

Dimensionality reduction for supervised learning

Daniel Hsu
djhsu@cs

7th April 2005

Based on Fradkin's and Madigan's *Experiments with Random Projections for Machine Learning*, ACM SIGKDD 2003

Outline

Motivation

Supervised learning

High dimensionality

Dimensionality reduction

Principal component analysis

Random projections

Experimental setup

Algorithms and datasets

Procedure

Results

Discussion

Motivation: supervised learning

We obtain a set T of n examples of a relation $\{(\vec{x}, y)\}$:

$$T = \{(\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n)\}$$

where $\vec{x}_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$.

For (\vec{x}, y) , think of $\vec{x} \in \mathbb{R}^d$ as the input and y as the label (positive or negative).

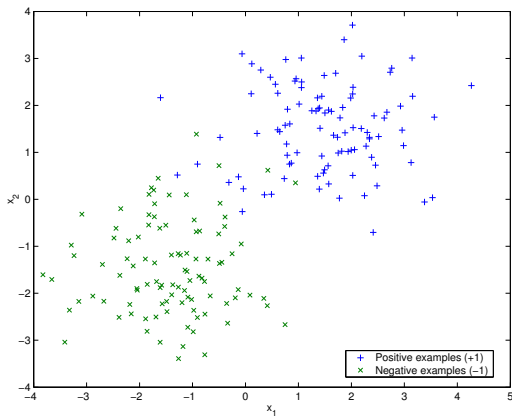
Call T the *training set*.

We want to infer a classifier \tilde{f} such that $\tilde{f}(\vec{x})$ correctly predicts the label of all \vec{x} , not just those in T .



Example

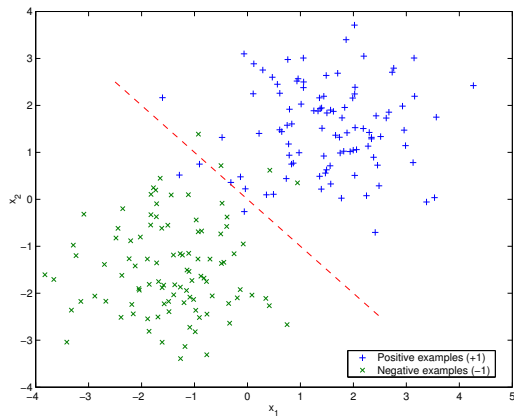
$n = 200$ and $d = 2$.





Example

$n = 200$ and $d = 2$.





How to learn \tilde{f}

Several methods exist to learn classifiers.

Some examples include:

1. Decision trees
2. Nearest neighbor
3. Support vector machine (kernel methods)

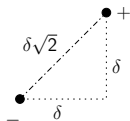
Not the focus of this talk.



Benefit of high dimensionality

Typically, each coordinate of \vec{x} represents a recorded feature of the input.

Suppose, on average, a positive example differs from a negative example on feature i by some distance $\geq \delta$. If this holds for d features, then they differ by $\geq \delta\sqrt{d}$, on average.



Intuitively, it should become easier to learn a classifier as the number of “useful” features increases.



Drawback of high dimensionality

Time complexity of learning methods is often high-polynomial (i.e. quadratic or worse) or even exponential in d .

Typical d is 10^2 or 10^3 .

Many standard learning methods become too expensive to use.

Can we reduce the dimensionality without losing the benefits?

Dimensionality reduction

Three general approaches:

1. Elimination of useless features (feature selection - next week)
2. Identification of important combinations of features (principal component analysis)
3. Distance-preserving embedding into lower dimensional subspace (random projections)

The remainder of this talk will focus on the latter two and their relative effectiveness when used with supervised learning.

Feature redundancy

The features contain a lot of redundancy. Why?

1. Some features are correlated with others.
2. Some features are irrelevant.

Can we find non-redundant combinations of the features?

Uncorrelating the features (1)

Let X be a d -dimensional random vector with mean μ and $d \times d$ covariance matrix Σ . Recall $\Sigma = \mathbb{E}[(X - \mu)(X - \mu)^T]$.

Since Σ is a covariance matrix, it is symmetric and positive semidefinite. So, we can decompose Σ as

$$\Sigma = RLR^T$$

where L is the diagonal matrix of nonnegative eigenvalues $\lambda_1, \dots, \lambda_d$ and R is the orthogonal matrix of corresponding eigenvectors $\vec{r}_1, \dots, \vec{r}_d$. Recall that $\Sigma \vec{r}_i = \lambda_i \vec{r}_i$ and $R^T R = I$.

Uncorrelating the features (2)

Question: Can we transform X so that the resulting features are uncorrelated?

Answer: Yes. Let $\tilde{X} = R^T X$. Then

$$\begin{aligned} \mathbb{E}[\tilde{X}] &= R^T \mu \\ \text{Cov}(\tilde{X}) &= R^T \Sigma R \\ &= R^T (RLR^T)R \\ &= L \end{aligned}$$

Since L is diagonal, the components of \tilde{X} are uncorrelated. The resulting features are linear combinations of the original ones, specified by R .

Eliminating dimensions of low variation

One dimensionality reduction approach is to eliminate dimensions of low variation.

The covariance matrix $L = \text{diag}(\lambda_1, \dots, \lambda_d)$. Each λ_i is the variance along one dimension of \tilde{X} .

If some λ_i are relatively small, they represent dimensions of low variation.

Principal component analysis

It is plausible that the data effectively lie in a space of some dimension $k < d$. Is there a principled (haha) way to discover this k -dimensional subspace?

One answer lies in *principal component analysis* (PCA). This derivation is loosely based on that in Hastie et al, 2001.

Suppose, for each \vec{x}_i ,

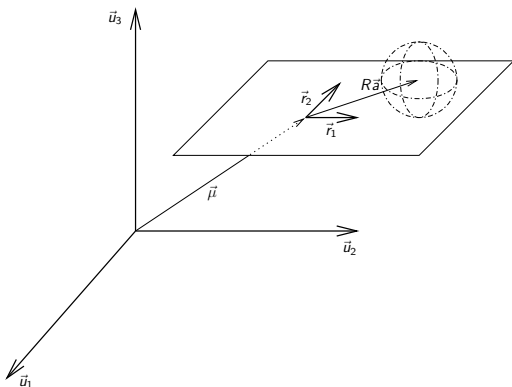
$$\vec{x}_i = \vec{\mu} + R\vec{a}_i + \mathcal{N}(\vec{0}, I)$$

where $\vec{\mu} = \frac{1}{n} \sum_{i=1}^n \vec{x}_i$, R is a $d \times k$ matrix with orthonormal columns, and $\vec{a}_i \in \mathbb{R}^k$. That is, the data lie in the affine subspace spanned by the columns of R and lifted up by $\vec{\mu}$, but the data are corrupted by a standard Gaussian noise.



Example

Here, $\{\vec{u}_1, \vec{u}_2, \vec{u}_3\}$ is the standard basis, and $\vec{x} = \vec{\mu} + R\vec{a}$ ($= \vec{\mu} + a_1\vec{r}_1 + a_2\vec{r}_2$). The sphere around \vec{x} represents the standard Gaussian noise corrupting \vec{x} .



Maximum likelihood approach

The aim of PCA is to maximize the likelihood of the data, a function of the parameters R and $\{\vec{a}_i\}$ (we'll jointly call them $\vec{\theta}$ for succinctness).

$$\begin{aligned} p(\vec{x}_1, \dots, \vec{x}_n | \vec{\theta}) &= \prod_{i=1}^n \frac{1}{(2\pi)^{d/2}} \exp \left\{ -\frac{1}{2} \|\vec{x}_i - (\vec{\mu} + R\vec{a}_i)\|^2 \right\} \\ &= \frac{1}{(2\pi)^{nd/2}} \exp \left\{ -\frac{1}{2} \sum_{i=1}^n \|\vec{x}_i - (\vec{\mu} + R\vec{a}_i)\|^2 \right\} \end{aligned}$$

$$\log p(\vec{x}_1, \dots, \vec{x}_n | \vec{\theta}) = \text{Constant} - \frac{1}{2} \sum_{i=1}^n \|\vec{x}_i - \vec{\mu} - R\vec{a}_i\|^2$$

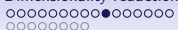
Maximizing this is equivalent to minimizing the squared-error loss.

Minimizing squared-error loss (1)

To minimize the squared-error loss $L(\vec{\theta})$, first note that

$$\begin{aligned}
 L(\vec{\theta}) &= \sum_{i=1}^n \|\vec{x}_i - \vec{\mu} - R\vec{a}_i\|^2 \\
 &= \sum_{i=1}^n \{ \|\vec{x}_i - \vec{\mu}\|^2 - 2(\vec{x}_i - \vec{\mu})^T R\vec{a}_i - (R\vec{a}_i)^T (R\vec{a}_i) \} \\
 &= \sum_{i=1}^n \{ \|\vec{x}_i - \vec{\mu}\|^2 - 2\vec{a}_i^T R^T (\vec{x}_i - \vec{\mu}) - \vec{a}_i^T R^T R \vec{a}_i \} \\
 &= \sum_{i=1}^n \{ \|\vec{x}_i - \vec{\mu}\|^2 - 2\vec{a}_i^T R^T (\vec{x}_i - \vec{\mu}) - \vec{a}_i^T \vec{a}_i \}
 \end{aligned}$$

where in the last step we use the fact that R has orthonormal columns.



Minimizing squared-error loss (2)

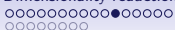
The vector derivative with respect to \vec{a}_i

$$\nabla_{\vec{a}_i} L(\vec{\theta}) = -2 \underbrace{R^T}_{k \times d} \underbrace{(\vec{x}_i - \vec{\mu})}_{d \times 1} - 2 \underbrace{\vec{a}_i}_{k \times 1}$$

is zero at $\vec{a}_i = R^T(\vec{x}_i - \vec{\mu})$.

Assume without loss of generality that $\vec{\mu} = \vec{0}$ (or else replace the \vec{x}_i with $\vec{x}_i - \vec{\mu}$). Now, the loss function to minimize becomes

$$L(\vec{\theta}) = \sum_{i=1}^n \|\vec{x}_i - RR^T \vec{x}_i\|^2$$



Minimizing squared-error loss (3)

Rewriting $L(\vec{\theta})$,

$$\begin{aligned}
 L(\vec{\theta}) &= \sum_{i=1}^n \|\vec{x}_i - RR^T \vec{x}_i\|^2 \\
 &= \sum_{i=1}^n \vec{x}_i^T \vec{x}_i - 2\vec{x}_i^T RR^T \vec{x}_i + \vec{x}_i^T RR^T RR^T \vec{x}_i \\
 &= \sum_{i=1}^n \vec{x}_i^T \vec{x}_i - \vec{x}_i^T RR^T \vec{x}_i \\
 &= \sum_{i=1}^n \|\vec{x}_i\|^2 - \|R^T \vec{x}_i\|^2
 \end{aligned}$$

we see that the objective is to maximize $\sum_{i=1}^n \|R^T \vec{x}_i\|^2$.

Minimizing squared-error loss (4)

Let the j th column of R be \vec{r}_j , and let X be the matrix with the \vec{x}_i^T as rows. Then, the function to maximize becomes

$$\begin{aligned} f(R) &= \sum_{i=1}^n \|R^T \vec{x}_i\|^2 = \sum_{i=1}^n \sum_{j=1}^k (\vec{r}_j^T \vec{x}_i)^2 \\ &= \sum_{j=1}^k \vec{r}_j^T X^T X \vec{r}_j = \sum_{j=1}^k \vec{r}_j^T \Sigma \vec{r}_j \end{aligned}$$

where $\Sigma = X^T X$ is the (scaled) empirical covariance matrix. Since the maximization problem is subject to the constraints that $\vec{r}_j^T \vec{r}_j = 1$, we can use the method of Lagrange multipliers to find the maximizers.

Minimizing squared-error loss (5)

Setting the derivative of the Lagrangian to zero and solving,

$$\begin{aligned}\mathcal{L}(R, \lambda_1, \dots, \lambda_k) &= \sum_{j=1}^k \vec{r}_j^T \Sigma \vec{r}_j - \lambda_j \vec{r}_j^T \vec{r}_j \\ \nabla_{\vec{r}_j} \mathcal{L}(R, \lambda_1, \dots, \lambda_k) &= 2\Sigma \vec{r}_j - 2\lambda_j \vec{r}_j := 0\end{aligned}$$

we conclude that the \vec{r}_j satisfy $\Sigma \vec{r}_j = \lambda_j \vec{r}_j$, i.e. the \vec{r}_j are eigenvectors of Σ . Using this fact, it follows that choosing the eigenvectors corresponding to the k largest eigenvalues solve the maximization problem, and thus they also solve the original squared-error loss minimization problem.

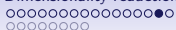
Singular value decomposition

It turns out that often computing the singular value decomposition (SVD) of X is more efficient than computing the eigen-decomposition of $X^T X$.

The SVD of X is $X = USV^T$, where U is a $n \times d$ orthogonal matrix, $S = \text{diag}(\sigma_1, \dots, \sigma_d)$, and V is $d \times d$ orthogonal matrix. Indeed,

$$\begin{aligned} X^T X &= (USV^T)^T (USV^T) \\ &= VSU^T USV^T \\ &= VS^2 V^T \end{aligned}$$

so the columns of V^T are the eigenvectors of $X^T X$ and the singular values $\sigma_1, \dots, \sigma_d$ are the square-roots of the eigenvalues of $X^T X$.



The PCA algorithm

Input: $\vec{x}_1, \dots, \vec{x}_n \in \mathbb{R}^d$, $k < d$.

1. Let X be the matrix with the \vec{x}_i as its rows.
2. Recenter X by subtracting $\frac{1}{n} \sum_{i=1}^n \vec{x}_i$ from each row of X .
3. Compute the singular value decomposition $X = USV^T$.
4. Let the columns of $R = V$ be the singular vectors corresponding to the largest k singular values in S .
5. Multiply R^T on the left of each \vec{x}_i .

Computing R has time complexity dominated by the cost of computing the singular value decomposition. If X is of rank r , the time complexity is $O(drn)$. Applying R^T to each \vec{x}_i is just n matrix multiplications and thus has time complexity $O(ndk)$.

Drawbacks of PCA

PCA can eliminate the dimension that is best for discriminating positive examples from negative examples. Suppose the data are spread parallel on each side of the linear separator. It is easy to see that the discriminating dimension will be eliminated.



Also, the time complexity of PCA is potentially quadratic in d (if $r \approx d$).

Low dimensional embeddings

Instead of looking for useless features, we can instead look for distance-preserving embeddings into lower dimensional subspaces.

What does it mean to be distance-preserving?

Suppose the distance between two points is r . After projecting these points, we want the distance to not change by much. That is, we want it to change only by at most a factor of $(1 \pm \epsilon)$ for some small ϵ .

A lemma due to Johnson and Lindenstrauss says this is possible with high probability. Moreover, the projection is easy to construct: a random projection will suffice.

The Johnson-Lindenstrauss lemma

Lemma (Johnson and Lindenstrauss, 1984)

Fix $\epsilon > 0$. Then, for any $\vec{x}_1, \dots, \vec{x}_n \in \mathbb{R}^d$, there exists $f : \mathbb{R}^d \rightarrow \mathbb{R}^k$, where $k \geq O(\epsilon^{-2} \log n)$, such that if

$$\|\vec{x}_i - \vec{x}_j\|^2 = r$$

then

$$(1 - \epsilon)r \leq \|f(\vec{x}_i) - f(\vec{x}_j)\|^2 \leq (1 + \epsilon)r.$$

The Johnson-Lindenstrauss lemma: history

The original proof of the lemma uses geometric approximation techniques. This proof was simplified by Frankl and Meahara (1988), then later simplified even more by Indyk and Motwani (1998) and, independently, by Dasgupta and Gupta (1999).

The general proof shows that a randomly chosen projection preserves the pairwise distances with high probability.

The Johnson-Lindenstrauss lemma: proof idea

1. Projecting a vector onto a random subspace is the same as fixing a subspace and projecting a random vector onto it.
2. Let $X = (X_1, \dots, X_d)$. Then its squared-length $\|X\|^2 = X_1^2 + \dots + X_d^2$ is sharply concentrated around $\mathbb{E}[\|X\|^2]$ with high probability.
3. Such a statement also holds for the squared distance $\|X - Y\|^2$ between two random vectors X and Y , since $X - Y$ is just another random vector.
4. We expect the distance to contract when projecting to lower dimensions; all that is needed to restore the original distance is to scale up each vector.

An alternative construction

An alternative version of the lemma due to Achlioptas uses a different type of random projection.

The random projection matrix will have entries in $\{\pm 1\}$. So, it is more easily constructed, and the projection only requires multiplication by ± 1 .

Similar guarantees about the projection hold.

The Achlioptas theorem

Theorem (Achlioptas, 2003)

Fix $\epsilon > 0, \beta > 0$. Let $\vec{x}_1, \dots, \vec{x}_n \in \mathbb{R}^d$, and

$$k \geq \frac{4 + 2\beta}{\epsilon^2/2 - \epsilon^3/3} \log n.$$

Now, let R be a random $d \times k$ matrix where each position is chosen independently from $\{\pm 1\}$ with equal probability. Then, if

$$\|\vec{x}_i - \vec{x}_j\|^2 = r$$

then

$$(1 - \epsilon)r \leq \left\| \frac{1}{k} (R^T \vec{x}_i - R^T \vec{x}_j) \right\|^2 \leq (1 + \epsilon)r$$

with probability at least $1 - n^{-\beta}$.

The random projection algorithm

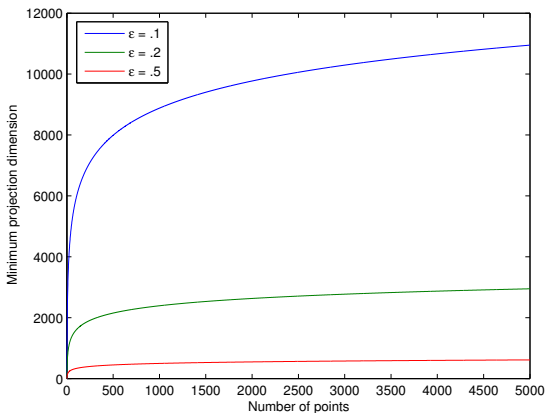
Input: $\vec{x}_1, \dots, \vec{x}_n \in \mathbb{R}^d$, $k < d$.

1. Construct a $d \times k$ matrix with entries each independently chosen to be ± 1 equally likely.
2. Multiply $\frac{1}{\sqrt{k}}R^T$ on the left of each \vec{x}_i .

Fradkin and Madigan note that, in supervised learning, only separation between points is important; actual pairwise distances need not be preserved. Consequently, the $\frac{1}{\sqrt{k}}$ scaling can be omitted.

Example of performance parameters

Fix $\beta = 1$, so the probability of achieving the performance guarantees is $1 - 1/n$.

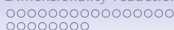


Experimental setup (1)

Fradkin and Madigan designed experiments to compare PCA and random projection in the context of supervised learning.

Used a variety of standard supervised learning tools with default settings:

- Decision trees (C4.5 implementation)
- Nearest neighbor methods
- Support vector machine (SVMLight implementation)



Experimental setup (2)

Used a variety of data sets from the UCI repository as well as from biological/medical literature.

Recall that n is the size of the dataset, and d is the original dimensionality of the data.

From UCI:

- Ionosphere ($n = 351$, $d = 34$)
- Spambase ($n = 4601$, $d = 57$)
- Internet ads ($n = 3279$, $d = 1554$)

From biological/medical literature:

- Colon ($n = 62$, $d = 2000$)
- Leukemia ($n = 72$, $d = 3571$)

Experimental procedure

The procedure for each dataset varies the projection dimensions $\{d_1, \dots, d_m\}$.

- Do s times:
 1. Split the dataset into a training set and test set.
 2. Normalize the data using empirical mean and variance of the training set.
 3. For each projection dimension k :
 - 3.1 Perform PCA on the training set with target dimension k to get the projection matrix R . Project both the training set and test set.
 - 3.2 Perform random projection on the training set and test set with target dimension k .
 - 3.3 Train on each low dimensional training set using each learning tool. Apply classifier to each low dimensional test set and evaluate the performance.

Training set and testing set splits

Performed $s = 30$ splits on the Internet ads dataset, and $s = 100$ on the other datasets.

The size of the test set was kept constant for a given dataset. Kept about 1/3 of large datasets for the test set. For smaller datasets, reserved more of the dataset for the training set.

- Ionosphere 51
- Spambase 1601
- Internet ads 1079
- Colon 12
- Leukemia 12

Projection dimensions

The training set sizes are upper bounds for the target projection dimensions.

Random projection has an $O(\log n)$ lower bound on the target projection.

So, use both low and high projection dimensions to compare.

- Ionosphere and Spambase $k \in \{5, 10, 15, 20, 25, 30\}$
- Colon and Leukemia $k \in \{5, 10, 25, 50, 100, 200, 500\}$
- Spambase $k \in \{5, 10, 25, 50, 100, 200, 500\}$

Results (1)

The performance measure is classification accuracy.

Nearest neighbor methods are least affected by choice between PCA and random projection.

Both PCA and random projection are *efficient* in projection dimension versus accuracy.

Nearest neighbor methods rely on distance computations for performance; separation of classes and usefulness of features are less important. So performance of random projection not surprising in this case.

Results (2)

Support vector machine performance worse in lower dimensions, but performance increases rapidly as the dimensionality increases.

Number of support vectors used at each dimension suggests level of complexity of the data set (more support vectors means harder to separate the classes).

Results (3)

Using PCA on the training sets reduces the number of support vectors for the Internet ads, Colon, and Leukemia training sets, compared to the number of support vectors without dimensionality reduction. Using random projection on the training sets kept the number of support vectors about the same on the Colon and Leukemia training sets. The small size of the Colon and Leukemia datasets could make it easier to separate the classes.

In low dimensions, using random projection always led to more support vectors than when using PCA. Random projection adversely affects separability of classes more so than PCA.

Results (4)

Both PCA and random projection cause poor decision tree performance.

Decision trees are not very tolerant of data transformation and noise.

Discussion

Random projection best suited for nearest neighbor methods.

Random projection also suitable for support vector machine methods, but does not perform as well as PCA.

Dimensionality reduction is not suitable for decision tree methods.

Random projection is attractive due to its computational efficiency and strong performance guarantee.

An alternative comparison of PCA and random projection would allow methods using random projection the same amount of time as methods employing PCA in the preprocessing (allow more trials to get a better classifier).

References

Dimitris Achlioptas, *Database-friendly random projections*, Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 2001.

Sanjoy Dasgupta, *Experiments with Random Projections*, Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence, 2000.

Dmitriy Fradkin and David Madigan, *Experiments with Random Projections for Machine Learning*, Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The Elements of Statistical Learning*, Springer-Verlag, 2001.