Dimensionality Reduction PCA

Machine Learning – CSE4546

Sham Kakade

University of Washington

November 8, 2016

Announcements:

- Project Milestones due date passed.
- HW3 due on Monday
  - It'll be collaborative
- HW2 grades posted today
  - Out of 82 points
- Today:
  - Questions:
  - Review: Generalization
  - Start: unsupervised learning

Generalization/Model Comparisons

Machine Learning – CSE446

Sham Kakade

University of Washington

November 1, 2016

What method should I use?

- Linear regression, logistic, SVMs?
- No regularization? Ridge? L1?

- I ran SGD without any regularization and it was ok?

Generalization

- You get N samples.
- You learn a classifier/regression f^.
- How close are you to optimal?

$$L(f^)-L(f^*) < ???$$

- (We can look at the above in expectation or with 'high' probability).

Finite Case:

- You get N samples.
- You learn a classifier/regressor f^ among K classifiers:

$$L(f^)-L(f^*) < \sqrt{\frac{\log(K/\delta)}{N}}$$

with probability $\geq 1-\delta$

Linear Regression

- N samples, d dimensions.
- L is the square loss.
- w^ is the least squares estimate.

$$L(w^)-L(w^*) < O(d/N)$$

- Need about N=O(d) samples

Sparse Linear Regression

- N samples, d dimensions, L is the square loss.
- f^ is best fit line which only uses k features (computationally intractable)

$$L(w^)-L(w^*) < k \log(d)/N$$

- true of Lasso under stronger assumptions: ”incoherence”
- When do like sparse regression??
  - When we believe there are a few of GOOD features.

Learning a Halfspace

- You get N samples, in D dimensions.
- L is the 0/1 loss.
- ŵ is the empirical risk minimizer (computationally infeasible to compute)

$$L(\hat{w}) - L(w^*) < \sqrt{d \log(N)/N}$$

- Need N=O(d) samples

What about Regularization?

- Let's look at (dual) constrained problem
- Minimize:

$$\min \hat{L}(w)$$

$$\text{such } ||w||_{??} < W_+$$

- Where L^ is our training error.

Optimization and Regularization?

- I did SGD without regularization and it was fine?
- "Early stopping" implicitly regularizes (in L2)

L2 Regularization

- Assume $||w||_2 < W_2$ $||x||_2 < R_2$
- L is some convex loss (logistic,hinge,square)
- w^ is the constrained minimizer (computationally tractable to compute)

$$L(w^)-L(w^*)<W_2R_2/\sqrt{N}$$

- DIMENSION FREE "margin" Bound!

L1 Regularization

- Assume $||w||_1 < W_1$ $||x||_\infty < R_\infty$
- L is some convex loss (logistic,hinge,square)
- w^ is the constrained minimizer (computationally tractable to compute)

$$L(w^)-L(w^*)<\frac{W_1 R_\infty log(d)}{\sqrt{N}}$$

- Promotes sparsity, one can think of W1 as the "sparsity level/k" (mild dimension dependence, log(d).

Dimensionality Reduction PCA

Machine Learning – CSE4546

Sham Kakade

University of Washington

November 8, 2016

Dimensionality reduction

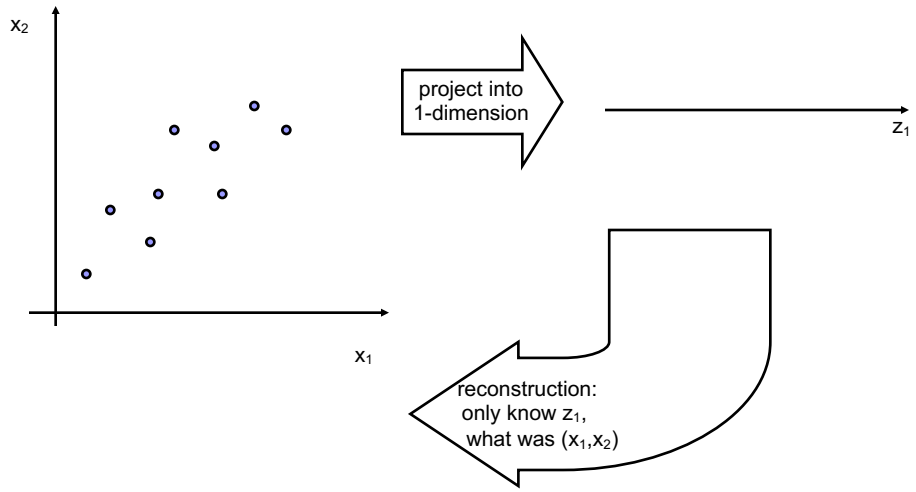
- Input data may have thousands or millions of dimensions!
  - e.g., text data has
- **Dimensionality reduction**: represent data with fewer dimensions
  - easier learning – fewer parameters
  - visualization – hard to visualize more than 3D or 4D
  - discover "intrinsic dimensionality" of data
    - high dimensional data that is truly lower dimensional

Lower dimensional projections

- Rather than picking a subset of the features, we can new features that are combinations of existing features

- Let's see this in the unsupervised setting
  - just **X**, but no Y

Linear projection and reconstruction



Principal component analysis – basic idea

- Project n-dimensional data into k-dimensional space while preserving information:
  - e.g., project space of 10000 words into 3-dimensions
  - e.g., project 3-d into 2-d
- Choose projection with minimum reconstruction error

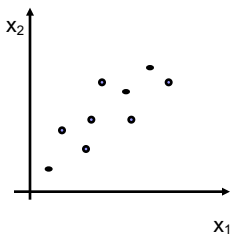
Linear projections, a review

- Project a point into a (lower dimensional) space:
  - **point**: $\mathbf{x} = (x_1,\ldots,x_d)$
  - **select a basis** – set of basis vectors – $(\mathbf{u}_1,\ldots,\mathbf{u}_k)$
    - we consider orthonormal basis:
      - $\mathbf{u}_i \bullet \mathbf{u}_i = 1$, and $\mathbf{u}_i \bullet \mathbf{u}_j = 0$ for $i \neq j$
  - **select a center** – $\bar{\mathbf{x}}$, defines offset of space
  - **best coordinates** in lower dimensional space defined by dot-products: $(z_1,\ldots,z_k)$, $z_i = (\mathbf{x}-\bar{\mathbf{x}}) \bullet \mathbf{u}_i$
    - minimum squared error

PCA finds projection that minimizes reconstruction error

- Given N data points: $\mathbf{x}^i = (x_1^i,\ldots,x_d^i)$, i=1…N
- Will represent each point as a projection:
  - $\hat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$ where: $\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}^i$ and $z_j^i = (\mathbf{x}^i - \bar{\mathbf{x}}) \cdot \mathbf{u}_j$
- PCA:
  - Given k<<d, find $(\mathbf{u}_1,\ldots,\mathbf{u}_k)$ minimizing reconstruction error:

$$error_k = \sum_{i=1}^{N} (\mathbf{x}^i - \hat{\mathbf{x}}^i)^2$$



Understanding the reconstruction error

- Note that $\mathbf{x}^i$ can be represented exactly by d-dimensional projection:

$$\mathbf{x}^i = \bar{\mathbf{x}} + \sum_{j=1}^{d} z_j^i \mathbf{u}_j$$

- Rewriting error:

$$\hat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$$

$$z_j^i = (\mathbf{x}^i - \bar{\mathbf{x}}) \cdot \mathbf{u}_j$$

- Given k<<d, find ($\mathbf{u}_1$,...,$\mathbf{u}_k$) minimizing reconstruction error:

$$error_k = \sum_{i=1}^{N} (\mathbf{x}^i - \hat{\mathbf{x}}^i)^2$$

Reconstruction error and covariance matrix

$$error_k = \sum_{i=1}^{N} \sum_{j=k+1}^{d} [\mathbf{u}_j \cdot (\mathbf{x}^i - \bar{\mathbf{x}})]^2$$

$$\Sigma = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}^i - \bar{\mathbf{x}})(\mathbf{x}^i - \bar{\mathbf{x}})^T$$

Minimizing reconstruction error and eigen vectors

- Minimizing reconstruction error equivalent to picking orthonormal basis ($\mathbf{u}_1,\ldots,\mathbf{u}_d$) minimizing:

$$error_k = N \sum_{j=k+1}^{d} \mathbf{u}_j^T \Sigma \mathbf{u}_j$$

- Eigen vector:

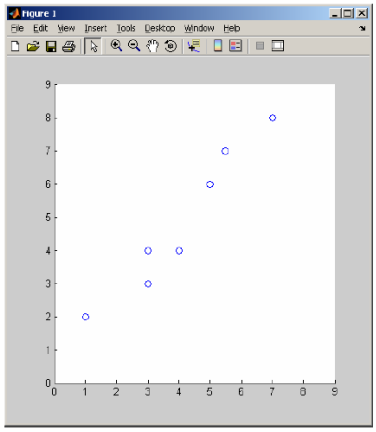
- Minimizing reconstruction error equivalent to picking ($\mathbf{u}_{k+1},\ldots,\mathbf{u}_d$) to be eigen vectors with smallest eigen values

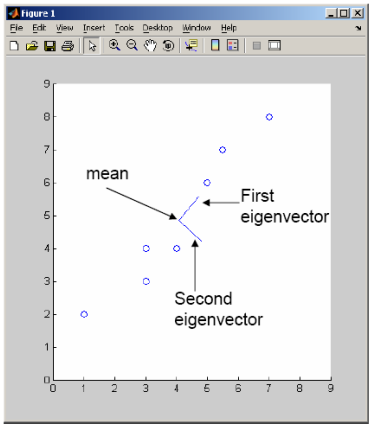
Basic PCA algoritm

- Start from m by n data matrix **X**
- **Recenter**: subtract mean from each row of **X**
  - $\mathbf{X_c} \leftarrow \mathbf{X} - \overline{\mathbf{X}}$
- **Compute covariance matrix**:
  - $\Sigma \leftarrow 1/N\ \mathbf{X_c}^T \mathbf{X_c}$
- Find **eigen vectors and values** of Σ
- **Principal components:** k eigen vectors with highest eigen values

PCA example

$$\hat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$$

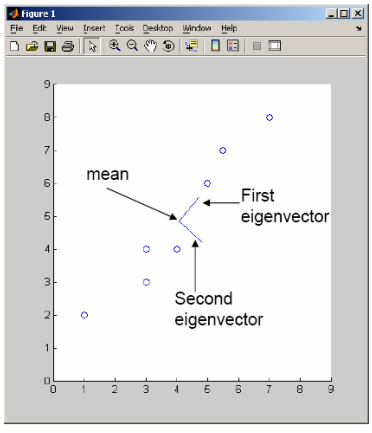


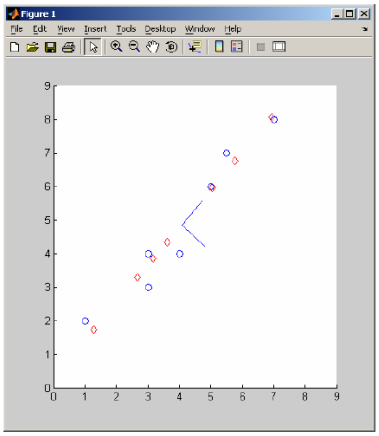


PCA example – reconstruction

$$\hat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$$

only used first principal component

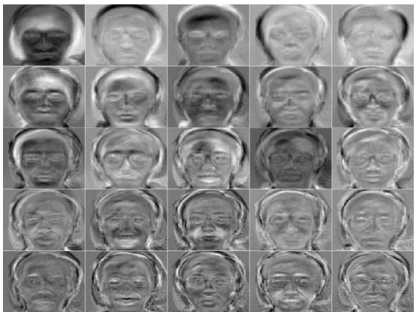




Eigenfaces [Turk, Pentland '91]

- Input images:

- Principal components:



Eigenfaces reconstruction

- Each image corresponds to adding 8 principal components:

Scaling up

- Covariance matrix can be really big!
  - Σ is d by d
  - Say, only 10000 features
  - finding eigenvectors is very slow…
- Use singular value decomposition (SVD)
  - finds to k eigenvectors
  - great implementations available, e.g., python, R, Matlab svd

SVD

- Write $\mathbf{X} = \mathbf{W}\,\mathbf{S}\,\mathbf{V}^T$
  - $\mathbf{X} \leftarrow$ data matrix, one row per datapoint
  - $\mathbf{W} \leftarrow$ weight matrix, one row per datapoint – coordinate of $\mathbf{x}^i$ in eigenspace
  - $\mathbf{S} \leftarrow$ singular value matrix, diagonal matrix
    - in our setting each entry is eigenvalue λ_j
  - $\mathbf{V}^T \leftarrow$ singular vector matrix
    - in our setting each row is eigenvector $\mathbf{v}_j$

PCA using SVD algoritm

- Start from m by n data matrix **X**
- **Recenter**: subtract mean from each row of **X**
  - $\mathbf{X_c} \leftarrow \mathbf{X} - \overline{\mathbf{X}}$
- Call SVD algorithm on $\mathbf{X_c}$ – ask for k singular vectors
- **Principal components:** k singular vectors with highest singular values (rows of $\mathbf{V}^T$)
  - **Coefficients** become:

What you need to know

- Dimensionality reduction
  - why and when it's important
- Simple feature selection
- Principal component analysis
  - minimizing reconstruction error
  - relationship to covariance matrix and eigenvectors
  - using SVD