CS 1674: Intro to Computer Vision
Visual Recognition

Prof. Seong Jae Hwang
University of Pittsburgh
March 11, 2021
Plan for this lecture

• What is recognition?
  – a.k.a. classification, categorization

• Support vector machines (SVM)
  – Separable case / non-separable case
  – Linear / non-linear (kernels)

• The importance of generalization
  – The bias-variance trade-off (applies to all classifiers)

• Example approach for scene classification
Classification

- Given a feature representation for images, how do we learn a model for distinguishing features from different classes?
Classification

• Assign input vector to one of two or more classes
• Input space divided into *decision regions* separated by *decision boundaries*

\[ x_2 \]
\[ x_1 \]
\[ \mathcal{R}_1 \]
\[ \mathcal{R}_2 \]
\[ \mathcal{R}_3 \]
Examples of image classification

• Two-class (binary): Cat vs Dog
Examples of image classification

• Multi-class (often): Object recognition
Examples of image classification

- Fine-grained recognition

Visipedia Project
Examples of image classification

- Place recognition

Places Database [Zhou et al. NIPS 2014]
Examples of image classification

• Material recognition

[Bell et al. CVPR 2015]
Examples of image classification

• Dating historical photos

[Palermo et al. ECCV 2012]
Examples of image classification

- Image style recognition

Flickr Style: 80K images covering 20 styles.

Wikapaintings: 85K images for 25 art genres.

[Karayev et al. BMVC 2014]
Recognition: A machine learning approach

• Suppose this is our dataset with multiple classes:
  – Apple, pear, tomato, cow, dog, horse
The machine learning framework

- Apply a prediction function to a feature representation of the image to get the desired output:

\[ f(\text{apple}) = \text{“apple”} \]
\[ f(\text{tomato}) = \text{“tomato”} \]
\[ f(\text{cow}) = \text{“cow”} \]
The machine learning framework

\[ y = f(x) \]

- **Training:** given a *training set* of labeled examples \( \{(x_1,y_1), \ldots, (x_N,y_N)\} \), estimate the prediction function \( f \) by minimizing the prediction error on the training set
  - Evaluate multiple hypotheses \( f_1, f_2, f_H \ldots \) and pick the best one as \( f \)
- **Testing:** apply \( f \) to a *never before seen* test example \( x \) and output the predicted value \( y = f(x) \)

Slide credit: L. Lazebnik
The old-school way

Training

Training Images

Training Labels

Image Features

Training

Learned model

Testing

Test Image

Image Features

Learned model

Prediction

Slide credit: D. Hoiem and L. Lazebnik
The simplest classifier

\[ f(x) = \text{label of the training example nearest to } x \]

- All we need is a distance function for our inputs
- No training required!
K-Nearest Neighbors (KNN) classification

- For a new point, find the $k$ closest points from training data
- Labels of the $k$ points “vote” to classify

Black = negative  
Red = positive

If query lands here, the 5 NN consist of 3 negatives and 2 positives, so we classify it as negative.
im2gps: Estimating Geographic Information from a Single Image
James Hays and Alexei Efros, CVPR 2008

Where was this image taken?

Nearest Neighbors according to bag of SIFT + color histogram + a few others
The Importance of Data

Percentage of Geolocations within 200km

- First Nearest Neighbor Scene Match
- Chance: Random Scenes

Database size (thousands of images, log scale)

Slides: James Hays
KNN Demo

• https://lecture-demo.ira.uka.de/knn-demo/#
Linear classifier

- Find a **linear function** to separate the classes

\[ f(x) = \text{sign}(w_1x_1 + w_2x_2 + \ldots + w_Dx_D) = \text{sign}(\mathbf{w} \cdot \mathbf{x}) \]

\( \text{sign}(>0) \rightarrow \text{pos}, \text{sign}(<0) \rightarrow \text{neg} \)
Linear classifier

- Decision = \( \text{sign}(w^T x) = \text{sign}(w_1 x_1 + w_2 x_2) \)

- What should the weights \([w_1, w_2]\) be?
  - \([w_1, w_2]\) such that \(\text{sign}(w^T x) \rightarrow \text{pos}\) and \(\text{sign}(w^T x) \rightarrow \text{neg}\)
Lines in $\mathbb{R}^2$

$y = -\frac{a}{c} x$
$\iff ax + cy = 0$
$\iff [a \ c] \begin{bmatrix} x \\ y \end{bmatrix} = 0$
$\iff \mathbf{w}^T \mathbf{x} = 0$ for $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$, $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

Set of points $\mathbf{x}$ that are $\mathbf{w}^T \mathbf{x} = 0$
⇒ Set of points $(x, y)$ such that their vector forms $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$ are perpendicular to $\mathbf{w}$
⇒ Those points form a line.
⇒ Define a line $(y = -\frac{a}{c} x)$ with $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$
**Lines in $\mathbb{R}^2$**

\[
y = -\frac{a}{c} x - \frac{b}{c} \\
\iff ax + cy + b = 0 \\
\iff [a \ c] [x \ y] + b = 0 \\
\iff \mathbf{w}^T \mathbf{x} + b = 0 \text{ for } \mathbf{w} = [a, c], \mathbf{x} = [x, y]
\]

Set of points $\mathbf{x}$ that are $\mathbf{w}^T \mathbf{x} + b = 0$

$\Rightarrow$ Define a line $(y = -\frac{a}{c} x - \frac{b}{c})$ with $\mathbf{w} = [a, c]$ and $b$
Distance from P to line

\[ P = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \]

\[ D \]

\[ Q = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \]

\[ w = \begin{bmatrix} a \\ c \end{bmatrix} \]

\[ \overrightarrow{QP} \text{= vector from Q to } P = P - Q = \begin{bmatrix} x_0 - x_1 \\ y_0 - y_1 \end{bmatrix} \]

\[ w = \begin{bmatrix} a \\ c \end{bmatrix}, \quad \|w\| = \sqrt{a^2 + c^2} \]

\[ D = \text{Length of projecting } \overrightarrow{QP} \text{ on } w \]

\[ D = \frac{\|\overrightarrow{QP} \cdot w\|}{\|w\|} \]

\[ \overrightarrow{QP} \cdot w = \left( \begin{bmatrix} x_0 - x_1 \\ y_0 - y_1 \end{bmatrix} \right) \cdot \begin{bmatrix} a \\ c \end{bmatrix} = \begin{vmatrix} a(x_0 - x_1) + c(y_0 - y_1) \end{vmatrix} \]

\[ \|w\| = \sqrt{a^2 + c^2} \]

\[ D = \frac{\sqrt{a^2 + c^2}}{\sqrt{a^2 + c^2}} \]

Since Q is on the line, \( ax_1 + cy_1 + b = 0 \)

\[ D = \frac{|ax_0 + cy_0 + b|}{\sqrt{a^2 + c^2}} \]
Distance from a point to line

Given: Line defined by $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ and $b$

For a new point $\mathbf{x} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$, what is its distance to the line?

$$D = \frac{|ax_0 + cy_0 + b|}{\sqrt{a^2 + c^2}} = \frac{||\mathbf{w}^T\mathbf{x} + b||}{||\mathbf{w}||}$$
Given: Line defined by $\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix}$ and $b$
For a new point $\mathbf{x}$, which side is the point on with respect to the line?

$\mathbf{w}^T \mathbf{x} + b > 0 \Rightarrow \mathbf{x}$ on one side
$\mathbf{w}^T \mathbf{x} + b < 0 \Rightarrow \mathbf{x}$ on the other side
$\mathbf{w}^T \mathbf{x} + b = 0 \Rightarrow \mathbf{x}$ on the line
Linear classifiers

- Find linear function to separate positive and negative examples

Find \( \mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix} \) and \( b \) such that

\[
\mathbf{w}^T \mathbf{x}_i + b > 0 \Rightarrow \mathbf{x}_i \text{ is positive}
\]

\[
\mathbf{w}^T \mathbf{x}_i + b < 0 \Rightarrow \mathbf{x}_i \text{ is negative}
\]

Which line is best?

Support vector machines

- Discriminative classifier based on optimal separating line (for 2d case)
- Maximize the margin between the positive and negative training examples
Support vector machines

- Want line that maximizes the margin.

$$\mathbf{w}^T \mathbf{x}_i + b \geq 1 \Rightarrow \mathbf{x}_i \text{ is positive } (y_i = +1)$$

$$\mathbf{w}^T \mathbf{x}_i + b \leq -1 \Rightarrow \mathbf{x}_i \text{ is negative } (y_i = -1)$$

$$\mathbf{x}_i \text{ on the dashed lines are support vectors: } \mathbf{w}^T \mathbf{x}_i + b = \pm 1 = +1 \text{ or } -1$$

To compute the margin $M$:
Distance from line to pos support vector $\mathbf{x}_i$
Distance from line to neg support vector $\mathbf{x}_i$

$$M = \frac{|\mathbf{w}^T \mathbf{x}_i + b|}{||\mathbf{w}||} + \frac{|\mathbf{w}^T \mathbf{x}_i + b|}{||\mathbf{w}||}$$

$$= \frac{1}{||\mathbf{w}||} + \frac{1}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$$

Margin is $M = \frac{2}{||\mathbf{w}||}$
Support vector machines

- Want line that maximizes the margin.

\[ \mathbf{w}^T \mathbf{x} + b = -1 \]
\[ \mathbf{w}^T \mathbf{x} + b = +1 \]
\[ \mathbf{w}^T \mathbf{x} + b = 0 \]

SVM: Solve for \( \mathbf{w} \) such that

1. \( \mathbf{w}^T \mathbf{x}_i + b \geq 1 \Rightarrow \mathbf{x}_i \) is positive \( (y_i = +1) \)
2. \( \mathbf{w}^T \mathbf{x}_i + b \leq -1 \Rightarrow \mathbf{x}_i \) is negative \( (y_i = -1) \)
3. The margin \( M = \frac{2}{\|\mathbf{w}\|} \) is maximized

Support vector machines

- Want line that **maximizes the margin**.

**SVM**: Solve for \( \mathbf{w} \) such that

1. \( \mathbf{w}^T \mathbf{x}_i + b \geq 1 \Rightarrow \mathbf{x}_i \) is positive (\( y_i = +1 \))
2. \( \mathbf{w}^T \mathbf{x}_i + b \leq -1 \Rightarrow \mathbf{x}_i \) is negative (\( y_i = -1 \))
3. The margin \( M = \frac{2}{\|\mathbf{w}\|} \) is maximized

Support vector machines

- Want line that **maximizes the margin**.

**SVM:** Solve for \( \mathbf{w} \) such that

1. \( \mathbf{w}^T \mathbf{x}_i + b \geq 1 \Rightarrow \mathbf{x}_i \) is positive \((y_i = +1)\)
2. \( \mathbf{w}^T \mathbf{x}_i + b \leq -1 \Rightarrow \mathbf{x}_i \) is negative \((y_i = -1)\)
3. The margin \( M = \frac{2}{\|\mathbf{w}\|} \) is maximized

\[ w \] with \( \|w\| = 5 \]
Support vector machines

- Want line that **maximizes the margin**.

**SVM:** Solve for \( \mathbf{w} \) such that

1. \( \mathbf{w}^T \mathbf{x}_i + b \geq 1 \Rightarrow \mathbf{x}_i \) is positive \( (y_i = +1) \)
2. \( \mathbf{w}^T \mathbf{x}_i + b \leq -1 \Rightarrow \mathbf{x}_i \) is negative \( (y_i = -1) \)
3. The margin \( M = \frac{2}{\|\mathbf{w}\|} \) is maximized

\[ \mathbf{w} \text{ with } \|\mathbf{w}\| = 0.5 \]
Support vector machines

- Want line that maximizes the margin.

\[ \text{SVM: Solve for } \mathbf{w} \text{ such that} \]

1. \( \mathbf{w}^T \mathbf{x}_i + b \geq 1 \Rightarrow \mathbf{x}_i \text{ is positive } (y_i = +1) \)
2. \( \mathbf{w}^T \mathbf{x}_i + b \leq -1 \Rightarrow \mathbf{x}_i \text{ is negative } (y_i = -1) \)
3. The margin \( M = \frac{2}{\|\mathbf{w}\|} \) is maximized
   1. \( \|\mathbf{w}\| \) is minimized

Note that \( M = \frac{2}{\|\mathbf{w}\|} \) is inversely proportional to \( \|\mathbf{w}\| \)
Support vector machines

SVM: Solve for $\mathbf{w}$ such that

1. $\mathbf{w}^T \mathbf{x}_i + b \geq 1 \Rightarrow \mathbf{x}_i$ is positive ($y_i = +1$)
2. $\mathbf{w}^T \mathbf{x}_i + b \leq -1 \Rightarrow \mathbf{x}_i$ is negative ($y_i = -1$)
3. $\|\mathbf{w}\|$ is minimized

Further simplification of the condition 1 and 2:

$\mathbf{w}^T \mathbf{x}_i + b \geq +1$ for $\mathbf{x}_i$ positive ($y_i = +1$)
$\Rightarrow y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \cdot y_i$
$\Rightarrow y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

$\mathbf{w}^T \mathbf{x}_i + b \leq -1$ for $\mathbf{x}_i$ negative ($y_i = -1$)
$\Rightarrow y_i (\mathbf{w}^T \mathbf{x}_i + b) \leq -1 \cdot y_i = -1 \cdot -1$
$\Rightarrow y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

For both $\mathbf{x}_i$ positive ($y_i = +1$) and $\mathbf{x}_i$ negative ($y_i = -1$),
$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$
Support vector machines

SVM: Solve for $\mathbf{w}$ such that
1. $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$
2. $\|\mathbf{w}\|$ is minimized

Further simplification of the condition 1 and 2:

$\mathbf{w}^T \mathbf{x}_i + b \geq +1$ for $\mathbf{x}_i$ positive ($y_i = +1$)
$\Rightarrow y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \cdot y_i$
$\Rightarrow y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

$\mathbf{w}^T \mathbf{x}_i + b \leq -1$ for $\mathbf{x}_i$ negative ($y_i = -1$)
$\Rightarrow y_i(\mathbf{w}^T \mathbf{x}_i + b) \leq -1 \cdot y_i = -1 \cdot -1$
$\Rightarrow y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

For both $\mathbf{x}_i$ positive ($y_i = +1$) and $\mathbf{x}_i$ negative ($y_i = -1$),
$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998
Finding the maximum margin line

SVM: Solve for \( w \) such that

1. \( y_i(w^T x_i + b) \geq 1 \)
2. \( ||w|| \) is minimized

\[
\text{minimize } ||w|| \propto ||w||^2 = w^T w
\]

Quadratic optimization problem:
Find \( w \) such that for all examples \((x_1, y_1), \ldots, (x_N, y_N)\)

\[
\text{minimize } \frac{1}{2} w^T w
\]
subject to \( y_i(w^T x_i + b) \geq 1 \)

Once we find such \( w \) (and \( b \)), we can predict new \( x \):

\[ f(x) = \text{sign}(w^T x + b) \]
Finding the maximum margin line

Interesting property 1:
The solution $\mathbf{w}$ can be computed using the given dataset:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

For $\alpha_i \geq 0$.

In other words, we don’t have to blindly find $\mathbf{w}$ from scratch. Instead, we can find $\alpha_i$ such that $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$ is the solution.

Interesting property 2:

$\alpha_i > 0$ for support vectors $\mathbf{x}_i$

$\alpha_i = 0$ for all other points

Thus, we assume $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$ only consists of support vectors

New form of SVM

\[ w = \sum \alpha_i y_i x_i \]

Where \( \alpha_i > 0 \) for the support vectors \( x_i \).
Computing \( w \) gives us \( b = y_i - w^T x_i \).

Our SVM is a function with \( w \) (and \( b \)) such that we can predict new \( x \):

\[
  f(x_i) = \text{sign}(w^T x + b) = \text{sign} \left( \sum \alpha_i y_i x_i^T x + b \right)
\]
New form of SVM

New form of SVM entirely consists of $\alpha_i$ and support vectors $x_i$

$$f(x) = sign(w^T x + b)$$

$$= sign \left( \sum_i \alpha_i y_i x_i^T x + b \right)$$

No need to explicitly solve for $w$.

Instead, we have many inner products (dot products) between the given point $x$ and the support vectors $x_i$
New form of SVM

$f(x) = \text{sign} \left( \sum_i \alpha_i y_i x_i^T x + b \right)$

What does it mean intuitively:

1. Remember: Each support vector $x_i$ is associated with its label $y_i$
2. Inner product between $x_i^T$ and $x$ is
   1. large if similar (close)
   2. small if not similar (far)
3. We compare the new point $x$ to the support vectors (compute inner prod).
4. $f(x)$ above basically checks how close $x$ is to the support vectors and decide on its label:
   $f(x) = \text{sign} \left( \sum_i \alpha_i y_i x_i^T x + b \right)$
Nonlinear SVMs

• Datasets that are linearly separable work out great:

- But what if the dataset is just too hard?

- We can map it to a higher-dimensional space:
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 kernel: Example

• Consider the mapping \( \varphi(x) = (x, x^2) \)

\[
\varphi(x) \cdot \varphi(y) = (x, x^2) \cdot (y, y^2) = xy + x^2 y^2
\]

\[
K(x, y) = xy + x^2 y^2
\]
Finding the maximum margin line

Primal problem of SVM

\[
\text{minimize}_w \frac{1}{2} w^T w \\
\text{subject to } y_i (w^T x_i + b) \geq 1
\]

Feature mapping of \( x_i \rightarrow \phi(x_i) \)

\[
\text{minimize}_w \frac{1}{2} w^T w \\
\text{subject to } y_i (w^T \phi(x_i) + b) \geq 1
\]

Dual problem of SVM

\[
\text{maximize}_\alpha \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j \\
\text{subject to } \alpha_i \geq 0 \\
\sum_{i=1}^n \alpha_i y_i = 0
\]

Feature mapping of \( x_i \rightarrow \phi(x_i) \)

\[
\text{maximize}_\alpha \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) \\
\text{subject to } \alpha_i \geq 0 \\
\sum_{i=1}^n \alpha_i y_i = 0
\]

Can be precomputed

Finding the best \( w \) in primal is equivalent to finding the best \( \alpha_i \) in dual
The “Kernel Trick”

• The linear classifier relies on dot product between vectors $K(x_i, x_j) = x_i \cdot x_j$

• If every data point is mapped into high-dimensional space via some transformation $\Phi: x_i \rightarrow \varphi(x_i)$, the dot product becomes: $K(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j)$

• A kernel function is similarity function that corresponds to an inner product in some expanded feature space

• The kernel trick: instead of explicitly computing the lifting transformation $\varphi(x)$, define a kernel function $K$ such that: $K(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j)$
Examples of kernel functions

- Linear: \( K(x_i, x_j) = x_i^T x_j \)

- Polynomials of degree up to \( d \):
  \[
  K(x_i, x_j) = (x_i^T x_j + 1)^d
  \]

- Gaussian RBF:
  \[
  K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)
  \]
Polynomials feature map of degree \( d=2 \):

For 2 dimensional input:
\[ \mathbf{x} = [x_1, x_2], \quad \mathbf{y} = [y_1, y_2] \]

Feature map:
\[
\begin{align*}
\phi(\mathbf{x}) &= [x_2^2, x_1^2, \sqrt{2}x_2x_1, \sqrt{2}x_2, \sqrt{2}x_1, 1] \\
\phi(\mathbf{y}) &= [y_2^2, y_1^2, \sqrt{2}y_2y_1, \sqrt{2}y_2, \sqrt{2}y_1, 1]
\end{align*}
\]

Inner product:
\[
\phi(\mathbf{x})^T \phi(\mathbf{y}) = [x_2^2, x_1^2, \sqrt{2}x_2x_1, \sqrt{2}x_2, \sqrt{2}x_1, 1] \cdot [y_2^2, y_1^2, \sqrt{2}y_2y_1, \sqrt{2}y_2, \sqrt{2}y_1, 1]
\]

But with kernel:
\[
K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^2 = (x_1y_1 + x_2y_2 + 1)^2 = \phi(\mathbf{x})^T \phi(\mathbf{y})
\]

For n dimensional input:
\[ \mathbf{x} = [x_1, ..., x_n] \]

Feature map size is \( \frac{n(n-1)}{2} + 2n + 1 \)
\[
\phi(\mathbf{x}) = [x_n^2, ..., x_1^2, \sqrt{2}x_n x_{n-1}, ..., \sqrt{2}x_n x_1, \sqrt{2}x_{n-1} x_{n-2}, ..., \sqrt{2}x_1 x_1, \sqrt{2}x_n, ..., \sqrt{2}x_1, 1]
\]
RBF kernel

- Radial basis function (RBF) kernel

\[ K(x, x') = \exp \left( -\frac{||x-x'||^2}{2\sigma^2} \right) \]

- The feature map has an infinite number of dimensions.

\[
\exp \left( -\frac{1}{2}||x - x'||^2 \right) = \exp \left( \frac{2}{2} x^\top x' - \frac{1}{2}||x||^2 - \frac{1}{2}||x'||^2 \right) \\
= \exp(x^\top x') \exp \left( -\frac{1}{2}||x||^2 \right) \exp \left( -\frac{1}{2}||x'||^2 \right) \\
= \sum_{j=0}^{\infty} \frac{(x^\top x')^j}{j!} \exp \left( -\frac{1}{2}||x||^2 \right) \exp \left( -\frac{1}{2}||x'||^2 \right) \\
= \sum_{j=0}^{\infty} \left( \sum_{\sum n_i = j} \exp \left( -\frac{1}{2}||x||^2 \right) \frac{x_1^{n_1} \cdots x_k^{n_k}}{\sqrt{n_1! \cdots n_k!}} \exp \left( -\frac{1}{2}||x'||^2 \right) \frac{x'_1^{n_1} \cdots x'_k^{n_k}}{\sqrt{n_1! \cdots n_k!}} \right) \\

\]
Data is not always nice

minimize $\frac{1}{2} \mathbf{w}^T \mathbf{w}$
subject to $y_i \mathbf{w}^T \mathbf{x}_i \geq 1$

Where is $b$?
Now a part of $\mathbf{w}$
$\mathbf{w} = [w_1, w_2, b]$
$\mathbf{x} = [x_1, x_2, 1]$
Hard-margin SVM

Never allow misclassification!

\[
\text{minimize } \frac{1}{2} \mathbf{w}^T \mathbf{w}
\]

subject to \( y_i \mathbf{w}^T \mathbf{x}_i \geq 1 \)

Where is \( b \)?

Now a part of \( \mathbf{w} \)

\( \mathbf{w} = [w_1, w_2, b] \)

\( \mathbf{x} = [x_1, x_2, 1] \)
Soft-margin SVM

Allow misclassification!

\[ y_i w^T x_i < 1 \]
When \( 1 > \xi_i > 0 \)

\[ y_i w^T x_i > 0 \]
When \( \xi_i = 0 \)

\[ y_i w^T x_i < 0 \]
When \( \xi_i > 1 \)

minimize \[ w \]
\[ \begin{array}{c}
\frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i \\
\end{array} \]
subject to \[ y_i w^T x_i \geq 1 - \xi_i \]
\[ \xi_i \geq 0 \]

\( C \): misclassification cost to control how much misclassification we want to allow. 
- Small \( C \): Allow large \( \xi_i \)
- Large \( C \): Do not allow large \( \xi_i \)

\( \xi_i \): Slack variable to allow some points to be misclassified
SVM Demo

- https://jgreitemann.github.io/svm-demo
- https://cs.stanford.edu/~karpathy/svmjs/demo/
What about multi-class SVMs?

• Unfortunately, there is no “definitive” multi-class SVM formulation

• In practice, we have to obtain a multi-class SVM by combining multiple two-class SVMs

• One vs. others
  – Training: learn an SVM for each class vs. the others
  – Testing: apply each SVM to the test example, and assign it to the class of the SVM that returns the highest decision value

• One vs. one
  – Training: learn an SVM for each pair of classes
  – Testing: each learned SVM “votes” for a class to assign to the test example
Multi-class problems

• One-vs-all (a.k.a. one-vs-others)
  – Train K classifiers
  – In each, pos = data from class \( i \), neg = data from classes other than \( i \)
  – The class with the most confident prediction wins
  – Example:
    • You have 4 classes, train 4 classifiers
    • 1 vs others: score 3.5
    • 2 vs others: score 6.2
    • 3 vs others: score 1.4
    • 4 vs other: score 5.5
    • Final prediction: class 2
Multi-class problems

• One-vs-one (a.k.a. all-vs-all)
  – Train $K(K-1)/2$ binary classifiers (all pairs of classes)
  – They all vote for the label
  – Example:
    • You have 4 classes, then train 6 classifiers
    • 1 vs 2, 1 vs 3, 1 vs 4, 2 vs 3, 2 vs 4, 3 vs 4
    • Votes: 1, 1, 4, 2, 4, 4
    • Final prediction is class 4
Using SVMs

1. Select a kernel function.

2. Compute pairwise kernel values between labeled examples.

3. Use this “kernel matrix” to solve for SVM support vectors & alpha weights.

4. To classify a new example: compute kernel values between new input and support vectors, apply alpha weights, check sign of output.

Adapted from Kristen Grauman
Some SVM packages

- LIBLINEAR [https://www.csie.ntu.edu.tw/~cjlin/liblinear/](https://www.csie.ntu.edu.tw/~cjlin/liblinear/)
Linear classifiers vs nearest neighbors

- **Linear pros:**
  - Low-dimensional *parametric* representation
  - Very fast at test time
- **Linear cons:**
  - Can be tricky to select best kernel function for a problem
  - Learning can take a very long time for large-scale problem
- **NN pros:**
  - Works for any number of classes
  - Decision boundaries not necessarily linear
  - *Nonparametric* method
  - Simple to implement
- **NN cons:**
  - Slow at test time (large search problem to find neighbors)
  - Storage of data
  - Especially need good distance function (but true for all classifiers)

Adapted from L. Lazebnik
Training vs Testing

• What do we want?
  – High accuracy on training data?
  – No, high accuracy on unseen/new/test data!
  – Why is this tricky?

• Training data
  – Features (x) and labels (y) used to learn mapping f

• Test data
  – Features (x) used to make a prediction
  – Labels (y) only used to see how well we’ve learned f!!!

• Validation data
  – Held-out set of the training data
  – Can use both features (x) and labels (y) to tune parameters of the model we’re learning
Generalization

• How well does a learned model generalize from the data it was trained on to a new test set?
Generalization

• Components of generalization error
  – **Noise** in our observations: unavoidable
  – **Bias**: due to inaccurate assumptions/simplifications by model
  – **Variance**: models estimated from different training sets differ greatly from each other

• **Underfitting**: model is too “simple” to represent all the relevant class characteristics
  – High bias and low variance
  – High training error and high test error

• **Overfitting**: model is too “complex” and fits irrelevant characteristics (noise) in the data
  – Low bias and high variance
  – Low training error and high test error
Generalization

Dataset subset 1

Dataset subset 2

Dataset subset 3

Dataset subset 4

Dataset subset 5

Model 1

Model 2

Model 3

Model 4

Model 5

Average model

True model?

Variance: How different is the model given different examples?

Bias: How off is your model from the true model?
The model is consistent, but it is too simple. e.g.: mean color of images

The model is making consistently decent training predictions. e.g.: filter banks, BoW

The model is making good training predictions, but it will do poorly on new examples. e.g.: entire pixel intensities
Generalization

• Models with too few parameters are inaccurate because of a large bias (not enough flexibility).

• Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).

Red dots = training data (all that we see before we ship off our model!)

Green curve = true underlying model

Blue curve = our predicted model/fit

Purple dots = possible test points

Adapted from D. Hoiem
Training vs test error

Underfitting

Overfitting

Error

Complexity

High Bias
Low Variance

Low Bias
High Variance

Slide credit: D. Hoiem
The effect of training set size

Complexity

Test Error

High Bias
Low Variance

Low Bias
High Variance

Slide credit: D. Hoiem
Choosing the trade-off between bias and variance

- Need validation set (separate from the test set)

---

Slide credit: D. Hoiem
Which line is train/val/test?

- **Early stopping**: Stop training before the model overfits to the training data (low train loss, high test loss)
- **Ex**: train 80%, val 10%, test 10%
- As you train, check val loss (assuming val ~ test)
- Best model is the one trained up to the point of best validation
- Less important when
  - Data is extremely large such that train ~ val ~ test (low train loss => low val and test loss)
Generalization tips

• Try simple classifiers first
• Better to have smart features and simple classifiers than simple features and smart classifiers
• Use increasingly powerful classifiers with more training data
• As an additional technique for reducing variance, try regularizing the parameters (penalize high magnitude weights)
KNN and SVM

• Bias and Variance in KNN and SVM?

KNN

• https://lecture-demo.ira.uka.de/knn-demo/

SVM

• https://jgreitemann.github.io/svm-demo
Classification: Example of Training and Testing

Dataset: Scene Classification Dataset

1. What is your data?
   - Images of scenes. Each is of size 256 by 256 (resized and grayscale).
   - I also have true labels

2. How many examples? How many classes?
   - 8 classes. 150 images per class. Total of 1200 images.

3. What is your training and testing set?
   - For each class, randomly pick 100 images for training and 50 for testing. Now I have 800 training images (8 classes x 100 images), and 400 test images (8 classes x 50 images)
Classification: Example of Training and Testing

Training:
1. What is the feature representation of each image?
   1. Examples: Bag-of-Words representations of SIFT features
2. What classifier?
   1. KNN with some k
   2. SVM with linear kernel
3. Do I have what I need?: X (feature), Y (labels), Training Set, Test Set, Classifier of my choice
4. Train using Training set
   1. Input: feature representations of the training images and their labels
   2. Train of your classifier:
      1. KNN requires no training.
      2. SVM requires training: model = train_svm(train_set_features, train_set_labels)
3. How well does your model work on the training set?
   1. Train_set_prediction = KNN(train_set_features, train_set_labels, train_set_features)
   2. Train_set_prediction = SVM(model, train_set_features)
4. Train set performance
   1. Train_accuracy = sum(Train_set_prediction == train_set_labels) / num_examples

Examples to predict training examples and labels needed for KNN prediction
Classification: Example of Training and Testing

Testing:

1. Test using **Test set**
   1. Input: **feature representations** of the test images and their labels
   2. How well does your model work on the **test set**?
      1. Test_set_prediction = KNN(train_set_features, train_set_labels, test_set_features)
      2. Test_set_prediction = SVM(model, test_set_features)

3. Test set performance
   1. Test_accuracy = sum(Test_set_prediction == test_set_labels) / num_examples

4. Analyze
   1. Did my model overfit?: Train_accuracy vs. Test_accuracy
   2. Can I improve my model?: Try different model-related parameters
      1. K for KNN
      2. Kernel (linear, RBF) or slack variable for SVM
Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories

• CVPR 2006

• Svetlana Lazebnik (slazebni@uiuc.edu)
  • Beckman Institute, University of Illinois at Urbana-Champaign

• Cordelia Schmid (cordelia.schmid@inrialpes.fr)
  • INRIA Rhône-Alpes, France

• Jean Ponce (ponce@di.ens.fr)
  • Ecole Normale Supérieure, France

Winner of 2016 Longuet-Higgins Prize
Scene category dataset

Fei-Fei & Perona (2005), Oliva & Torralba (2001)

http://www-cvr.ai.uiuc.edu/ponce_grp/data

Slide credit: L. Lazebnik
Bag-of-words representation

1. Extract local features
2. Learn “visual vocabulary” using clustering
3. Quantize local features (construct histogram) using visual vocabulary as cluster centers (means)
4. Represent images by frequencies of “visual words”
Image categorization with bag of words

Training
1. Extract keypoints/descriptors for train images
2. Compute bag-of-words (BoW) representation using the cluster centers (aka means or visual words)
3. Train classifier on labeled examples using BoW representation as features
4. Labels are the scene types (e.g. mountain vs field)

Testing
1. Extract keypoints/descriptors for test images
2. Compute bag-of-words (BoW) representation using the cluster centers also used in training
3. Compute labels on test images using classifier obtained at training time
4. Evaluation: Measure accuracy of test predictions by comparing them to ground-truth test labels (obtained from humans or already provided in the original dataset)

Adapted from D. Hoiem
Feature extraction (on which BOW is based)

**Weak features**

Edge points at 2 scales and 8 orientations (vocabulary size 16)

**Strong features**

SIFT descriptors of 16x16 patches sampled on a regular grid, quantized to form visual vocabulary (size 200, 400)

Slide credit: L. Lazebnik
What about spatial layout?

All of these images have the same color histogram
Spatial pyramid

Compute histogram in each spatial bin
Spatial pyramid

- Level 0
- Level 1
- Level 2

[LaZebnik et al. CVPR 2006]
Same color histogram, same Spatial Pyramid?

Level-0 SPM: Same or different?
Level-1 SPM: Same or different?
Scene category dataset
Fei-Fei & Perona (2005), Oliva & Torralba (2001)
http://www-cvr.ai.uiuc.edu/ponce_grp/data

Multi-class classification results (100 training images per class)

<table>
<thead>
<tr>
<th>Level</th>
<th>Weak features (vocabulary size: 16)</th>
<th>Strong features (vocabulary size: 200)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single-level</td>
<td>Pyramid</td>
</tr>
<tr>
<td>0 (1 × 1)</td>
<td>45.3 ±0.5</td>
<td>56.2 ±0.6</td>
</tr>
<tr>
<td>1 (2 × 2)</td>
<td>53.6 ±0.3</td>
<td>64.7 ±0.7</td>
</tr>
<tr>
<td>2 (4 × 4)</td>
<td>61.7 ±0.6</td>
<td>66.8 ±0.6</td>
</tr>
<tr>
<td>3 (8 × 8)</td>
<td>63.3 ±0.8</td>
<td></td>
</tr>
</tbody>
</table>

Fei-Fei & Perona: 65.2%

Standard deviation of 10 runs, random train/test split
Scene category confusions

row $i$, col $j$: % of object $i$ classified as object $j$

Difficult indoor images

kitchen
living room
bedroom

Slide credit: L. Lazebnik
Multi-class classification results (30 training images and up to 50 test images per class)

<table>
<thead>
<tr>
<th>Level</th>
<th>Weak features (16)</th>
<th>Strong features (200)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single-level</td>
<td>Pyramid</td>
</tr>
<tr>
<td>0</td>
<td>15.5 ±0.9</td>
<td>41.2 ±1.2</td>
</tr>
<tr>
<td>1</td>
<td>31.4 ±1.2</td>
<td>32.8 ±1.3</td>
</tr>
<tr>
<td>2</td>
<td>47.2 ±1.1</td>
<td>49.3 ±1.4</td>
</tr>
<tr>
<td>3</td>
<td>52.2 ±0.8</td>
<td>54.0 ±1.1</td>
</tr>
</tbody>
</table>

Caltech101 dataset
Fei-Fei et al. (2004)