

Lecture 8: The Curse of Dimensionality

Attendance code: 3MQWR9DH

Iain Styles

5 November 2018

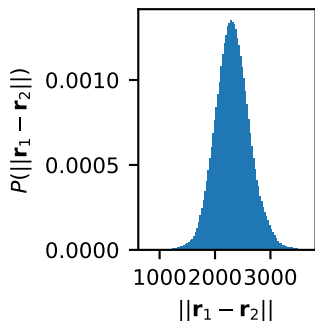
Learning Outcomes

By the end of this lecture you should:

- ▶ Understand why the dimensionality of data can be reduced without losing information
- ▶ Understand and explain the properties of high dimensional random vectors
- ▶ Explain the implications of the Johnson-Lindenstrauss lemma
- ▶ Reduce the dimensionality of a dataset using random projections

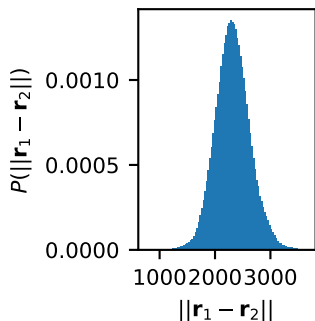
Distances in MNIST

- ▶ 1000 points from the test set and 1000 points from the training set



Distances in MNIST

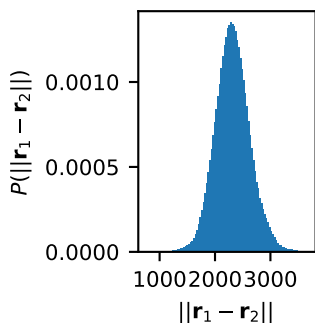
- ▶ 1000 points from the test set and 1000 points from the training set



- ▶ Mean/median of ≈ 2300 and a standard deviation of ≈ 300 .
- ▶ 68% of pairwise distances lie between 2000 and 2600, and 95% between 1700 and 2900.

Distances in MNIST

- ▶ 1000 points from the test set and 1000 points from the training set



- ▶ Mean/median of ≈ 2300 and a standard deviation of ≈ 300 .
- ▶ 68% of pairwise distances lie between 2000 and 2600, and 95% between 1700 and 2900.
- ▶ Not as "bad" as we might expect? Why?

Characteristics of "real" data

- ▶ Data is not uniformly distributed

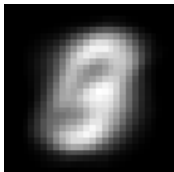
Characteristics of "real" data

- ▶ Data is not uniformly distributed
- ▶ Lies on some low-dimensional structure embedded in the high-dimensional space.

Characteristics of "real" data

- ▶ Data is not uniformly distributed
- ▶ Lies on some low-dimensional structure embedded in the high-dimensional space.
- ▶ Example: non-varying pixels in MNIST

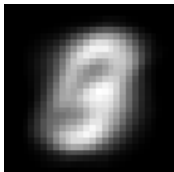
Mean



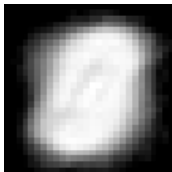
Characteristics of "real" data

- ▶ Data is not uniformly distributed
- ▶ Lies on some low-dimensional structure embedded in the high-dimensional space.
- ▶ Example: non-varying pixels in MNIST

Mean

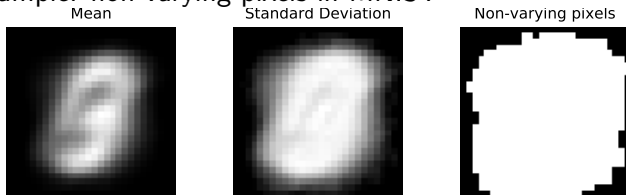


Standard Deviation



Characteristics of "real" data

- ▶ Data is not uniformly distributed
- ▶ Lies on some low-dimensional structure embedded in the high-dimensional space.
- ▶ Example: non-varying pixels in MNIST



- ▶ 175 pixels (22%) do not vary and can be ignored.

Intrinsic Dimensionality

- ▶ More generally, low-dimensional structure can be bent into high-dimensional objects

Intrinsic Dimensionality

- ▶ More generally, low-dimensional structure can be bent into high-dimensional objects
- ▶ A rolled-up poster (intrinsically 2d); cooked spaghetti (1d)

Intrinsic Dimensionality

- ▶ More generally, low-dimensional structure can be bent into high-dimensional objects
- ▶ A rolled-up poster (intrinsically 2d); cooked spaghetti (1d)
- ▶ MNIST digits have intrinsic (underlying) degree of freedom
 - ▶ Ten digit class.
 - ▶ Width and height
 - ▶ Minor shape variation (1 vs 1)
- ▶ A few ten's of intrinsic variables
- ▶ How do we extract them?

Dimensionality Reduction

- ▶ Limit to *linear* methods: single, global linear transformation.
- ▶ Aim: find a transformation to a new coordinate system that preserves structure and reduces the dimensionality.
- ▶ Hope: that it will mitigate the issues seen in high-dimensional spaces.
- ▶ Many methods: PCA, NMF, LDA
- ▶ We will study random projections: simple, low-cost, elegant theory.

Random Projections

- ▶ Very simple basic idea
- ▶ N samples in M dimensions, arranged as an $M \times N$ matrix \mathbf{X}

Random Projections

- ▶ Very simple basic idea
- ▶ N samples in M dimensions, arranged as an $M \times N$ matrix \mathbf{X}
 - ▶ Generate K random vectors with M components randomly sampled from $\mathcal{N}(0,1)$. Arrange these as a matrix \mathbf{R} of size $M \times K$, with one vector per column.

Random Projections

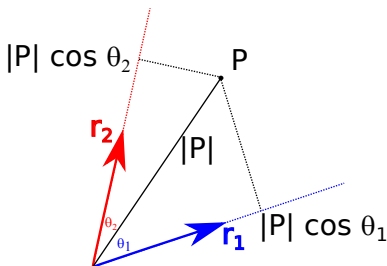
- ▶ Very simple basic idea
- ▶ N samples in M dimensions, arranged as an $M \times N$ matrix \mathbf{X}
 - ▶ Generate K random vectors with M components randomly sampled from $\mathcal{N}(0,1)$. Arrange these as a matrix \mathbf{R} of size $M \times K$, with one vector per column.
 - ▶ Normalise the columns of \mathbf{R} so each has unit length.

Random Projections

- ▶ Very simple basic idea
- ▶ N samples in M dimensions, arranged as an $M \times N$ matrix \mathbf{X}
 - ▶ Generate K random vectors with M components randomly sampled from $\mathcal{N}(0, 1)$. Arrange these as a matrix \mathbf{R} of size $M \times K$, with one vector per column.
 - ▶ Normalise the columns of \mathbf{R} so each has unit length.
 - ▶ Compute $\mathbf{X}' = \mathbf{R}^T \mathbf{X}$

Random Projections

- ▶ Very simple basic idea
- ▶ N samples in M dimensions, arranged as an $M \times N$ matrix \mathbf{X}
 - ▶ Generate K random vectors with M components randomly sampled from $\mathcal{N}(0, 1)$. Arrange these as a matrix \mathbf{R} of size $M \times K$, with one vector per column.
 - ▶ Normalise the columns of \mathbf{R} so each has unit length.
 - ▶ Compute $\mathbf{X}' = \mathbf{R}^T \mathbf{X}$
 - ▶ The columns of \mathbf{X}' contain the samples projected onto K dimensions



Why does random projection work?

- ▶ Distances in large n are of limited use
- ▶ Desire to map to a lower-dimensional space to make distances meaningful again
- ▶ Map should preserve local structure (nearest-neighbours stay nearest neighbours, etc)

Why does random projection work?

- ▶ Distances in large n are of limited use
- ▶ Desire to map to a lower-dimensional space to make distances meaningful again
- ▶ Map should preserve local structure (nearest-neighbours stay nearest neighbours, etc)
- ▶ Key: The **Johnson-Lindenstrauss lemma**
- ▶ Given a set X of N data points in $\mathbb{R}^M \dots$

Why does random projection work?

- ▶ Distances in large n are of limited use
- ▶ Desire to map to a lower-dimensional space to make distances meaningful again
- ▶ Map should preserve local structure (nearest-neighbours stay nearest neighbours, etc)
- ▶ Key: The **Johnson-Lindenstrauss lemma**
- ▶ Given a set X of N data points in $\mathbb{R}^M \dots$
- ▶ There is a linear map $f : \mathbb{R}^M \mapsto \mathbb{R}^K$ where $K < M$

Why does random projection work?

- ▶ Distances in large n are of limited use
- ▶ Desire to map to a lower-dimensional space to make distances meaningful again
- ▶ Map should preserve local structure (nearest-neighbours stay nearest neighbours, etc)
- ▶ Key: The **Johnson-Lindenstrauss lemma**
- ▶ Given a set X of N data points in $\mathbb{R}^M \dots$
- ▶ There is a linear map $f : \mathbb{R}^M \mapsto \mathbb{R}^K$ where $K < M$
- ▶ The map obeys

$$(1 - \varepsilon)\|x_1 - x_2\|^2 \leq \|f(x_1) - f(x_2)\|^2 \leq (1 + \varepsilon)\|x_1 - x_2\|^2$$

for all $x_1, x_2 \in X$ and for $0 < \varepsilon < 1$ and $K > 8 \ln(N)/\varepsilon^2$.

Interpreting Johnson-Lindenstrauss

- Helpful to rewrite as

$$1 - \varepsilon \leq \frac{\|f(x_1) - f(x_2)\|^2}{\|x_1 - x_2\|} \leq 1 + \varepsilon \quad (1)$$

Interpreting Johnson-Lindenstrauss

- ▶ Helpful to rewrite as

$$1 - \varepsilon \leq \frac{\|f(x_1) - f(x_2)\|^2}{\|x_1 - x_2\|^2} \leq 1 + \varepsilon \quad (1)$$

- ▶ Clarifies that J-L is a statement about *relative* distances.
- ▶ The map f preserves relative distances to a range $1 \pm \varepsilon$
- ▶ Since $K > 8 \ln(N)/\varepsilon^2$, smaller ε requires larger K .

Finding the map f

- ▶ J-L states that the linear map $f : \mathbb{R}^M \mapsto \mathbb{R}^K$ is onto a *random subspace*.
- ▶ Refinement by Frankl and Maehara (2008) states that J-L holds when “ f is a random, orthonormal linear transformation”
- ▶ Random? ✓
- ▶ Linear? ✓
- ▶ Orthonormal?

Finding the map f

- ▶ J-L states that the linear map $f : \mathbb{R}^M \mapsto \mathbb{R}^K$ is onto a *random subspace*.
- ▶ Refinement by Frankl and Maehara (2008) states that J-L holds when “ f is a random, orthonormal linear transformation”
- ▶ Random? ✓
- ▶ Linear? ✓
- ▶ Orthonormal?
- ▶ Normalisation ($\mathbf{r}_i \cdot \mathbf{r}_i = 1$) is easy
- ▶ Orthogonality ($\mathbf{r}_i \cdot \mathbf{r}_j = 0, i \neq j$) is less so - can be costly

Finding the map f

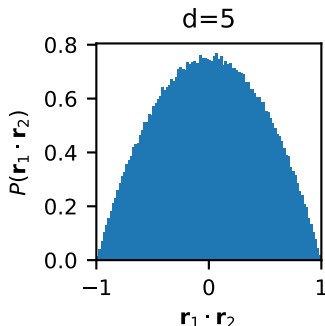
- ▶ J-L states that the linear map $f : \mathbb{R}^M \mapsto \mathbb{R}^K$ is onto a *random subspace*.
- ▶ Refinement by Frankl and Maehara (2008) states that J-L holds when “ f is a random, orthonormal linear transformation”
- ▶ Random? ✓
- ▶ Linear? ✓
- ▶ Orthonormal?
- ▶ Normalisation ($\mathbf{r}_i \cdot \mathbf{r}_i = 1$) is easy
- ▶ Orthogonality ($\mathbf{r}_i \cdot \mathbf{r}_j = 0, i \neq j$) is less so - can be costly
- ▶ The curse of dimensionality to the rescue...!

Orthogonality of Random Vectors

- ▶ $\mathbf{r}_i \cdot \mathbf{r}_j = 0 \rightarrow \theta = 0$
- ▶ Generate 100,000 normalised random vectors of different dimensionality
- ▶ Compute the angle between each pair $(\mathbf{r}_i \cdot \mathbf{r}_j)$, plot histogram

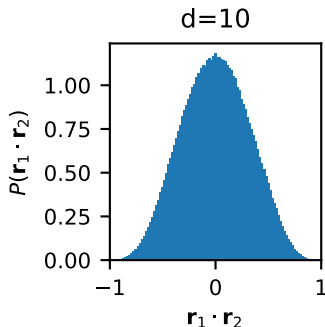
Orthogonality of Random Vectors

- ▶ $\mathbf{r}_i \cdot \mathbf{r}_j = 0 \rightarrow \theta = 0$
- ▶ Generate 100,000 normalised random vectors of different dimensionality
- ▶ Compute the angle between each pair $(\mathbf{r}_i \cdot \mathbf{r}_j)$, plot histogram



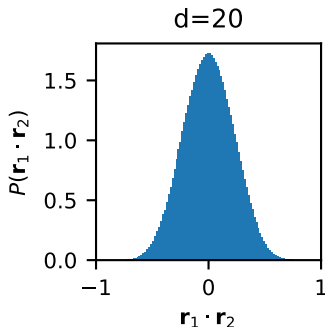
Orthogonality of Random Vectors

- ▶ $\mathbf{r}_i \cdot \mathbf{r}_j = 0 \rightarrow \theta = 0$
- ▶ Generate 100,000 normalised random vectors of different dimensionality
- ▶ Compute the angle between each pair $(\mathbf{r}_i \cdot \mathbf{r}_j)$, plot histogram



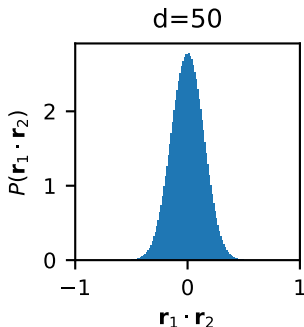
Orthogonality of Random Vectors

- ▶ $\mathbf{r}_i \cdot \mathbf{r}_j = 0 \rightarrow \theta = 0$
- ▶ Generate 100,000 normalised random vectors of different dimensionality
- ▶ Compute the angle between each pair $(\mathbf{r}_i \cdot \mathbf{r}_j)$, plot histogram



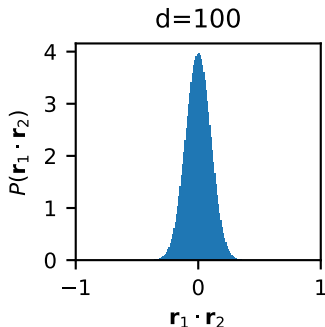
Orthogonality of Random Vectors

- ▶ $\mathbf{r}_i \cdot \mathbf{r}_j = 0 \rightarrow \theta = 0$
- ▶ Generate 100,000 normalised random vectors of different dimensionality
- ▶ Compute the angle between each pair $(\mathbf{r}_i \cdot \mathbf{r}_j)$, plot histogram



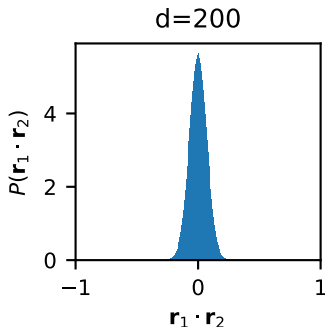
Orthogonality of Random Vectors

- ▶ $\mathbf{r}_i \cdot \mathbf{r}_j = 0 \rightarrow \theta = 0$
- ▶ Generate 100,000 normalised random vectors of different dimensionality
- ▶ Compute the angle between each pair $(\mathbf{r}_i \cdot \mathbf{r}_j)$, plot histogram



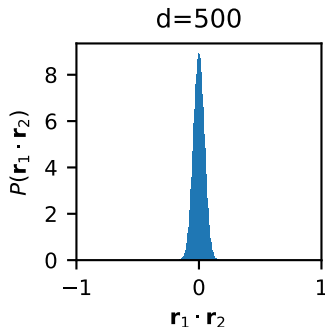
Orthogonality of Random Vectors

- ▶ $\mathbf{r}_i \cdot \mathbf{r}_j = 0 \rightarrow \theta = 0$
- ▶ Generate 100,000 normalised random vectors of different dimensionality
- ▶ Compute the angle between each pair $(\mathbf{r}_i \cdot \mathbf{r}_j)$, plot histogram



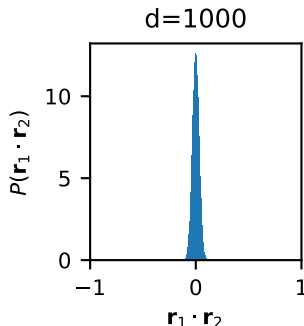
Orthogonality of Random Vectors

- ▶ $\mathbf{r}_i \cdot \mathbf{r}_j = 0 \rightarrow \theta = 0$
- ▶ Generate 100,000 normalised random vectors of different dimensionality
- ▶ Compute the angle between each pair $(\mathbf{r}_i \cdot \mathbf{r}_j)$, plot histogram



Orthogonality of Random Vectors

- ▶ $\mathbf{r}_i \cdot \mathbf{r}_j = 0 \rightarrow \theta = 0$
- ▶ Generate 100,000 normalised random vectors of different dimensionality
- ▶ Compute the angle between each pair $(\mathbf{r}_i \cdot \mathbf{r}_j)$, plot histogram



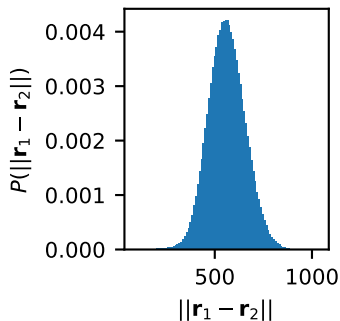
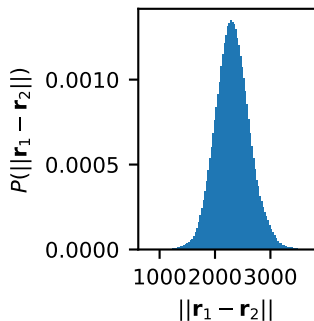
Orthogonality of Random Vectors

- ▶ Random vectors in high dimensions are (nearly) orthogonal!

Orthogonality of Random Vectors

- ▶ Random vectors in high dimensions are (nearly) orthogonal!
- ▶ So no need to explicitly orthogonalise
- ▶ Means random projection is very cheap indeed!

Effect on MNIST Data



- ▶ Absolute distances reduced
- ▶ Relative distances preserved to within factor ≈ 3

Summary

- ▶ Problems in high dimensions
- ▶ One way for overcoming them

Summary

- ▶ Problems in high dimensions
- ▶ One way for overcoming them
- ▶ Next time: pushing the performance limits
- ▶ An alternative generative approach to classification