Geometric methods in vector spaces

Distributional Semantic Models

Stefan Evert[1] & Alessandro Lenci[2]

[1]University of Osnabrück, Germany
[2]University of Pisa, Italy

ESSLLI'09 Bordeaux

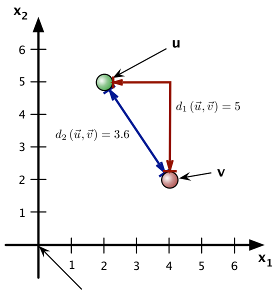
Geometry and meaning

- So far: apply vector methods and matrix algebra to DSMs
- Geometric intuition: distance $\simeq$ semantic (dis)similarity
  - nearest neighbours
  - clustering
  - semantic maps
  - representation for connectionist models

☞ We need a mathematical notion of distance!

Measuring distance

- **Distance** between vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ ➜ (dis)**similarity** of data points
  - $\mathbf{u} = (u_1, \dots, u_n)$
  - $\mathbf{v} = (v_1, \dots, v_n)$
- **Euclidean** distance $d_2(\mathbf{u}, \mathbf{v})$
- "City block" **Manhattan** distance $d_1(\mathbf{u}, \mathbf{v})$
- Both are special cases of the **Minkowski** p-distance $d_p(\mathbf{u}, \mathbf{v})$ (for $p \in [1, \infty]$)



$$d_p(\mathbf{u}, \mathbf{v}) := \left(|u_1 - v_1|^p + \cdots + |u_n - v_n|^p\right)^{1/p}$$

$$d_\infty(\mathbf{u}, \mathbf{v}) = \max\left\{|u_1 - v_1|, \dots, |u_n - v_n|\right\}$$

Metric: a measure of distance

- A **metric** is a general measure of the distance $d(\mathbf{u}, \mathbf{v})$ between points $\mathbf{u}$ and $\mathbf{v}$, which satisfies the following **axioms**:
  - $d(\mathbf{u}, \mathbf{v}) = d(\mathbf{v}, \mathbf{u})$
  - $d(\mathbf{u}, \mathbf{v}) > 0$ for $\mathbf{u} \neq \mathbf{v}$
  - $d(\mathbf{u}, \mathbf{u}) = 0$
  - $d(\mathbf{u}, \mathbf{w}) \leq d(\mathbf{u}, \mathbf{v}) + d(\mathbf{v}, \mathbf{w})$ (**triangle inequality**)
- Metrics form a very broad class of distance measures, some of which do not fit in well with our geometric intuitions
- E.g., metric need not be **translation-invariant**

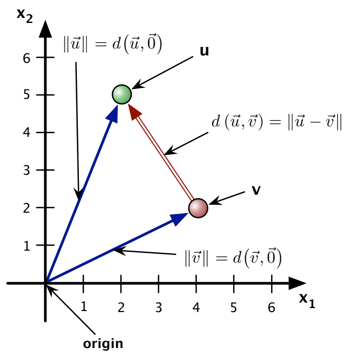
$$d(\mathbf{u} + \mathbf{x}, \mathbf{v} + \mathbf{x}) \neq d(\mathbf{u}, \mathbf{v})$$

- Another unintuitive example is the **discrete metric**

$$d(\mathbf{u}, \mathbf{v}) = \begin{cases} 0 & \mathbf{u} = \mathbf{v} \\ 1 & \mathbf{u} \neq \mathbf{v} \end{cases}$$

Distance vs. norm

- Intuitively, **distance** $d(\mathbf{u}, \mathbf{v})$ should correspond to **length** $\|\mathbf{u} - \mathbf{v}\|$ of displacement vector $\mathbf{u} - \mathbf{v}$
  - $d(\mathbf{u}, \mathbf{v})$ is a **metric**
  - $\|\mathbf{u} - \mathbf{v}\|$ is a **norm**
  - $\|\mathbf{u}\| = d(\mathbf{u}, \mathbf{0})$
- Such a metric is always **translation-invariant**



- $d_p(\mathbf{u}, \mathbf{v}) = \|\mathbf{v} - \mathbf{u}\|_p$
- **Minkowski *p*-norm** for $p \in [1, \infty]$:

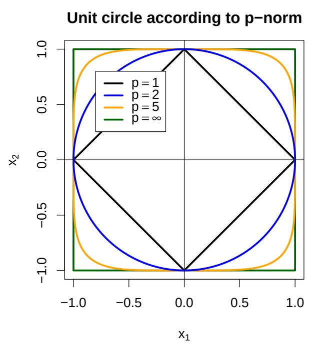
$$\|\mathbf{u}\|_p := \left(|u_1|^p + \cdots + |u_n|^p\right)^{1/p}$$

Norm: a measure of length

- A general **norm** $\|\mathbf{u}\|$ for the length of a vector $\mathbf{u}$ must satisfy the following **axioms**:
  - $\|\mathbf{u}\| > 0$ for $\mathbf{u} \neq \mathbf{0}$
  - $\|\lambda\mathbf{u}\| = |\lambda| \cdot \|\mathbf{u}\|$ (**homogeneity**, not req'd for metric)
  - $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$ (**triangle inequality**)
- every norm defines a translation-invariant metric

$$d(\mathbf{u}, \mathbf{v}) := \|\mathbf{u} - \mathbf{v}\|$$

Norm: a measure of length



- Visualisation of norms in $\mathbb{R}^2$ by plotting **unit circle** for each norm, i.e. points $\mathbf{u}$ with $\|\mathbf{u}\| = 1$
- Here: *p*-norms $\|\cdot\|_p$ for different values of p
- Triangle inequality $\iff$ unit circle is **convex**
- This shows that *p*-norms with $p < 1$ would violate the triangle inequality
- Consequence for DSM: $p \gg 2$ "favours" small differences in many coordinates, $p \ll 2$ differences in few coordinates

Operator and matrix norm

- The **norm** of a linear map (or "operator") $f : U \to V$ between normed vector spaces U and V is defined as

$$\|f\| := \max\left\{\|f(\mathbf{u})\| \mid \mathbf{u} \in U, \|\mathbf{u}\| = 1\right\}$$

  - $\|f\|$ depends on the norms chosen in U and V!
- The definition of the operator norm implies

$$\|f(\mathbf{u})\| \leq \|f\| \cdot \|\mathbf{u}\|$$

- Norm of a matrix $\mathbf{A}$ = norm of corresponding map f
  - NB: this is not the same as a *p*-norm of $\mathbf{A}$ in $\mathbb{R}^{k \cdot n}$
  - **spectral norm** induced by Euclidean vector norms in U and V = largest **singular value** of $\mathbf{A}$ (➜ SVD)

Which metric should I use?

- Choice of metric or norm is one of the parameters of a DSM
- Measures of **distance** between points:
  - intuitive Euclidean norm $\|\cdot\|_2$
  - "city-block" Manhattan distance $\|\cdot\|_1$
  - maximum distance $\|\cdot\|_\infty$
  - general Minkowski p-norm $\|\cdot\|_p$
  - and many other formulae ...
- Measures of the **similarity** of arrows:
  - "cosine distance" $\sim u_1v_1 + \cdots + u_nv_n$
  - Dice coefficient (matching non-zero coordinates)
  - and, of course, many other formulae ...
  - ☞ these measures determine **angles** between arrows
- Similarity and distance measures are equivalent!
  - ☞ I'm a fan of the Euclidean norm because of its intuitive geometric properties (angles, orthogonality, shortest path, ...)

Norms & distance measures in R

```
# We will use the cooccurrence matrix M from the last session
> print(M)
      eat get hear kill see use
boat    0  59    4    0  39  23
cat     6  52    4   26  58   4
cup     1  98    2    0  14   6
dog    33 115   42   17  83  10
knife   3  51    0    0  20  84
pig     9  12    2   27  17   3

# Note: you can save selected variables with the save() command,
# and restore them in your next session (similar to saving R's workspace)
> save(M, O, E, M.mds, file="dsm_lab.RData")

# load() restores the variables under the same names!
> load("dsm_lab.RData")
```


Norms & distance measures in R

```
# Define functions for general Minkowski norm and distance;
# parameter p is optional and defaults to p = 2
> p.norm <- function (x, p=2) (sum(abs(x)^p))^(1/p)
> p.dist <- function (x, y, p=2) p.norm(x - y, p)

> round(apply(M, 1, p.norm, p=1), 2)
 boat   cat   cup   dog knife   pig
  125   150   121   300   158    70
> round(apply(M, 1, p.norm, p=2), 2)
  boat    cat    cup    dog  knife    pig
 74.48  82.53  99.20 152.83 100.33  35.44
> round(apply(M, 1, p.norm, p=4), 2)
  boat    cat    cup    dog  knife    pig
 61.93  66.10  98.01 122.71  86.78  28.31
> round(apply(M, 1, p.norm, p=99), 2)
 boat   cat   cup   dog knife   pig
   59    58    98   115    84    27
```


Norms & distance measures in R

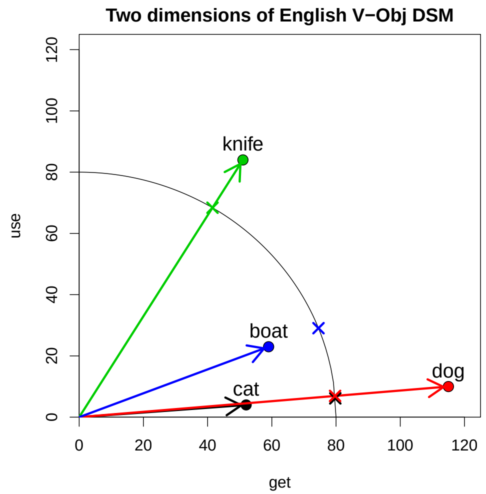
```
# Here's a nice trick to normalise the row vectors quickly
> normalise <- function (M, p=2) M / apply(M, 1, p.norm, p=p)

# dist() function also supports Minkowski p-metric
# (must normalise rows in order to compare different metrics)
> round(dist(normalise(M, p=1), method="minkowski", p=1), 2)
      boat  cat  cup  dog knife
cat   0.58
cup   0.69 0.97
dog   0.55 0.45 0.89
knife 0.73 1.01 1.01 1.00
pig   1.03 0.64 1.29 0.71  1.28

# Try different p-norms: how do the distances change?
> round(dist(normalise(M, p=2), method="minkowski", p=2), 2)
> round(dist(normalise(M, p=4), method="minkowski", p=4), 2)
> round(dist(normalise(M, p=99), method="minkowski", p=99), 2)
```

Why it is important to normalise vectors

before computing a distance matrix



Euclidean norm & inner product

- The Euclidean norm $\|\mathbf{u}\|_2 = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle}$ is special because it can be derived from the **inner product**:

$$\langle \mathbf{u}, \mathbf{v} \rangle := \mathbf{x}^T \mathbf{y} = x_1 y_1 + \cdots + x_n y_n$$

  where $\mathbf{u} \equiv_E \mathbf{x}$ and $\mathbf{v} \equiv_E \mathbf{y}$ are the standard coordinates of $\mathbf{u}$ and $\mathbf{v}$ (certain other coordinate systems also work)

- The inner product is a **positive definite** and **symmetric bilinear form** with the following properties:
  - $\langle \lambda \mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{u}, \lambda \mathbf{v} \rangle = \lambda \langle \mathbf{u}, \mathbf{v} \rangle$
  - $\langle \mathbf{u} + \mathbf{u}', \mathbf{v} \rangle = \langle \mathbf{u}, \mathbf{v} \rangle + \langle \mathbf{u}', \mathbf{v} \rangle$
  - $\langle \mathbf{u}, \mathbf{v} + \mathbf{v}' \rangle = \langle \mathbf{u}, \mathbf{v} \rangle + \langle \mathbf{u}, \mathbf{v}' \rangle$
  - $\langle \mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{v}, \mathbf{u} \rangle$ (**symmetric**)
  - $\langle \mathbf{u}, \mathbf{u} \rangle = \|\mathbf{u}\|^2 > 0$ for $\mathbf{u} \neq \mathbf{0}$ (**positive definite**)
  - also called **dot product** or **scalar product**

Angles and orthogonality

- The Euclidean inner product has an important **geometric** interpretation ➜ angles and orthogonality
- **Cauchy-Schwarz inequality**:

$$\left| \langle \mathbf{u}, \mathbf{v} \rangle \right| \leq \|\mathbf{u}\| \cdot \|\mathbf{v}\|$$

- **Angle** ϕ between vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$:

$$\cos \phi := \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{u}\| \cdot \|\mathbf{v}\|}$$

  - $\cos \phi$ is the "cosine similarity" measure
- $\mathbf{u}$ and $\mathbf{v}$ are **orthogonal** iff $\langle \mathbf{u}, \mathbf{v} \rangle = 0$
  - the **shortest connection** between a point $\mathbf{u}$ and a subspace U is orthogonal to all vectors $\mathbf{v} \in U$

Cosine similarity in R

The `dist()` function does not calculate the cosine measure (because it is a similarity rather than distance value), but:

$$\mathbf{M} \cdot \mathbf{M}^T = \begin{bmatrix} \cdots & \mathbf{u}^{(1)} & \cdots \\ \cdots & \mathbf{u}^{(2)} & \cdots \\ & & \\ \cdots & \mathbf{u}^{(n)} & \cdots \end{bmatrix} \cdot \begin{bmatrix} \vdots & \vdots & & \vdots \\ \mathbf{u}^{(1)} & \mathbf{u}^{(2)} & & \mathbf{u}^{(n)} \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

$$\Rightarrow \quad \left(\mathbf{M} \cdot \mathbf{M}^T \right)_{ij} = \left\langle \mathbf{u}^{(i)}, \mathbf{u}^{(j)} \right\rangle$$

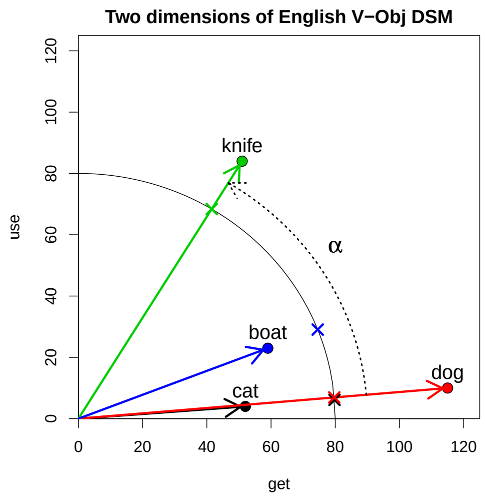
```
# Matrix of cosine similarities between rows of M:
> M.norm <- normalise(M, p=2)   # only works with Euclidean norm!
> M.norm %*% t(M.norm)
```

Euclidean distance or cosine similarity?

- Which is better, Euclidean distance or cosine similarity?
- They are equivalent: if vectors are normalised ($\|\mathbf{u}\|_2 = 1$), both lead to the same neighbour ranking

$$
\begin{aligned}
d_2(\mathbf{u}, \mathbf{v}) &= \sqrt{\|\mathbf{u} - \mathbf{v}\|_2} = \sqrt{\langle \mathbf{u} - \mathbf{v}, \mathbf{u} - \mathbf{v} \rangle} \\
&= \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle + \langle \mathbf{v}, \mathbf{v} \rangle - 2 \langle \mathbf{u}, \mathbf{v} \rangle} \\
&= \sqrt{\|\mathbf{u}\|_2 + \|\mathbf{v}\|_2 - 2 \langle \mathbf{u}, \mathbf{v} \rangle} \\
&= \sqrt{2 - 2 \cos \phi}
\end{aligned}
$$

Euclidean distance and cosine similarity



Cartesian coordinates

- A set of vectors $\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(n)}$ is called **orthonormal** if the vectors are pairwise orthogonal and of unit length:
  - $\langle \mathbf{b}^{(j)}, \mathbf{b}^{(k)} \rangle = 0$ for $j \neq k$
  - $\langle \mathbf{b}^{(k)}, \mathbf{b}^{(k)} \rangle = \|\mathbf{b}^{(k)}\|^2 = 1$
- An orthonormal basis and the corresponding coordinates are called **Cartesian**
- Cartesian coordinates are particularly intuitive, and the inner product has the same form wrt. every Cartesian basis B: for $\mathbf{u} \equiv_B \mathbf{x}'$ and $\mathbf{v} \equiv_B \mathbf{y}'$, we have

$$
\langle \mathbf{u}, \mathbf{v} \rangle = (\mathbf{x}')^T \mathbf{y}' = x'_1 y'_1 + \cdots + x'_n y'_n
$$

- NB: the column vectors of the matrix $\mathbf{B}$ are orthonormal
  - recall that the columns of $\mathbf{B}$ specify the standard coordinates of the vectors $\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(n)}$

Orthogonal projection

- Cartesian coordinates $\mathbf{u} \equiv_B \mathbf{x}$ can easily be computed:

$$
\begin{aligned}
\left\langle \mathbf{u}, \mathbf{b}^{(k)} \right\rangle &= \left\langle \sum_{j=1}^{n} x_j \mathbf{b}^{(j)}, \mathbf{b}^{(k)} \right\rangle \\
&= \sum_{j=1}^{n} x_j \underbrace{\left\langle \mathbf{b}^{(j)}, \mathbf{b}^{(k)} \right\rangle}_{=\delta_{jk}} = x_k
\end{aligned}
$$

  - Kronecker delta: $\delta_{jk} = 1$ for $j = k$ and 0 for $j \neq k$

- **Orthogonal projection** $P_V : \mathbb{R}^n \to V$ to subspace $V := \mathrm{sp}\left(\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(k)}\right)$ (for $k < n$) is given by

$$
P_V \mathbf{u} := \sum_{j=1}^{k} \mathbf{b}^{(j)} \left\langle \mathbf{u}, \mathbf{b}^{(j)} \right\rangle
$$

Hyperplanes & normal vectors

A hyperplane is the decision boundary of a linear classifier!

- A hyperplane $U \subseteq \mathbb{R}^n$ through the origin $\mathbf{0}$ can be characterized by the equation

$$U = \{\mathbf{u} \in \mathbb{R}^n \mid \langle \mathbf{u}, \mathbf{n} \rangle = 0\}$$

  for a suitable $\mathbf{n} \in \mathbb{R}^n$ with $\|\mathbf{n}\| = 1$
- $\mathbf{n}$ is called the **normal vector** of U
- The orthogonal projection P_U into U is given by

$$P_U \mathbf{v} := \mathbf{v} - \mathbf{n} \langle \mathbf{v}, \mathbf{n} \rangle$$

- An arbitrary hyperplane $\Gamma \subseteq \mathbb{R}^n$ can analogously be characterized by

$$\Gamma = \{\mathbf{u} \in \mathbb{R}^n \mid \langle \mathbf{u}, \mathbf{n} \rangle = a\}$$

  where $a \in \mathbb{R}$ is the (signed) **distance** of Γ from $\mathbf{0}$

Orthogonal matrices

- A matrix $\mathbf{A}$ whose column vectors are orthonormal is called an **orthogonal** matrix
- $\mathbf{A}^T$ is orthogonal iff $\mathbf{A}$ is orthogonal
- The **inverse** of an orthogonal matrix is simply its transpose:

$$\mathbf{A}^{-1} = \mathbf{A}^T \quad \text{if } \mathbf{A} \text{ is orthogonal}$$

  - it is easy to show $\mathbf{A}^T\mathbf{A} = \mathbf{I}$ by matrix multiplication, since the columns of $\mathbf{A}$ are orthonormal
  - since $\mathbf{A}^T$ is also orthogonal, it follows that $\mathbf{A}\mathbf{A}^T = (\mathbf{A}^T)^T\mathbf{A}^T = \mathbf{I}$
  - side remark: the transposition operator $\cdot^T$ is called an **involution** because $(\mathbf{A}^T)^T = \mathbf{A}$

Isometric maps

- An endomorphism $f : \mathbb{R}^n \to \mathbb{R}^n$ is called an **isometry** iff $\langle f(\mathbf{u}), f(\mathbf{v}) \rangle = \langle \mathbf{u}, \mathbf{v} \rangle$ for all $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$
- Geometric interpretation: isometries preserve angles and distances (which are defined in terms of $\langle \cdot, \cdot \rangle$)
- f is an isometry iff its matrix $\mathbf{A}$ is orthogonal
- Coordinate transformations between Cartesian systems are isometric (because $\mathbf{B}$ and $\mathbf{B}^{-1} = \mathbf{B}^T$ are orthogonal)
- Every isometric endomorphism of $\mathbb{R}^n$ can be written as a combination of **planar rotations** and **axial reflections** in a suitable Cartesian coordinate system

$$R_\phi^{(1,3)} = \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix}, \quad Q^{(2)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Summary: orthogonal matrices

- The column vectors of an orthogonal $n \times n$ matrix $\mathbf{B}$ form a Cartesian basis $\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(n)}$ of $\mathbb{R}^n$
- $\mathbf{B}^{-1} = \mathbf{B}^T$, i.e. we have $\mathbf{B}^T\mathbf{B} = \mathbf{B}\mathbf{B}^T = \mathbf{I}$
- The coordinate transformation $\mathbf{B}^T$ into B-coordinates is an isometry, i.e. all distances and angles are preserved
- The first $k < n$ columns of $\mathbf{B}$ form a Cartesian basis of a subspace $V = \text{sp}\left(\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(k)}\right)$ of $\mathbb{R}^n$
- The corresponding rectangular matrix $\hat{\mathbf{B}} = \left[\mathbf{b}^{(1)}, \ldots, \mathbf{b}^{(k)}\right]$ performs an orthogonal projection into V:

$$\begin{aligned} P_V\mathbf{u} &\equiv_B \hat{\mathbf{B}}^T\mathbf{x} \qquad (\text{for } \mathbf{u} \equiv_E \mathbf{x}) \\ &\equiv_E \hat{\mathbf{B}}\hat{\mathbf{B}}^T\mathbf{x} \end{aligned}$$

➥ These properties will become important later today!

General inner products

- Can we also introduce geometric notions such as angles and orthogonality for other metrics, e.g. the Manhattan distance?
  - ☞ norm must be derived from appropriate inner product
- General **inner products** are defined by

$$\langle \mathbf{u}, \mathbf{v} \rangle_B := (\mathbf{x}')^T \mathbf{y}' = x'_1 y'_1 + \cdots + x'_y y'_n$$

  wrt. non-Cartesian basis B ($\mathbf{u} \equiv_B \mathbf{x}'$, $\mathbf{v} \equiv_B \mathbf{y}'$)

- $\langle \cdot, \cdot \rangle_B$ can be expressed in standard coordinates $\mathbf{u} \equiv_E \mathbf{x}$, $\mathbf{v} \equiv_E \mathbf{y}$ using the transformation matrix $\mathbf{B}$:

$$\begin{aligned} \langle \mathbf{u}, \mathbf{v} \rangle_B &= (\mathbf{x}')^T \mathbf{y}' = \left(\mathbf{B}^{-1}\mathbf{x}\right)^T \left(\mathbf{B}^{-1}\mathbf{y}\right) \\ &= \mathbf{x}^T (\mathbf{B}^{-1})^T \mathbf{B}^{-1} \mathbf{y} =: \mathbf{x}^T \mathbf{C} \mathbf{y} \end{aligned}$$

General inner products

- The coefficient matrix $\mathbf{C} := (\mathbf{B}^{-1})^T \mathbf{B}^{-1}$ of the general inner product is **symmetric**

$$\mathbf{C}^T = (\mathbf{B}^{-1})^T ((\mathbf{B}^{-1})^T)^T = (\mathbf{B}^{-1})^T \mathbf{B}^{-1} = \mathbf{C}$$

  and **positive definite**

$$\mathbf{x}^T \mathbf{C} \mathbf{x} = \left(\mathbf{B}^{-1}\mathbf{x}\right)^T \left(\mathbf{B}^{-1}\mathbf{x}\right) = (\mathbf{x}')^T \mathbf{x}' \geq 0$$

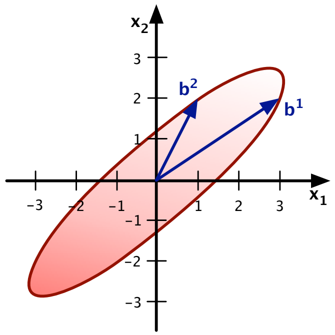
- It is (relatively) easy to show that every positive definite and symmetric bilinear form can be written in this way.
  - ☞ i.e. every norm that is derived from an inner product can be expressed in terms of a coefficient matrix $\mathbf{C}$ or basis B

General inner products

An example:

- $\mathbf{b}^{(1)} = (3, 2)$, $\mathbf{b}^{(2)} = (1, 2)$
- $\mathbf{B} = \begin{bmatrix} 3 & 1 \\ 2 & 2 \end{bmatrix}$
- $\mathbf{B}^{-1} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{4} \\ -\frac{1}{2} & \frac{3}{4} \end{bmatrix}$
- $\mathbf{C} = \begin{bmatrix} .5 & -.5 \\ -.5 & .625 \end{bmatrix}$
- Graph shows **unit circle** of the inner product $\mathbf{C}$, i.e. points $\mathbf{x}$ with

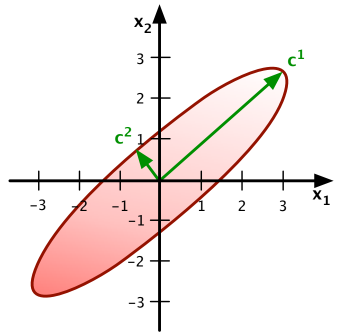
$$\mathbf{x}^T \mathbf{C} \mathbf{x} = 1$$



General inner products

- $\mathbf{C}$ is a symmetric matrix
- There is always an orthonormal basis such that $\mathbf{C}$ has diagonal form
- "Standard" dot product with additional scaling factors (wrt. this orthonormal basis)
- Intuition: unit circle is a squashed and rotated disk

➥ Every "geometric" norm is equivalent to the Euclidean norm except for a rotation and rescaling of the axes

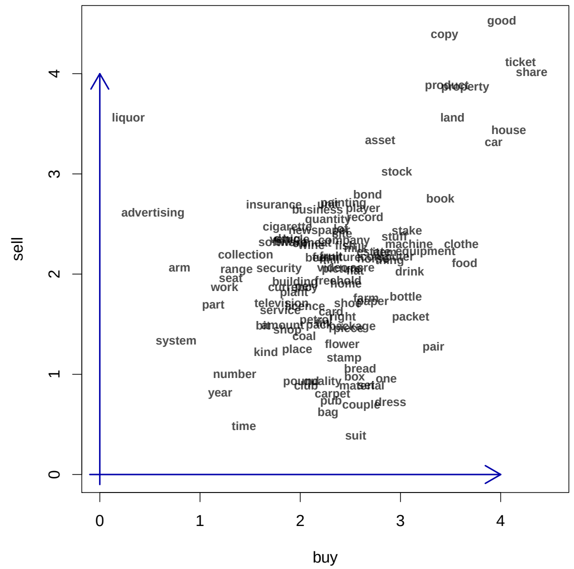


Motivating latent dimensions: example data

- Example: term-term matrix
- V-Obj cooc's extracted from BNC
  - targets = noun lemmas
  - features = verb lemmas
- feature scaling: association scores (modified log Dice coefficient)
- $k = 111$ nouns with $f \geq 20$ (must have non-zero row vectors)
- $n = 2$ dimensions: *buy* and *sell*

noun	*buy*	*sell*
bond	0.28	0.77
cigarette	-0.52	0.44
dress	0.51	-1.30
freehold	-0.01	-0.08
land	1.13	1.54
number	-1.05	-1.02
per	-0.35	-0.16
pub	-0.08	-1.30
share	1.92	1.99
system	-1.63	-0.70

Motivating latent dimensions & subspace projection

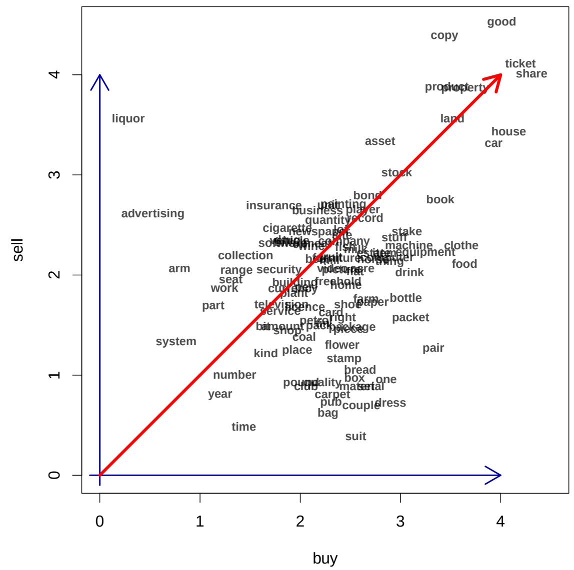


Motivating latent dimensions & subspace projection

- The **latent property** of being a commodity is "expressed" through associations with several verbs: *sell*, *buy*, *acquire*, ...
- Consequence: these DSM dimensions will be **correlated**

- Identify **latent dimension** by looking for strong correlations (or weaker correlations between large sets of features)
- Projection into subspace V of $k < n$ latent dimensions as a "**noise reduction**" technique ➜ **LSA**
- Assumptions of this approach:
  - "latent" distances in V are semantically meaningful
  - other "residual" dimensions represent chance co-occurrence patterns, often particular to the corpus underlying the DSM

The latent "commodity" dimension



The variance of a data set

- Rationale: find the dimensions that give the best (statistical) explanation for the **variance** (or "spread") of the data
- Definition of the variance of a set of vectors
  - ☞ you remember the equations for one-dimensional data, right?

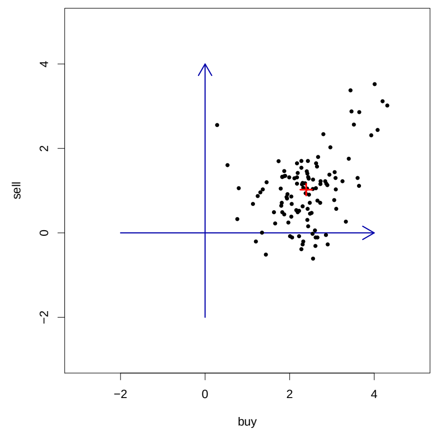
$$\sigma^2 = \frac{1}{k-1}\sum_{i=1}^{k}\|\mathbf{x}^{(i)} - \boldsymbol{\mu}\|^2$$

$$\boldsymbol{\mu} = \frac{1}{k}\sum_{i=1}^{k}\mathbf{x}^{(i)}$$

- Easier to calculate if we **center** the data so that $\boldsymbol{\mu} = \mathbf{0}$

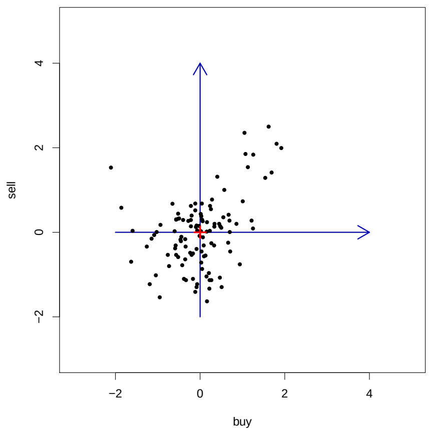
Centering the data set

- **Uncentered data set**
- Centered data set
- Variance of centered data



Centering the data set

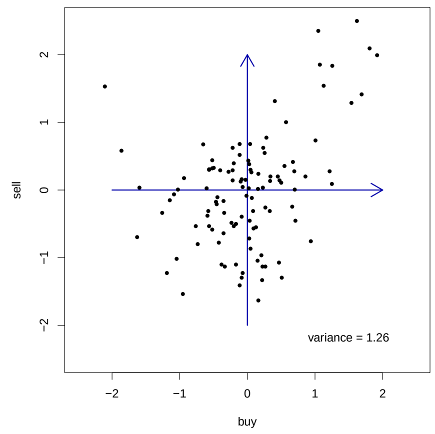
- Uncentered data set
- **Centered data set**
- Variance of centered data



Centering the data set

- Uncentered data set
- Centered data set
- **Variance of centered data**

$$\sigma^2 = \frac{1}{k-1}\sum_{i=1}^{k}\|\mathbf{x}^{(i)}\|^2$$



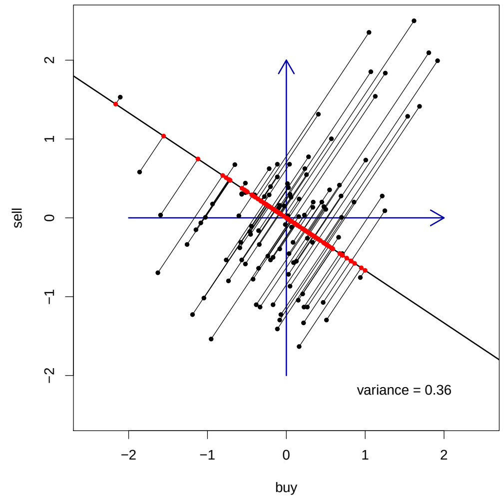
Principal components analysis (PCA)

- We want to project the data points to a lower-dimensional subspace, but preserve distances as well as possible
- Insight 1: variance = average squared distance

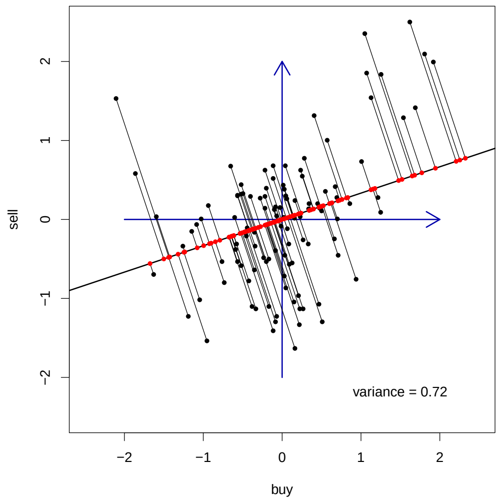
$$\frac{1}{k(k-1)} \sum_{i=1}^{k} \sum_{j=1}^{k} \|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|^2 = \frac{2}{k-1} \sum_{i=1}^{k} \|\mathbf{x}^{(i)}\|^2 = 2\sigma^2$$

- Insight 2: orthogonal projection always reduces distances
  ➜ difference in squared distances = loss of variance
- If we reduced the data set to just a single dimension, which dimension would still have the highest variance?
- Mathematically, we project the points onto a line through the origin and calculate one-dimensional variance on this line
  - we'll see in a moment how to compute such projections
  - but first, let us look at a few examples

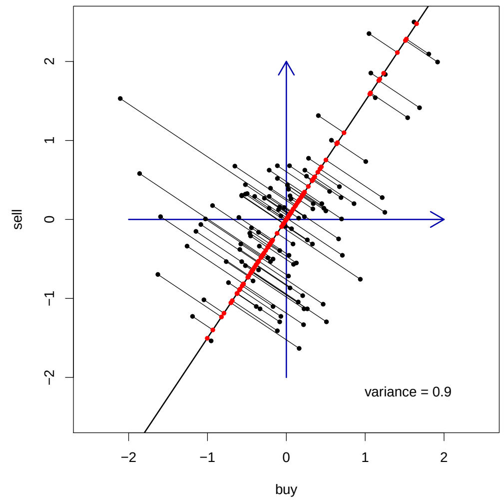
Projection and preserved variance: examples



Projection and preserved variance: examples

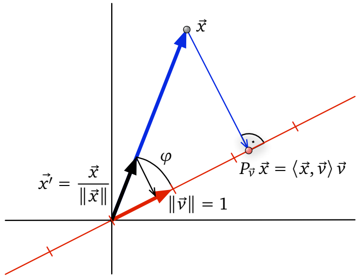


Projection and preserved variance: examples



The mathematics of projections

- Line through origin given by unit vector $\|\mathbf{v}\| = 1$
- For a point $\mathbf{x}$ and the corresponding unit vector $\mathbf{x}' = \mathbf{x}/\|\mathbf{x}\|$, we have $\cos\varphi = \langle \mathbf{x}', \mathbf{v} \rangle$



- Trigonometry: position of projected point on the line is $\|\mathbf{x}\| \cdot \cos\varphi = \|\mathbf{x}\| \cdot \langle \mathbf{x}', \mathbf{v} \rangle = \langle \mathbf{x}, \mathbf{v} \rangle$
- Preserved variance = one-dimensional variance on the line (note that data set is still centered after projection)

$$\sigma_{\mathbf{v}}^2 = \frac{1}{k-1} \sum_{i=1}^{k} \langle \mathbf{x}_i, \mathbf{v} \rangle^2$$

The covariance matrix

- Find the direction $\mathbf{v}$ with maximal $\sigma_{\mathbf{v}}^2$, which is given by:

$$\begin{aligned}
\sigma_{\mathbf{v}}^2 &= \tfrac{1}{k-1} \sum_{i=1}^{k} \langle \mathbf{x}_i, \mathbf{v} \rangle^2 \\
&= \tfrac{1}{k-1} \sum_{i=1}^{k} \left(\mathbf{x}_i^T \mathbf{v}\right)^T \cdot \left(\mathbf{x}_i^T \mathbf{v}\right) \\
&= \tfrac{1}{k-1} \sum_{i=1}^{k} \mathbf{v}^T \left(\mathbf{x}_i \mathbf{x}_i^T\right) \mathbf{v} \\
&= \mathbf{v}^T \left(\tfrac{1}{k-1} \sum_{i=1}^{k} \mathbf{x}_i \mathbf{x}_i^T \right) \mathbf{v} \\
&= \mathbf{v}^T \mathbf{C} \mathbf{v}
\end{aligned}$$

The covariance matrix

- $\mathbf{C}$ is the **covariance matrix** of the data points
  - $\mathbf{C}$ is a square $n \times n$ matrix (2×2 in our example)
- Preserved variance after projection onto a line $\mathbf{v}$ can easily be calculated as $\sigma_{\mathbf{v}}^2 = \mathbf{v}^T \mathbf{C} \mathbf{v}$
- The original variance of the data set is given by $\sigma^2 = \operatorname{tr}(\mathbf{C}) = C_{11} + C_{22} + \cdots + C_{nn}$

$$\mathbf{C} = \begin{pmatrix} \sigma_1^2 & C_{12} & \cdots & C_{1n} \\ C_{21} & \sigma_2^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & C_{n-1,n} \\ C_{n1} & \cdots & C_{n,n-1} & \sigma_n^2 \end{pmatrix}$$

Maximizing preserved variance

- In our example, we want to find the axis $\mathbf{v}_1$ that preserves the largest amount of variance by maximizing $\mathbf{v}_1^T \mathbf{C} \mathbf{v}_1$
- For higher-dimensional data set, we also want to find the axis $\mathbf{v}_2$ with the second largest amount of variance, etc.
  - ☞ Should not include variance that has already been accounted for: $\mathbf{v}_2$ must be orthogonal to $\mathbf{v}_1$, i.e. $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$
- Orthogonal dimensions $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots$ **partition** variance:

$$\sigma^2 = \sigma_{\mathbf{v}^{(1)}}^2 + \sigma_{\mathbf{v}^{(2)}}^2 + \ldots$$

- Useful result from linear algebra: every symmetric matrix $\mathbf{C} = \mathbf{C}^T$ has an **eigenvalue decomposition** with orthogonal **eigenvectors** $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$ and corresponding **eigenvalues** $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$

Eigenvalue decomposition

- The eigenvalue decomposition of $\mathbf{C}$ can be written in the form

$$\mathbf{C} = \mathbf{U} \cdot \mathbf{D} \cdot \mathbf{U}^T$$

where $\mathbf{U}$ is an orthogonal matrix of eigenvectors (columns) and $\mathbf{D} = \text{Diag}(\lambda_1, \dots, \lambda_n)$ a diagonal matrix of eigenvalues

$$\mathbf{U} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} \lambda_1 & & & & \\ & \lambda_2 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \lambda_n \end{bmatrix}$$

  - note that both $\mathbf{U}$ and $\mathbf{D}$ are $n \times n$ square matrices

The PCA algorithm

- With the eigenvalue decomposition of $\mathbf{C}$, we have

$$\sigma_{\mathbf{v}}^2 = \mathbf{v}^T\mathbf{C}\mathbf{v} = \mathbf{v}^T\mathbf{U}\mathbf{D}\mathbf{U}^T\mathbf{v} = (\mathbf{U}^T\mathbf{v})^T\mathbf{D}(\mathbf{U}^T\mathbf{v}) = \mathbf{y}^T\mathbf{D}\mathbf{y}$$

where $\mathbf{y} = \mathbf{U}^T\mathbf{v} = [y_1, y_2, \dots, y_n]^T$ are the coordinates of $\mathbf{v}$ in the Cartesian basis formed by the eigenvectors of $\mathbf{C}$
- $\|\mathbf{y}\| = 1$ since $\mathbf{U}^T$ is an isometry (orthogonal matrix)
- We therefore want to maximize

$$\mathbf{v}^T\mathbf{C}\mathbf{v} = \lambda_1(y_1)^2 + \lambda_2(y_2)^2 \dots + \lambda_n(y_n)^2$$

under the constraint $(y_1)^2 + (y_2)^2 + \dots + (y_n)^2 = 1$
- Solution: $\mathbf{y} = [1, 0, \dots, 0]^T$ (since λ_1 is the largest eigenvalue)
- This corresponds to $\mathbf{v} = \mathbf{a}_1$ (the first eigenvector of $\mathbf{C}$) and a preserved amount of variance given by $\sigma_{\mathbf{v}}^2 = \mathbf{a}_1^T\mathbf{C}\mathbf{a}_1 = \lambda_1$

The PCA algorithm

- In order to find the dimension of second highest variance, we have to look for an axis $\mathbf{v}$ orthogonal to $\mathbf{a}_1$
  - ☞ $\mathbf{U}^T$ is orthogonal, so the coordinates $\mathbf{y} = \mathbf{U}^T\mathbf{v}$ must be orthogonal to first axis $[1, 0, \dots, 0]^T$, i.e. $\mathbf{y} = [0, y_2, \dots, y_n]^T$
- In other words, we have to maximize

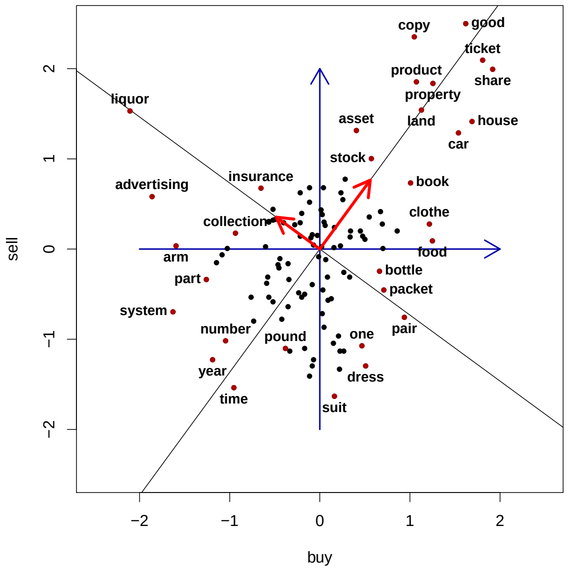
$$\mathbf{v}^T\mathbf{C}\mathbf{v} = \lambda_2(y_2)^2 \dots + \lambda_n(y_n)^2$$

under constraints $y_1 = 0$ and $(y_2)^2 + \dots + (y_n)^2 = 1$
- Again, solution is $\mathbf{y} = [0, 1, 0, \dots, 0]^T$, corresponding to the second eigenvector $\mathbf{v} = \mathbf{a}_2$ and preserved variance $\sigma_{\mathbf{v}}^2 = \lambda_2$
- Similarly for the third, fourth, ... axis

The PCA algorithm

- The eigenvectors $\mathbf{a}_i$ of the covariance matrix $\mathbf{C}$ are called the **principal components** of the data set
- The amount of variance preserved (or "explained") by the i-th principal component is given by the eigenvalue λ_i
- Since $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$, the first principal component accounts for the largest amount of variance etc.
- Coordinates of a point $\mathbf{x}$ in PCA space are given by $\mathbf{U}^T\mathbf{x}$ (note: these are the projections on the principal components)
- For the purpose of "noise reduction", only the first $n' < n$ principal components (with highest variance) are retained, and the other dimensions in PCA space are dropped
  - ☞ i.e. data points are projected into the subspace V spanned by the first n' column vectors of $\mathbf{U}$

PCA example



PCA in R

```
> pca <- prcomp(M)   # for the buy/sell example data

> summary(pca)
Importance of components:
                         PC1   PC2
Standard deviation     0.947 0.599
Proportion of Variance 0.715 0.285
Cumulative Proportion  0.715 1.000

> print(pca)
Standard deviations:
[1] 0.9471326 0.5986067

Rotation:
            PC1        PC2
buy  -0.5907416  0.8068608
sell -0.8068608 -0.5907416
```


PCA in R

```
# Coordinates in PCA space
> pca$x[c("house","book","arm","time"), ]
             PC1        PC2
house -2.1390957  0.5274687
book  -1.1864783  0.3797070
arm    0.9141092 -1.3080504
time   1.8036445  0.1387165

# Transformation matrix U
> pca$rotation
            PC1        PC2
buy  -0.5907416  0.8068608
sell -0.8068608 -0.5907416

# Eigenvalues of the covariance matrix C
> (pca$sdev)^2
[1] 0.8970602 0.3583299
```