CS545: Dimensionality Reduction

Chuck Anderson

Department of Computer Science
Colorado State University

Fall, 2009
Dimensionality Reduction

Why? To make modeling algorithms work by dealing with

- linearly dependent columns (features)
- fewer samples than features
- computation time too long
- requires too much storage
- noisy or undersampled data, leading to poor generalization

How? Project data to smaller subspace (set of direction vectors)

- directions that capture the most variation in data (unsupervised)
- directions that best discriminate between classes (supervised)

- Principal Components Analysis (PCA)
- Fisher's method
Dimensionality Reduction

Why? To make modeling algorithms work by dealing with

- linearly dependent columns (features)
Dimensionality Reduction

- Why? To make modeling algorithms work by dealing with
  - linearly dependent columns (features)
  - fewer samples than features

How? Project data to smaller subspace (set of direction vectors)
  - directions that capture the most variation in data (unsupervised)
  - directions that best discriminate between classes (supervised)

Principal Components Analysis (PCA)
Fisher's method
Dimensionality Reduction

Why? To make modeling algorithms work by dealing with
- linearly dependent columns (features)
- fewer samples than features
- computation time too long
Dimensionality Reduction

Why? To make modeling algorithms work by dealing with
- linearly dependent columns (features)
- fewer samples than features
- computation time too long
- requires too much storage

How? Project data to smaller subspace (set of direction vectors)
- directions that capture the most variation in data (unsupervised)
- directions that best discriminate between classes (supervised)

Principal Components Analysis (PCA)
Fisher's method
Dimensionality Reduction

- Why? To make modeling algorithms work by dealing with
  - linearly dependent columns (features)
  - fewer samples than features
  - computation time too long
  - requires too much storage
  - noisy or undersampled data, leading to poor generalization
Dimensionality Reduction

- Why? To make modeling algorithms work by dealing with
  - linearly dependent columns (features)
  - fewer samples than features
  - computation time too long
  - requires too much storage
  - noisy or undersampled data, leading to poor generalization
- How? Project data to smaller subspace (set of direction vectors)
Dimensionality Reduction

Why? To make modeling algorithms work by dealing with
- linearly dependent columns (features)
- fewer samples than features
- computation time too long
- requires too much storage
- noisy or undersampled data, leading to poor generalization

How? Project data to smaller subspace (set of direction vectors)
- directions that capture the most variation in data (unsupervised)
Dimensionality Reduction

- **Why?** To make modeling algorithms work by dealing with
  - linearly dependent columns (features)
  - fewer samples than features
  - computation time too long
  - requires too much storage
  - noisy or undersampled data, leading to poor generalization

- **How?** Project data to smaller subspace (set of direction vectors)
  - directions that capture the most variation in data (unsupervised)
    - Principal Components Analysis (PCA)
Dimensionality Reduction

**Why?** To make modeling algorithms work by dealing with
- linearly dependent columns (features)
- fewer samples than features
- computation time too long
- requires too much storage
- noisy or undersampled data, leading to poor generalization

**How?** Project data to smaller subspace (set of direction vectors)
- directions that capture the most variation in data (unsupervised)
  - Principal Components Analysis (PCA)
- directions that best discriminate between classes (supervised)
Dimensionality Reduction

- **Why?** To make modeling algorithms work by dealing with
  - linearly dependent columns (features)
  - fewer samples than features
  - computation time too long
  - requires too much storage
  - noisy or undersampled data, leading to poor generalization

- **How?** Project data to smaller subspace (set of direction vectors)
  - directions that capture the most variation in data (unsupervised)
    - Principal Components Analysis (PCA)
  - directions that best discriminate between classes (supervised)
    - Fisher’s method
Outline
Why PCA, when targets unknown?

- If one feature is always the same value, we can throw it away.
Why PCA, when targets unknown?

- If one feature is always the same value, we can throw it away.
- If two or more features are highly correlated, we can throw away all but one.
Why PCA, when targets unknown?

- If one feature is always the same value, we can throw it away.
- If two or more features are highly correlated, we can throw away all but one.
- If we don’t know target values, let’s project to the lower-dimensional subspace that captures the most information, in terms of variance.
Why PCA, when targets unknown?

- If one feature is always the same value, we can throw it away.
- If two or more features are highly correlated, we can throw away all but one.
- If we don’t know target values, let’s project to the lower-dimensional subspace that captures the most information, in terms of variance.
- How many dimensions? It depends.
 Desired Subspace

Let samples be \( \{x_1, x_2, \ldots, x_N\} \), each having \( D \) components. \( \bar{x} \) is the mean of the data.
Desired Subspace

- Let samples be $\{x_1, x_2, \ldots, x_N\}$, each having $D$ components. $\bar{x}$ is the mean of the data.
- Let $u_1, u_2, \ldots, u_M$ be orthonormal (orthogonal and unit length) vectors that define an $M$-dimensional subspace ($M \leq D$) within the data sample space.
Desired Subspace

- Let samples be \( \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \), each having \( D \) components. \( \bar{\mathbf{x}} \) is the mean of the data.

- Let \( \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M \) be orthonormal (orthogonal and unit length) vectors that define an \( M \)-dimensional subspace \( (M \leq D) \) within the data sample space.

- Each sample \( \mathbf{x}_n \) can be re-represented in the coordinates with respect to axes defined by \( \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M \).

\[
\mathbf{x}_n = \sum_{i=1}^{D} \alpha_{ni} \mathbf{u}_i + \bar{\mathbf{x}}
\]

But, to reduce the dimensionality of the data, we want \( \mathbf{u}_i \)'s for which

\[
\mathbf{x}_n \approx \tilde{\mathbf{x}}_n = \sum_{i=1}^{M} \alpha_{ni} \mathbf{u}_i + \bar{\mathbf{x}}
\]
The best approximation is given by those $u_i$'s that minimize

$$J = \sum_{n=1}^{N} (x_n - \tilde{x}_n)^2$$

$$= \sum_{n=1}^{N} \left( x_n - \sum_{i=1}^{M} \alpha_{ni} u_i - \bar{x} \right)^2$$

$$= \sum_{n=1}^{N} \left( x_n - \bar{x} - \sum_{i=1}^{M} \alpha_{ni} u_i \right)^2$$

$$= \sum_{n=1}^{N} \left( \hat{x}_n - \sum_{i=1}^{M} \alpha_{ni} u_i \right)^2$$

$$= \sum_{n=1}^{N} \hat{x}_n^2 - 2 \sum_{n=1}^{N} \hat{x}_n^T \left( \sum_{i=1}^{M} \alpha_{ni} u_i \right) + \sum_{n=1}^{N} \sum_{i=1}^{M} \alpha_{ni}^2$$

$$= \sum_{n=1}^{N} \hat{x}_n^2 - 2 \sum_{n=1}^{N} \sum_{i=1}^{M} \alpha_{ni} \hat{x}_n^T u_i + \sum_{n=1}^{N} \sum_{i=1}^{M} \alpha_{ni}^2$$
Derivatives, with respect to $\alpha$'s, set equal to zero, and solve for $\alpha$'s.

\[
J = \sum_{n=1}^{N} \hat{x}_n^2 - 2 \sum_{n=1}^{N} \sum_{i=1}^{M} \alpha_{ni} \hat{x}_n^T u_i + \sum_{n=1}^{N} \sum_{i=1}^{M} \alpha_{ni}^2
\]

\[
\frac{\partial J}{\partial \alpha_{mj}} = 0 = -2\hat{x}_m^T u_j + 2\alpha_{mj}
\]

\[
\alpha_{mj} = \hat{x}_m^T u_j
\]
Derivatives, with respect to $\alpha$’s, set equal to zero, and solve for $\alpha$’s.

\[
J = \sum_{n=1}^{N} \hat{x}_n^2 - 2 \sum_{n=1}^{N} \sum_{i=1}^{M} \alpha_{ni} \hat{x}_n^T u_i + \sum_{n=1}^{N} \sum_{i=1}^{M} \alpha_{ni}^2
\]

\[
\frac{\partial J}{\partial \alpha_{mj}} = 0 = -2 \hat{x}_m^T u_j + 2 \alpha_{mj}
\]

\[
\alpha_{mj} = \hat{x}_m^T u_j
\]

Plug this back into expression for $J$

\[
J = \sum_{n=1}^{N} \hat{x}_n^2 - 2 \sum_{n=1}^{N} \sum_{i=1}^{M} \alpha_{ni} \hat{x}_n^T u_i + \sum_{n=1}^{N} \sum_{i=1}^{M} \alpha_{ni}^2
\]

\[
= \sum_{n=1}^{N} \hat{x}_n^2 - 2 \sum_{n=1}^{N} \sum_{i=1}^{M} (\hat{x}_n^T u_i) \hat{x}_n^T u_i + \sum_{n=1}^{N} \sum_{i=1}^{M} (\hat{x}_n^T u_i)^2
\]

\[
= \sum_{n=1}^{N} \hat{x}_n^2 - \sum_{n=1}^{N} \sum_{i=1}^{M} (\hat{x}_n^T u_i)^2
\]
Since \((\hat{x}_n^T u_i)^2 = (\hat{x}_n^T u_i)(\hat{x}_n^T u_i) = (u_i^T \hat{x}_n)(\hat{x}_n^T u_i),\)

\[
J = \sum_{n=1}^{N} \hat{x}_n^2 - \sum_{n=1}^{N} \sum_{i=1}^{M} (\hat{x}_n^T u_i)^2 \\
= \sum_{n=1}^{N} \hat{x}_n^2 - \sum_{i=1}^{M} u_i^T \left( \sum_{n=1}^{N} \hat{x}_n \hat{x}_n^T \right) u_i \\
= \sum_{n=1}^{N} \hat{x}_n^2 - \sum_{i=1}^{M} u_i^T S u_i
\]

where \(S\), the “scatter matrix”, is

\[
S = \sum_{n=1}^{N} \hat{x}_n \hat{x}_n^T \\
= (N - 1) \frac{1}{N - 1} \sum_{n=1}^{N} (x_n - \bar{x})(x_n - \bar{x})^T \\
= (N - 1) \Sigma
\]
We still need $u_i$’s that minimize

$$J = \sum_{n=1}^{N} \hat{x}_n^2 - \sum_{i=1}^{M} u_i^T S u_i$$
- We still need $u_i$’s that minimize

$$J = \sum_{n=1}^{N} \hat{x}_n^2 - \sum_{i=1}^{M} u_i^T S u_i$$

- Same as maximizing $\sum_{i=1}^{M} u_i^T S u_i$. 

Must also guarantee that $u_i$’s are unit length. Do this with Lagrange multipliers, resulting in new expression $J_2$ to be maximized.

$$J_2 = \sum_{i=1}^{M} u_i^T S u_i - \sum_{j=1}^{M} \lambda_j (u_j^T u_j - 1)$$

Maximize by taking gradient with respect to each $u_k$

$$\nabla u_k J_2 = 2S u_k - 2\lambda_k u_k$$

$S u_k = \lambda_k u_k$ showing that $u_k$ and $\lambda_k$ are eigenvector-eigenvalue pairs of our scatter matrix $S$. 


We still need $u_i$'s that minimize

$$J = \sum_{n=1}^{N} \hat{x}_n^2 - \sum_{i=1}^{M} u_i^T S u_i$$

Same as maximizing $\sum_{i=1}^{M} u_i^T S u_i$.

Must also guarantee that $u_i$'s are unit length. Do this with Lagrange multipliers, resulting in new expression $J_2$ to be maximized.

$$J_2 = \sum_{i=1}^{M} u_i^T S u_i - \sum_{j=1}^{M} \lambda_j (u_j^T u_j - 1)$$
We still need \( u_i \)’s that minimize

\[
J = \sum_{n=1}^{N} \hat{x}_n^2 - \sum_{i=1}^{M} u_i^T S u_i
\]

Same as maximizing \( \sum_{i=1}^{M} u_i^T S u_i \).

Must also guarantee that \( u_i \)’s are unit length. Do this with Lagrange multipliers, resulting in new expression \( J_2 \) to be maximized.

\[
J_2 = \sum_{i=1}^{M} u_i^T S u_i - \sum_{j=1}^{M} \lambda_j (u_j^T u_j - 1)
\]

Maximize by taking gradient with respect to each \( u_k \)

\[
\nabla_{u_k} J_2 = 2 S u_k - 2 \lambda_k u_k = 0
\]

\[
S u_k = \lambda_k u_k
\]

showing that \( u_k \) and \( \lambda_k \) are eigenvector-eigenvalue pairs of our scatter matrix \( S \).
Now we know that $Su_k = \lambda_k u_k$. Plugging into original $J$ gives

$$J = \sum_{n=1}^{N} \hat{x}_n^2 - \sum_{i=1}^{M} u_i^T Su_i$$

$$= \sum_{n=1}^{N} \hat{x}_n^2 - \sum_{i=1}^{M} u_i^T \lambda_i u_i$$

$$= \sum_{n=1}^{N} \hat{x}_n^2 - \sum_{i=1}^{M} \lambda_i$$
Now we know that $S \mathbf{u}_k = \lambda_k \mathbf{u}_k$. Plugging into original $J$ gives

$$J = \sum_{n=1}^{N} \hat{x}_n^2 - \sum_{i=1}^{M} \mathbf{u}_i^T S \mathbf{u}_i$$

$$= \sum_{n=1}^{N} \hat{x}_n^2 - \sum_{i=1}^{M} \mathbf{u}_i^T \lambda_i \mathbf{u}_i$$

$$= \sum_{n=1}^{N} \hat{x}_n^2 - \sum_{i=1}^{M} \lambda_i$$

So, to minimize $J$, must pick the $M$ largest eigenvalues, $\lambda_i$, which determines (by association) the eigenvectors $\mathbf{u}_i$ onto which we must project to obtain data samples of reduced dimension $M$ with maximum variance.
Perform eigendecomposition of covariance matrix of sample matrix.

\[ r \leftarrow \text{eigen(cov}(X)) \]
PCA In R

- Perform eigendecomposition of covariance matrix of sample matrix.

```r
r <- eigen(cov(X))
```

- Eigenvalues in `r$values` and eigenvectors in `r$vectors`.

- Decide new dimensionality $M$.
- Project mean-subtracted samples to first $M$ vectors.

If $X$ is 200 $\times$ 100 and $M = 5$, then $newX$ is 200 $\times$ 10.
**PCA In R**

- Perform eigendecomposition of covariance matrix of sample matrix.

  ```r
  r <- eigen(cov(X))
  ```

- Eigenvalues in `r$values` and eigenvectors in `r$vectors`.

- Decide new dimensionality $M$. 

```r
Xz <- X - matrix(colMeans(X), nrow(X), ncol(X), byrow=TRUE)
newX <- Xz %*% r$vectors[,1:M]
```

If $X$ is 200×100 and $M = 5$, then $newX$ is 200×10. 
PCA In R

- Perform eigendecomposition of covariance matrix of sample matrix.
  
  ```r
  r <- eigen(cov(X))
  ```

- Eigenvalues in `r$values` and eigenvectors in `r$vectors`.

- Decide new dimensionality $M$.

- Project mean-subtracted samples to first $M$ vectors.
  
  ```r
  Xz <- X - matrix(colMeans(X), nrow(X), ncol(X), byrow=TRUE)
  newX <- Xz %*% r$vectors[, 1:M]
  ```

- If $X$ is 200 $\times$ 100 and $M = 5$, then $newX$ is 200 $\times$ 10$. 

PCA In R

- Perform eigendecomposition of covariance matrix of sample matrix.
  
  ```
r <- eigen(cov(X))
  ```

- Eigenvalues in `r$values` and eigenvectors in `r$vectors`.

- Decide new dimensionality $M$.

- Project mean-subtracted samples to first $M$ vectors.
  
  ```
  Xz <- X - matrix(colMeans(X),nrow(X),ncol(X),byrow=TRUE)
  newX <- Xz %*% r$vectors[,1:M]
  ```

- If $X$ is $200 \times 100$ and $M = 5$, then `newX` is $200 \times 10$. 
Examples of PCA

\[ x_1 \quad x_2 \]
\[ e_1 \quad e_2 \]

\[ 0 \quad 1 \quad 2 \quad 3 \]
\[ 1 \quad 2 \quad 3 \quad 4 \]

\[ -3 \quad -2 \quad -1 \quad 0 \quad 1 \quad 2 \]
\[ -3 \quad -2 \quad -1 \quad 0 \quad 1 \quad 2 \]

\[ 1 \quad 2 \quad 3 \quad 4 \]
\[ -4 \quad -2 \quad 0 \quad 2 \quad 4 \]
Four components (dimensions) capture most of the variation.

<table>
<thead>
<tr>
<th></th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>V5</th>
<th>V6</th>
<th>V7</th>
</tr>
</thead>
<tbody>
<tr>
<td>cylinders</td>
<td>-0.438</td>
<td>-0.120</td>
<td>-0.026</td>
<td>0.241</td>
<td>0.700</td>
<td>-0.195</td>
<td>-0.455</td>
</tr>
<tr>
<td>displacement</td>
<td>-0.453</td>
<td>-0.107</td>
<td>-0.022</td>
<td>0.159</td>
<td>0.176</td>
<td>-0.064</td>
<td>0.850</td>
</tr>
<tr>
<td>horsepower</td>
<td>-0.438</td>
<td>0.142</td>
<td>-0.179</td>
<td>0.125</td>
<td>-0.579</td>
<td>-0.614</td>
<td>-0.169</td>
</tr>
<tr>
<td>weight</td>
<td>-0.432</td>
<td>-0.203</td>
<td>0.013</td>
<td>0.335</td>
<td>-0.349</td>
<td>0.708</td>
<td>-0.193</td>
</tr>
<tr>
<td>acceleration</td>
<td>0.298</td>
<td>-0.482</td>
<td>0.559</td>
<td>0.527</td>
<td>-0.121</td>
<td>-0.271</td>
<td>0.018</td>
</tr>
<tr>
<td>year</td>
<td>0.215</td>
<td>-0.642</td>
<td>-0.732</td>
<td>0.029</td>
<td>-0.024</td>
<td>-0.063</td>
<td>0.009</td>
</tr>
<tr>
<td>origin</td>
<td>0.300</td>
<td>0.517</td>
<td>-0.345</td>
<td>0.714</td>
<td>0.079</td>
<td>0.057</td>
<td>0.070</td>
</tr>
</tbody>
</table>
Zipcode Digits

256 dimensional samples. Less than 50 significant principal components. Showing the first 9.
Outline
Why Fisher, when target classifications are known?

- If one feature is always the same value, we can throw it away.
Why Fisher, when target classifications are known?

- If one feature is always the same value, we can throw it away.
- If classes are not separated along one feature, we can throw it away.
Why Fisher, when target classifications are known?

- If one feature is always the same value, we can throw it away.
- If classes are not separated along one feature, we can throw it away.
- Keep directions (linear combinations of features) for which classes are well separated and tightly clustered within classes.

How many dimensions? It depends. Can only get up to $K-1$ dimensions, where $K$ is the number of classes.
Why Fisher, when target classifications are known?

- If one feature is always the same value, we can throw it away.
- If classes are not separated along one feature, we can throw it away.
- Keep directions (linear combinations of features) for which classes are well separated and tightly clustered within classes.
- How many dimensions? It depends. Can only get up to $K - 1$ dimensions, where $K$ is number of classes.
Want direction vector along which distance between classes is maximized while distance between samples within each class is minimized.
Want direction vector along which distance between classes is maximized while distance between samples within each class is minimized.

Let \( w \) be the vector and \( \mu_k \) be the mean for class \( k \) and \( \mu \) be overall mean. For two classes, the squared difference between the projected means, which we want to maximize, is

\[
(w^T \mu_1 - w^T \mu_2)^2 = w^T (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T w
\]

\[
= w^T S_B w
\]
Need measure of how compact each class is. Fisher defined within class scatter matrix in the projected space, which we want to minimize, to be

\[
\tilde{S}_k = \sum_{x \in \text{Class } k} \left( \mathbf{w}^T \mathbf{x} - \mathbf{w}^T \mu_k \right)^2
\]

\[
= \sum_{x \in \text{Class } k} \left( \mathbf{w}^T (\mathbf{x} - \mu_k) \right)^T \left( \mathbf{w}^T (\mathbf{x} - \mu_k) \right)
\]

\[
= \sum_{x \in \text{Class } k} \mathbf{w}^T (\mathbf{x} - \mu_k))(\mathbf{x} - \mu_k)^T \mathbf{w}
\]

\[
= \mathbf{w}^T S_k \mathbf{w}
\]

where \( S_k = \sum_{x \in \text{Class } k} (\mathbf{x} - \mu_k)(\mathbf{x} - \mu_k)^T \).
Combine these in one expression to be maximized, after summing the $S_k$'s into $S_W$

$$J = \frac{w^T S_B w}{w^T S_W w}$$

$$\nabla_w J = \frac{(\nabla_w w^T S_B w) w^T S_W w - (\nabla_w w^T S_W w) w^T S_B w}{(w^T S_W w)^2}$$

$$= \frac{(2S_B w) w^T S_W w - (2S_W w) w^T S_B w}{(w^T S_W w)^2}$$

which is a generalized eigenvalue problem.
• Combine these in one expression to be maximized, after summing the $S_k$'s into $S_W$

$$J = \frac{w^T S_B w}{w^T S_W w}$$

$$\nabla_w J = \frac{(\nabla_w w^T S_B w)w^T S_W w - (\nabla_w w^T S_W w)w^T S_B w}{(w^T S_W w)^2}$$

$$= \frac{(2S_B w)w^T S_W w - (2S_W w)w^T S_B w}{(w^T S_W w)^2}$$

• Setting equal to zero

$$0 = \frac{(2S_B w)w^T S_W w - (2S_W w)w^T S_B w}{(w^T S_W w)^2}$$

$$= (S_B w)w^T S_W w - (S_W w)w^T S_B w$$

$$= \frac{(S_B w)w^T S_W w}{w^T S_W w} - \frac{(S_W w)w^T S_B w}{w^T S_W w}$$

$$= S_B w - (S_W w)\frac{w^T S_B w}{w^T S_W w}$$

$$= S_B w - (S_W w)\lambda$$

$$S_B w = \lambda(S_W w)$$

which is a generalized eigenvalue problem.
If $S_W$ has full rank, so its inverse exists, this becomes

$$S_W^{-1} S_B w = \lambda w$$

a regular eigenvalue problem.
• If $S_W$ has full rank, so its inverse exists, this becomes

$$S_w^{-1} S_B w = \lambda w$$

a regular eigenvalue problem.
• But, what if $S_w^{-1}$ does not exist? One alternative is to regularize $S_W$ much like we did in ridge regression.

$$(S_w + \sigma I)^{-1} S_B w = \lambda w$$
If $S_W$ has full rank, so its inverse exists, this becomes
\[ S_w^{-1} S_B w = \lambda w \]
a regular eigenvalue problem.

But, what if $S_w^{-1}$ does not exist? One alternative is to regularize $S_W$ much like we did in ridge regression.
\[ (S_w + \sigma I)^{-1} S_B w = \lambda w \]

One way to solve for $w$ was recently described by Zhang, Dai and Jordan (“A Flexible and Efficient Algorithm for Regularized Fisher Discriminant Analysis”, In Proceedings of the Machine Learning and Knowledge Discovery in Databases: European Conference (ECML PKDD), Bled, Slovenia, pages 632-647, 2009. www.cs.berkeley.edu/~jordan/papers/zhang-dai-jordan-ecml09.pdf)
If $S_W$ has full rank, so its inverse exists, this becomes

$$S_w^{-1} S_B w = \lambda w$$

a regular eigenvalue problem.

But, what if $S_w^{-1}$ does not exist? One alternative is to regularize $S_W$ much like we did in ridge regression.

$$(S_w + \sigma I)^{-1} S_B w = \lambda w$$

One way to solve for $w$ was recently described by Zhang, Dai and Jordan (“A Flexible and Efficient Algorithm for Regularized Fisher Discriminant Analysis”, In Proceedings of the Machine Learning and Knowledge Discovery in Databases: European Conference (ECML PKDD), Bled, Slovenia, pages 632-647, 2009. www.cs.berkeley.edu/~jordan/papers/zhang-dai-jordan-ecml09.pdf)

See my translation of Matlab code in fisherRRSVD.R.
2D Examples

- $e_1, e_2$ eigenvectors of covariance
- $f_1, f_2$ eigenvectors of Fisher criterion
Zipcode Digits

Fisher vectors 1
Fisher vectors 2
Fisher vectors 3
Fisher vectors 4
Fisher vectors 5
Fisher vectors 6
Fisher vectors 7
Fisher vectors 8
Fisher vectors 9