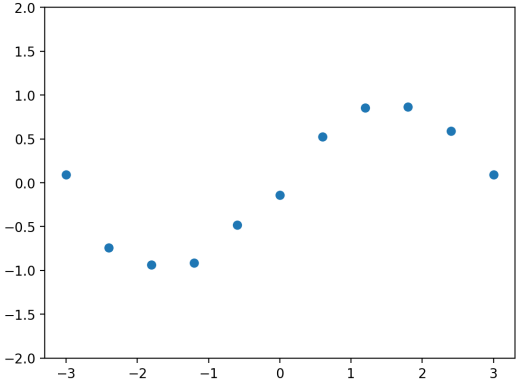
Machine Learning: Representation and Kernels

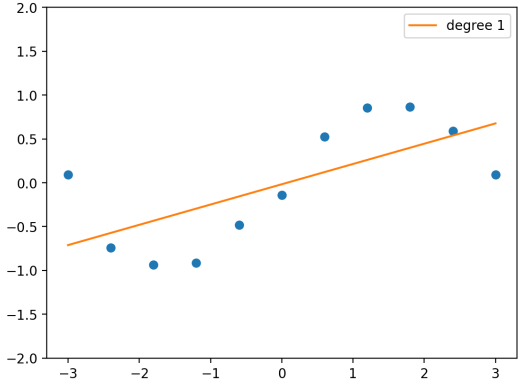
Hao Tang

February 5, 2024

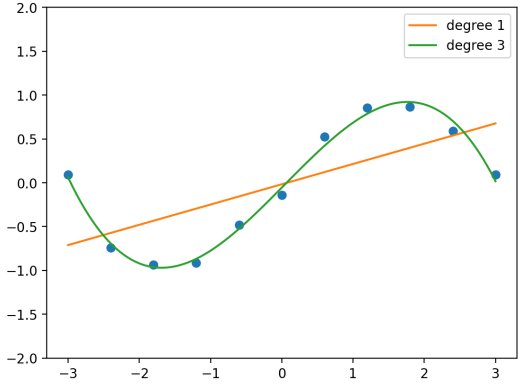
Polynomial regression



Polynomial regression



Polynomial regression



Data

x	y
-3.0	0.0927
-2.4	-0.7417
-1.8	-0.9344
-1.2	-0.9174
-0.6	-0.4811
0.0	-0.1402
⋮	⋮

Features

- Instead of writing $y = w^\top x + b$, we can add another dimension that is always 1 and have

$$y = w^\top x + b = \begin{bmatrix} w \\ b \end{bmatrix}^\top \begin{bmatrix} x \\ 1 \end{bmatrix} = \tilde{w}^\top \tilde{x}. \tag{1}$$

Data

	x	y
1	-3.0	0.0927
1	-2.4	-0.7417
1	-1.8	-0.9344
1	-1.2	-0.9174
1	-0.6	-0.4811
1	0.0	-0.1402
⋮	⋮	⋮

Features

- What happens if we add a dimension of x^2?

Features

- What happens if we add a dimension of x^2?

$$y = \begin{bmatrix} w_2 \\ w_1 \\ w_0 \end{bmatrix}^\top \begin{bmatrix} x^2 \\ x \\ 1 \end{bmatrix} = w_2 x^2 + w_1 x + w_0 \tag{2}$$

Features

- What happens if we add a dimension of x^2?

$$y = \begin{bmatrix} w_2 \\ w_1 \\ w_0 \end{bmatrix}^\top \begin{bmatrix} x^2 \\ x \\ 1 \end{bmatrix} = w_2 x^2 + w_1 x + w_0 \tag{2}$$

- If we add x^2 to the data, we can now fit a parabola.

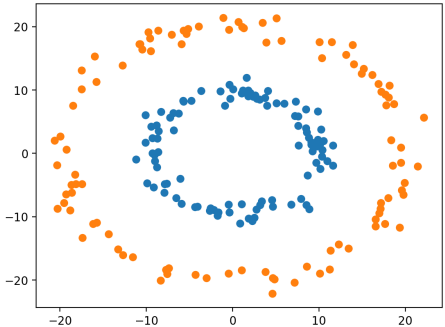
Data

	x	x^2	y
1	-3.0	9.0	0.0927
1	-2.4	5.76	-0.7417
1	-1.8	3.24	-0.9344
1	-1.2	1.44	-0.9174
1	-0.6	0.36	-0.4811
1	0.0	0.0	-0.1402
⋮	⋮	⋮	⋮

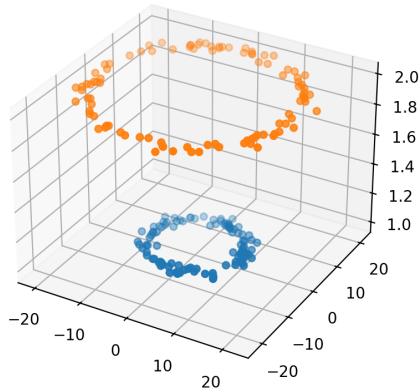
Features

- In general, we can add arbitrary high-degree terms.
- If we add degree-2 terms to $\begin{bmatrix} 1 & x & y \end{bmatrix}$, we get $\begin{bmatrix} 1 & x & y & x^2 & y^2 & xy \end{bmatrix}$.
- If we add degree-2 terms to $\begin{bmatrix} 1 & x & y & z \end{bmatrix}$, we get $\begin{bmatrix} 1 & x & y & z & x^2 & y^2 & z^2 & xy & xz & yz \end{bmatrix}$.
- The combination becomes many if we have more dimensions.

Features



Features



Features

- Polynomial features are generic (but often useful).
- Featuers don't need to be generic.
- Consider face detection. How do you describe what a face is?

Face detection

Face detection

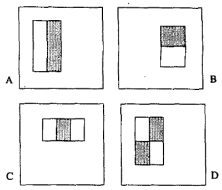


Figure 1: Example rectangle features shown relative to the enclosing detection window. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the grey rectangles. Two-rectangle features are shown in (A) and (B). Figure (C) shows a three-rectangle feature, and (D) a four-rectangle feature.

Face detection

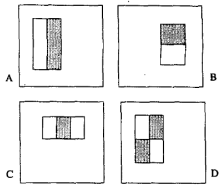


Figure 1: Example rectangle features shown relative to the enclosing detection window. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the grey rectangles. Two-rectangle features are shown in (A) and (B). Figure (C) shows a three-rectangle feature, and (D) a four-rectangle feature.

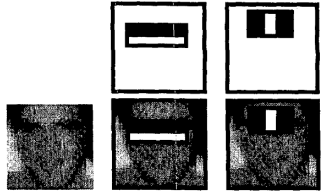


Figure 4: The first and second features selected by AdaBoost. The two features are shown in the top row and then overlayed on a typical training face in the bottom row. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalizes on the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose.

Face detection

Rapid Object Detection using a Boosted Cascade of Simple Features

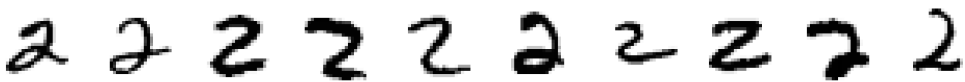
Paul Viola
viola@merl.com
Mitsubishi Electric Research Labs
201 Broadway, 8th FL
Cambridge, MA 02139

Michael Jones
michael.jones@compaq.com
Compaq Cambridge Research Lab
One Cambridge Center
Cambridge, MA 02142

- Viola-Jones end up using about 160,000 features.
- It is of course not as good as the modern neural networks, but it is robust and fast.
- The approach also later inspires how deep neural networks are designed.

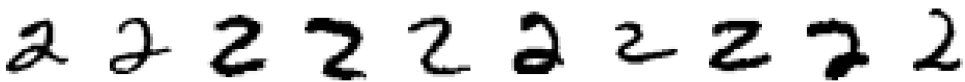
Digit recognition

- Consider digit recognition. How do you describe the digit two?



Digit recognition

- Consider digit recognition. How do you describe the digit two?



- It is a two if it is similar to one of those twos (sometimes called examplars).
- A feature can be how similar the sample is to one of those examplars.
- If the above examplars are $x_1, x_2, \ldots, x_{10}$,

$$\begin{bmatrix} 1 & x^\top x_1 & x^\top x_2 & \ldots & x^\top x_{10} \end{bmatrix} \tag{3}$$

Two approaches

- Feature
  - A feature describes something about the input.
  - The feature vector of x is written as $\phi(x)$.
  - We do $f(x) = w^\top \phi(x)$ to make a prediction.
- Kernel
  - A kernel describes similarities of the input to other samples.
  - The similarity of two samples x and x' is written as $k(x, x')$.
  - We do $f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i)$ to make a prediction.

Kernels and features

- A kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is defined as

$$k(x, x') = \phi(x)^\top \phi(x') \tag{4}$$

  for some feature function ϕ.

- We can immediately see that k is symmetric, i.e., $k(x, x') = k(x', x)$.

- We allow ϕ to return an infinite-dimensional vector, and want to avoid computing $\phi(x)^\top \phi(x')$.

Going from features to kernels

- The mean-squared error can be written as

$$L = \|\Phi w - y\|_2^2 \tag{5}$$

where

$$\Phi = \begin{bmatrix} — \ \phi(x_1) \ — \\ — \ \phi(x_2) \ — \\ \vdots \\ — \ \phi(x_n) \ — \end{bmatrix} \tag{6}$$

- The optimal solution is $w^* = (\Phi^\top \Phi)^{-1} \Phi^\top y$.

Going from features to kernels

- To make a prediction,

$$f(x) = w^\top \phi(x) = \left((\Phi^\top \Phi)^{-1} \Phi^\top y\right)^\top \phi(x) \tag{7}$$

$$= y^\top \Phi (\Phi^\top \Phi)^{-1} \phi(x) \tag{8}$$

- Note that $A(A^\top A)^{-1} = (AA^\top)^{-1} A$.

$$f(x) = y^\top \Phi (\Phi^\top \Phi)^{-1} \phi(x) = y^\top (\Phi \Phi^\top)^{-1} \Phi \phi(x) \tag{9}$$

Going from features to kernels

$$f(x) = y^\top (\Phi\Phi^\top)^{-1}\Phi\phi(x) \tag{10}$$

$$= y^\top \begin{bmatrix} \phi(x_1)^\top\phi(x_1) & \phi(x_1)^\top\phi(x_2) & \dots & \phi(x_1)^\top\phi(x_n) \\ \phi(x_2)^\top\phi(x_1) & \phi(x_2)^\top\phi(x_2) & \dots & \phi(x_2)^\top\phi(x_n) \\ \vdots & \vdots & & \vdots \\ \phi(x_n)^\top\phi(x_1) & \phi(x_n)^\top\phi(x_2) & \dots & \phi(x_n)^\top\phi(x_n) \end{bmatrix}^{-1} \begin{bmatrix} \phi(x_1)^\top\phi(x) \\ \phi(x_2)^\top\phi(x) \\ \vdots \\ \phi(x_n)^\top\phi(x) \end{bmatrix} \tag{11}$$

$$= y^\top \begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & \dots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & \dots & k(x_2,x_n) \\ \vdots & \vdots & & \vdots \\ k(x_n,x_1) & k(x_n,x_2) & \dots & k(x_n,x_n) \end{bmatrix}^{-1} \begin{bmatrix} k(x_1,x) \\ k(x_2,x) \\ \vdots \\ k(x_n,x) \end{bmatrix} = \sum_{i=1}^{n} \alpha_i k(x_i,x) \tag{12}$$

Some commonly used kernels

- Linear kernel: $k(x, x') = x^\top x'$
- Polynomial kernel: $k(x, x') = (r + x^\top x')^d$
- Gaussian (RBF) kernel: $k(x, x') = \exp\left(-\frac{\|x-x'\|_2^2}{2\sigma^2}\right)$

Some implications

- We suddenly can compute infinite-dimensional features. Does that mean we don't need to craft features anymore?
- How do we use kernels for classification?
- Are neural networks kernels?
- The runtime of computing the closed-form solution with kernels is $O(n^3)$.
- The inference time for computing $f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, n)$ is $O(n)$.