CSE 152: Computer Vision

Hao Su

Lecture 5: Machine Learning Crash Course: Unsupervised Learning

Photo: CMU Machine Learning Department Protests G20

Our approach

- We will look at ML as a tool. We will not detail the underpinnings of each learning method.

- Please take a machine learning course if you want to know more!

Machine Learning

- Learn from and make predictions on data.
- Arguably the greatest export from computing to other scientific fields.
- Statisticians might disagree with CompScis on the true origins…

Agenda

- Principal Component Analysis (PCA)
- RANSAC?

Agenda

- Principal Component Analysis (PCA)
- RANSAC?

Suppose you have **10 million** face images,

Question:

1. How can you find the 5 faces closest to a query (maybe yours!) in just 0.1 sec?

2. How can you show all of them in a single picture?

Suppose you have **10 million** face images,

Question:

1. How c … query (mayb …

2. How can you show all of them … picture?

SVD can help you do it!

Data as Points in a Euclidean Space

- While data can be represented as a high-dimensional vector,
- Lower-dimensional structure is often present in the data

Eigenfaces

The ATT face database (formerly the ORL database), 10 pictures of 40 subjects each

Data as Points in a Euclidean Space

- While data can be represented as a high-dimensional vector,
- Lower-dimensional structure is often present in the data
- Sometimes this lower-d structure corresponds to linear subspaces within the original space.

Principal Component Analysis

- Given data points in d dimensions
- Convert them to data points in $r<d$ dimensions
- With minimal loss of information

Data Compression

Reduce data from 2D to 1D

May be width and height of TVs

Data Compression

Reduce data from 2D to 1D

$$x^{(1)} \rightarrow z^{(1)}$$
$$x^{(2)} \rightarrow z^{(2)}$$
$$\vdots$$
$$x^{(m)} \rightarrow z^{(m)}$$

Andrew Ng

Data Compression

Reduce data from 3D to 2D

Andrew Ng

Principal Component Analysis (PCA) problem formulation

Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u^{(1)} \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.

Reduce from n-dimension to k-dimension: Find k vectors $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ onto which to project the data, so as to minimize the projection error.

Andrew Ng

Principal Component Analysis

Goal: Find r-dim projection that best preserves variance

1. Compute mean vector μ and covariance matrix C of original points
2. Compute eigenvectors and eigenvalues of C
3. Select top r eigenvectors
4. Project points onto subspace spanned by them:

$$y = U^T \cdot (x - \mu)$$

where y is the new point, x is the old one, and the rows of U^T are the eigenvectors

Variance

- Variance:
  - Measure of the deviation from the mean for points in one dimension

Covariance

- Variance:
  - Measure of the deviation from the mean for points in one dimension
- Covariance:
  - Measure of how much the variation of one dimension implies the variation of another

- **Covariance is measured between two dimensions**
- **Covariance sees if there is a relation between two dimensions**
- **Covariance between one dimension is the variance**

Positive: Both dimensions increase or decrease together

Negative: While one increase the other decrease

Covariance

- Used to find relationships between dimensions in high dimensional data sets

$$c_{jk} = \frac{1}{N}\sum_{i=1}^{N}(X_{ij} - \mathbb{E}(X_j))(X_{ik} - \mathbb{E}(X_k))$$

The Sample mean

Group all c_{jk} into a big covariance matrix C:

$$C = \frac{1}{N}\hat{X}^T\hat{X}$$

where $\hat{X}$ is the centered version of data matrix by removing mean

Principal Component Analysis

Input: $\mathbf{x} \in \mathbb{R}^D$: $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$

Set of basis vectors: $\mathbf{u}_1, \ldots, \mathbf{u}_K$

Summarize a *D* dimensional vector *X* with *K* dimensional feature vector *h(x)*

$$h(\mathbf{x}) = \begin{bmatrix} u_1^T \cdot \mathbf{x} \\ u_2^T \cdot \mathbf{x} \\ \ldots \\ u_K^T \cdot \mathbf{x} \end{bmatrix}$$

Principal Component Analysis

$$\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_K]$$

Basis vectors are orthonormal

$$\mathbf{u}_i^T \mathbf{u}_j = 0$$

$$||\mathbf{u}_j|| = 1$$

New data representation h(x)

$$z_j = u_j^T \cdot \mathbf{x}$$
$$h(\mathbf{x}) = [z_1, \ldots, z_K]^T$$

Principal Component Analysis

$$\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_K]$$

New data representation *h(x)*

$$h(\mathbf{x}) = \mathbf{U}^T \mathbf{x}$$

But centering the data x by subtracting the mean gives better results:

$$h(\mathbf{x}) = \mathbf{U}^T (\mathbf{x} - \mu_0)$$

Empirical mean of the data $\mu_0 = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$

PCA by SVD

Instead of computing the eigen-decomposition of the covariance $C = X^T X$ (assume that the data matrix X is centered), you can use SVD to solve PCA.

- Suppose that $X = U\Sigma V^T$, note that $X^T X = V\Sigma^2 V^T$.
- So eigenvectors are just the right singular vectors of X!

Why the algorithm is correct?

Suppose that the data matrix is $X = \begin{bmatrix} x_1^T \\ \cdots \\ x_n^T \end{bmatrix}$, i.e., each row is a data vector (already centered by mean).

Let $U = [u_1, \cdots, u_k]$ be the projection matrix, whose columns are the basis of the low-dimensional space.

Then, the coordinate of data in the projected space is $Z = XU$, i.e., each row is the coordinate of a data point.

Using the coordinates and the basis, we can reconstruct the data in the original space as $X' = ZU^T = XUU^T$.

- So the sum of projection error is $\|X' - X\|_{fro}^2$, where the $\|\cdot\|_{fro}^2$ is the square of the Frobenius norm of a matrix, defined as the sum of squares of all elements in X.

- Our goal is to solve the optimization problem:

$$\begin{array}{ll} \text{minimize}_U & \|XUU^T - X\|_{fro}^2 \\ \text{subject to} & U^T U = I \end{array}$$

- The solution U is just the eigenvectors of $X^T X$ corresponding to the top k eigenvalues.

- Proof will be posted on the course website for those interested.

The space of all face images

- When viewed as vectors of pixel values, face images are extremely high-dimensional
  - 100x100 image = 10,000 dimensions
  - Slow and lots of storage
- But very few 10,000-dimensional vectors are valid face images
- We want to effectively model the subspace of face images

slide by Derek Hoiem

Eigenfaces example

Mean: μ

Top eigenvectors: $u_1,\ldots u_k$

slide by Derek Hoiem

Representation and reconstruction

- Face **x** in "face space" coordinates:

$$\mathbf{x} \rightarrow [\mathbf{u}_1^{\mathrm{T}}(\mathbf{x}-\mu), \ldots, \mathbf{u}_k^{\mathrm{T}}(\mathbf{x}-\mu)]$$
$$= \quad [z_1, \ldots, z_K]$$

- Reconstruction:

$$\hat{x} = \mu + w_1u_1 + w_2u_2 + w_3u_3 + w_4u_4 + \ldots$$

slide by Derek Hoiem

Reconstruction

P = 4

P = 200

P = 400

After computing eigenfaces using 400 face images from ORL face database

slide by Derek Hoiem

Application: Image compression

Original Image

Divide the original 372x492 image into patches:

Each patch is an instance that contains 12x12 pixels on a grid

View each as a 144-D vector

PCA compression: 144D → 60D

PCA compression: 144D → 60D

16 most important eigenvectors

PCA compression: 144D → 6D

6 most important eigenvectors

PCA compression: 144D → 3D

3 most important eigenvectors

PCA compression: 144D → 1D

60 most important eigenvectors

Looks like the discrete cosine bases of JPG!...

2D Discrete Cosine Basis

http://en.wikipedia.org/wiki/Discrete_cosine_transform

Dimensionality reduction

- PCA (Principal Component Analysis):
  - Find projection that maximize the variance
- ICA (Independent Component Analysis):
  - Very similar to PCA except that it assumes non-Guassian features
- Multidimensional Scaling:
  - Find projection that best preserves inter-point distances
- LDA(Linear Discriminant Analysis):
  - Maximizing the component axes for class-separation
- …
- …

Agenda

- Principal Component Analysis (PCA)
- RANSAC!

Goal

- Discover geometric primitives from **noisy** data
- For example, discover line(s) from

Goal

- Discover geometric primitives from **noisy** data
- For example, discover line(s) from

Challenge

- We can use PCA to find a line
- In fact, a really good solution for clean data!
- However, when there are many outliers, they will confuse PCA…

Challenge

- We can use PCA to find a line
- In fact, a really good solution for clean data!
- However, when there are many outliers, they will confuse PCA…

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in '81.

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model
4. **Goto** 2 until convergence

Repeat until the best model is found with high confidence

RANSAC

Line fitting example

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model
4. **Goto** 2 until convergence

Repeat until the best model is found with high confidence

Illustration by Savarese

RANSAC

Line fitting example

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model
4. **Goto** 2 until convergence

Repeat until the best model is found with high confidence

RANSAC

Line fitting example

$$N_I = 6$$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model
4. **Goto** 2 until convergence

Repeat until the best model is found with high confidence

RANSAC

Line fitting example

$$N_I = 6$$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model
4. **Goto** 2 until convergence

Repeat until the best model is found with high confidence

RANSAC

Line fitting example

$$N_I = 6$$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model
4. **Goto** 2 until convergence

Repeat until the best model is found with high confidence

RANSAC

$N_I = 14$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model
4. **Goto** 2 until convergence

Repeat until the best model is found with high confidence

RANSAC conclusions

Good

- Robust to outliers
- Applicable for larger number of objective function parameters

Bad

- Computational time grows quickly with fraction of outliers and number of parameters
- Not as good for getting multiple fits (though one solution is to remove inliers after each fit and repeat)