx_i = 5Wx_i = 0$

$W$ can get any scale, so it may “blow up” in optimization.

We apply **regularization** to keep $W$ as small as possible.

An extended loss function:

$$L = \frac{1}{N} \sum_i L_i + \lambda R(W)$$

With an L$_2$ **regularization term**:

$$R(W) = \sum_k \sum_l W^2_{k,l}$$

Regularization also helps **generalization**! Since no one class $j$ can “take over” by setting its $W_{(j)}$ row to be very big.
Questions?
SVMs: Pros and cons

• Pros
  • Kernel-based framework is very powerful, flexible
  • Training is convex optimization, globally optimal solution can be found
  • Amenable to theoretical analysis
  • SVMs work very well in practice, even with very small training sample sizes

• Cons
  • No “direct” multi-class SVM, must combine two-class SVMs (e.g., with one-vs-others)
  • Computation, memory (esp. for nonlinear SVMs)
SVM gives (hard to interpret...) scores to each class.

**Softmax**: Give probabilities for each class - makes for a lot more intuitive classifier!

Keep \( f(W, x_i) = Wx_i \), look at \( f_j \) - the \( j \)-th position in the score vector.

New loss function:

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

or equivalently

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

SVM gives (hard to interpret...) scores to each class.

**Softmax**: Give probabilities for each class - makes for a lot more intuitive classifier!

Keep \( f(W, x_i) = Wx_i \), look at \( f_j \) - the \( j \)-th position in the score vector.

New loss function (cross-entropy loss):

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

Softmax: take scores \( f_j \) and squash them to a vector of **[0,1]** values that also sum up to 1.

It’s a probability density function.

In particular, a PDF of the estimated class probabilities.

\[
P(y_i \mid x_i; W) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \quad \leftarrow \text{Log prob.}
\]

\[
P(y_i \mid x_i; W) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \quad \leftarrow \text{Normalize}
\]

Probability of correct label \( y_i \) given image \( x_i \) and \( W \).
Questions?
Next Time...

Hyperparameters tuning

More non-linear classification

Stochastic Gradient Descent

Neural Networks

Convolutional Neural Networks
Wrap Up

What we’ve learned today:

- Machine Learning
- ML Pipelines
- Linear Classification
  - Simple Linear Classifier
  - Max-margin, hinge loss, SVMs
  - Softmax