Introduction to Probabilistic Machine Learning

Ralf Herbrich, Rainer Schlosser

Linear Basis Function Models

Overview

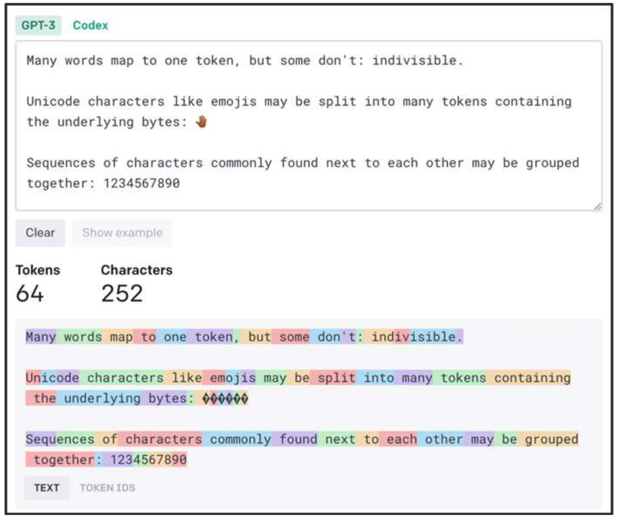
1. Modelling Data
   - Modelling Text
   - Modelling Images
2. Linear Basis Function Models
   - Vector Spaces
   - Linear Mappings and Matrices

Overview

1. **Modelling Data**
   - **Modelling Text**
   - Modelling Images
2. Linear Basis Function Models
   - Vector Spaces
   - Linear Mappings and Matrices

Modelling Text

- Text can be modelled at three levels of granularity:
  1. **Letters** ($\approx 10^2$ different letters in most alphabets)
  2. **Tokens** (10^3 to 10^4 different tokens in most alphabets)
  3. **Words** (10^5 to 10^6 different words in most languages)
- Modelling level of granularity depends on the application
  1. **Letters**: Compression algorithms for textual data
  2. **Tokens**: Sequence prediction for question-answering
  3. **Words**: Auto-correct function in smart keyboards



One-Hot Encoding

- Text is a sequence of text elements $s_1, s_2, s_3, \ldots$
- We often want to know the importance of each text element s_i to the target
  - **Example.** "This product shipped fast but was disappointing" has negative sentiment

    s_1 s_2 s_3 s_4 s_5 s_6 s_7

    Text element that is likely the expression of bad sentiment
- **One-Hot Encoding**. *Given a dictionary S and a text element $s \in S$, a one-hot encoding $\boldsymbol{\phi}_{\mathrm{OHE}}(s)$ is an $|S|$-dimensional unit vector indexed by the elements of S that consists of 0s in all dimensions with the exception of a single 1 in the dimension indexed by s.*
  - **Example (ctd)**. If we assume the indices are $s_1, s_2, \ldots, s_7$ then

$$\boldsymbol{\phi}_{\mathrm{OHE}}(\text{"fast"}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{array}{l} \leftarrow \text{This} \\ \leftarrow \text{product} \\ \leftarrow \text{shipped} \\ \leftarrow \text{fast} \\ \leftarrow \text{but} \\ \leftarrow \text{was} \\ \leftarrow \text{disappointing} \end{array}$$

Modelling Whole Text

- Two common techniques for encoding a whole text
  - **Variable-length text** $s_1 s_2 \cdots$: Sum of all one-hot encoded vectors ("bag of words")

$$\boldsymbol{\phi}_{\mathrm{BOW}}(s_1 s_2 \cdots) = \sum_j \boldsymbol{\phi}_{\mathrm{OHE}}(s_j)$$

  - **Fixed-length text** $s_1 s_2 \cdots s_n$: A stacked $n \cdot |S|$ dimensional vector

$$\boldsymbol{\phi}_{\mathrm{FL}}(s_1 s_2 \cdots s_n) = \begin{bmatrix} \boldsymbol{\phi}_{\mathrm{OHE}}(s_1) \\ \vdots \\ \boldsymbol{\phi}_{\mathrm{OHE}}(s_n) \end{bmatrix}$$

- **Example**: Consider the sentences
  - $\boldsymbol{s}_1$ = "This product shipped not fast and was disappointing"
  - $\boldsymbol{s}_2$ = "This product shipped fast and was not disappointing"

$$\boldsymbol{\phi}_{\mathrm{BOW}}(\boldsymbol{s}_1) = \begin{array}{c} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \end{array} \begin{array}{l} \leftarrow \text{This} \\ \leftarrow \text{product} \\ \leftarrow \text{shipped} \\ \leftarrow \text{not} \\ \leftarrow \text{fast} \\ \leftarrow \text{and} \\ \leftarrow \text{was} \\ \leftarrow \text{disappointing} \end{array} \qquad \boldsymbol{\phi}_{\mathrm{BOW}}(\boldsymbol{s}_2) = \begin{array}{c} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \end{array} \begin{array}{l} \leftarrow \text{This} \\ \leftarrow \text{product} \\ \leftarrow \text{shipped} \\ \leftarrow \text{not} \\ \leftarrow \text{fast} \\ \leftarrow \text{and} \\ \leftarrow \text{was} \\ \leftarrow \text{disappointing} \end{array}$$

Bag of words **cannot** learn "positional" effect of text elements

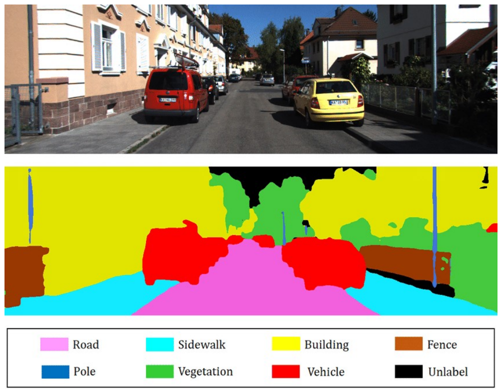
Overview

1. **Modelling Data**
   - Modelling Text
   - **Modelling Images**
2. Linear Basis Function Models
   - Vector Spaces
   - Linear Mappings and Matrices

Modelling Images

- **Raw image data**. *An image is a rectangular array of picture elements ("pixels") that consist of a triple of intensities of the base colors red, blue, and green.*

- Images can also be modelled at three levels of granularity depending on application
  1. **Pixels**: Image segmentation
  2. (Non-overlapping) **patches**: Object recognition
  3. **Whole image**: Image classification



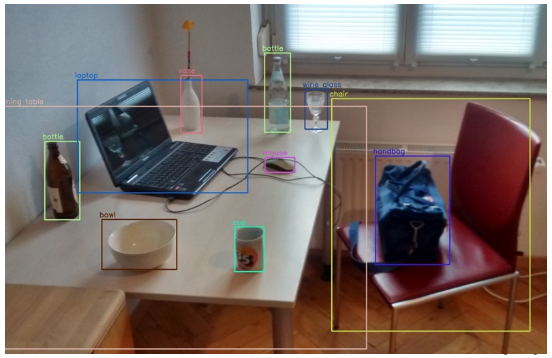


Image Content in Frequency and Location

- Image data contains signal (photon counts) at *fixed* locations in the image

$$\boldsymbol{\phi}_{\mathrm{RGB}}\Bigg($$

$$\Bigg) = \begin{bmatrix} 127 \\ 0 \\ \vdots \\ 32 \\ 6 \end{bmatrix}$$

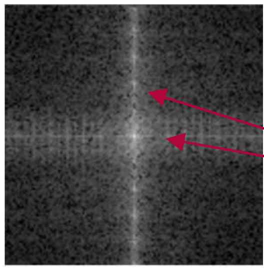
← Red value at location (1,1)
← Green value at location (1,1)
← Green value at location (800,600)
← Blue value at location (800,600)

- **Observation**. Recurring patterns are more visible in the frequency domain.
- **Idea**. Discrete Fourier transform (DFT) to transform the image

$$\boldsymbol{\phi}_{\mathrm{DFT}}(x) = \boldsymbol{F}_{\mathrm{DFT}}\boldsymbol{\phi}_{\mathrm{RGB}}(x)$$

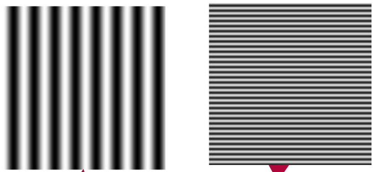
  - In practice, there is a $\mathcal{O}(\#\mathrm{pixel} \cdot \log(\#\mathrm{pixel}))$ algorithm for fast Fourier transform (FFT)

Raw image (in single-grey channel)



Fourier transform of image

Horizontal and vertical patterns by single features



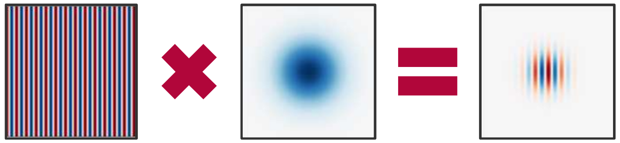
$$\boldsymbol{F}_{\mathrm{DFT}} = \boldsymbol{W}_{\mathrm{DFT}} \otimes \boldsymbol{W}_{\mathrm{DFT}}$$

Kronecker product of 1D discrete Fourier transform $\boldsymbol{W}_{\mathrm{DFT}}$

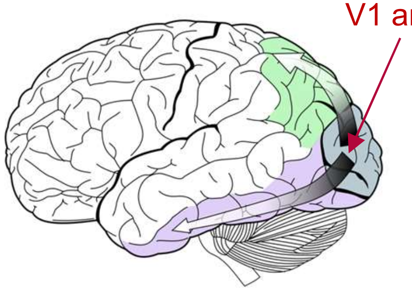
Gabor Wavelets: Mixing Location and Frequency

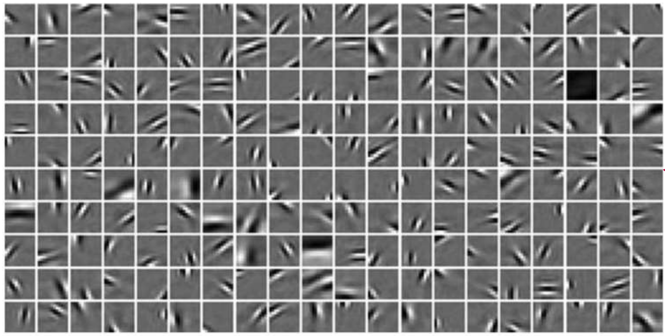
Dennis Gabor
(1900 – 1978)

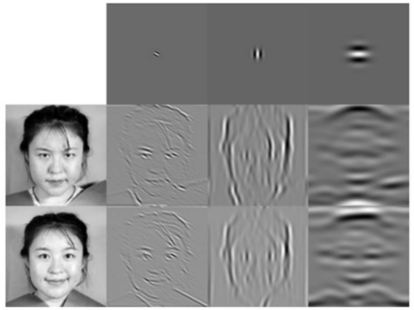
- Both representations are extremes:
  - **Raw image**: Single features $\phi_{\mathrm{RGB},i}(x)$ at a location without frequency
  - **DFT image**: Single features $\phi_{\mathrm{DFT},i}(x)$ in frequency without a location
- **Gabor wavelets**. A *Gabor wavelet* is a set of basis functions that is obtained by multiplying a Fourier basis function with a Gaussian density.



  - Combine both spatial and frequency information in image features
  - Basis functions for sparse encoding by the brain in V1







Filters “implemented” in the V1 (sparse coding!)

Bruno Olshausen & David Field (1997). Sparse Coding with Overcomplete Basis Set: A Strategy Employed by V1?

Overview

1. Modelling Data
   - Modelling Text
   - Modelling Images
2. **Linear Basis Function Models**
   - Vector Spaces
   - Linear Mappings and Matrices

Linear Basis Function Models

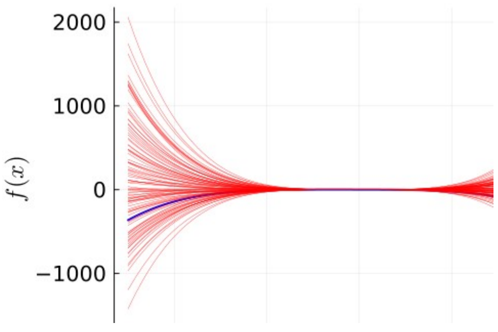
- **Linear Basis Function Models**. *Given an input space* $\mathcal{X}$ *and* D basis functions ("features") $\phi_j : \mathcal{X} \to \mathbb{R}$, *a* linear basis function model *is a function of the form*

$$f(x; \boldsymbol{w}) = w_0 + w_1 \cdot \phi_1(x) + w_2 \cdot \phi_2(x) + \cdots + w_D \cdot \phi_D(x)$$

- It's called a linear model, but the linearity is w.r.t. $\boldsymbol{w}$ not $x \in \mathcal{X}$!
- **Examples**: 4 basis function ($D = 4$) and 100 random parameters $\boldsymbol{w}$.

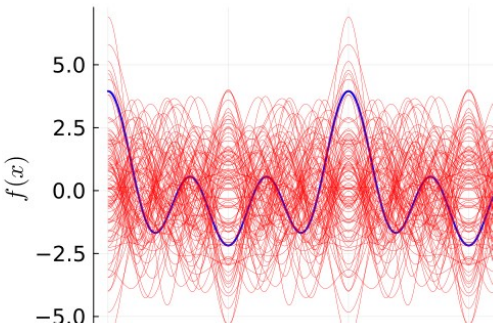
Polynomial Basis

$\phi_j(x) = x^j$



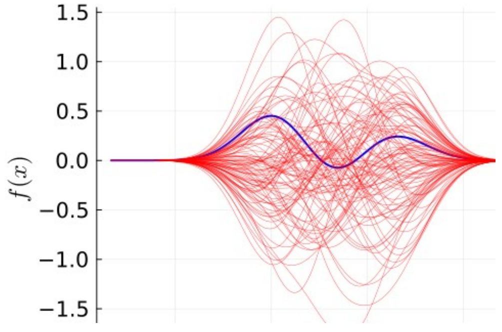
Fourier Basis

$\phi_j(x) = \cos(\pi j \cdot x)$



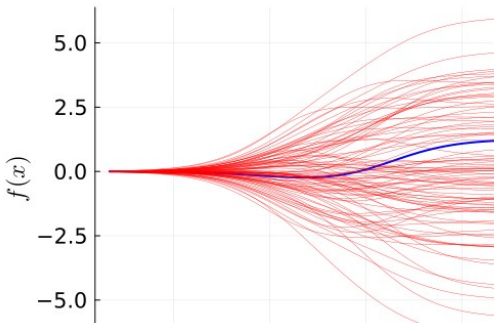
Gaussian Basis

$\phi_j(x) = \mathcal{N}(x; j, 1)$



Sigmoid Basis

$$\phi_j(x) = \frac{\exp(x - j)}{1 + \exp(x - j)}$$



Linear Models for Text

- A linear model captures the *effect* of presence of a text element!
- In practice, the feature vector for one-hot encoded data is *never* explicitly computed because

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}_{\mathrm{OHE}}(s) = \boxed{\sum_{i=1}^{|S|} w_i \cdot \phi_i(s)} = \boxed{w_{\mathrm{idx}(s)}}$$

$\mathcal{O}(|S|)$

$\mathcal{O}(\log_2|S|)$

- All we need is the inverse function mapping of an actual text element s

$$\mathrm{idx}(s) \coloneqq i \quad \Leftrightarrow \quad \phi_{\mathrm{OHE},i}(s) = 1$$

- Two common techniques for encoding a whole text
  - **Variable-length text** $s_1 s_2 \cdots$: Sum of all one-hot encoded vectors ("bag of words")

$$\boldsymbol{\phi}_{\mathrm{BOW}}(s_1 s_2 \cdots) = \boxed{\sum_j \boldsymbol{\phi}_{\mathrm{OHE}}(s_j)}$$

  - **Fixed-length text** $s_1 s_2 \cdots s_n$: A stacked $n \cdot |S|$ dimensional vector

$$\boldsymbol{\phi}_{\mathrm{FL}}(s_1 s_2 \cdots s_n) = \boxed{\begin{bmatrix} \boldsymbol{\phi}_{\mathrm{OHE}}(s_1) \\ \vdots \\ \boldsymbol{\phi}_{\mathrm{OHE}}(s_n) \end{bmatrix}}$$

A linear model is only $\mathcal{O}(n \cdot \log_2|S|)$

Overview

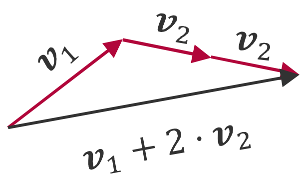
1. Modelling Data
   - Modelling Text
   - Modelling Images
2. Linear Basis Function Models
   - **Vector Spaces**
   - Linear Mappings and Matrices

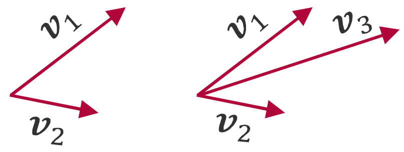
Linear Models and Vector Spaces

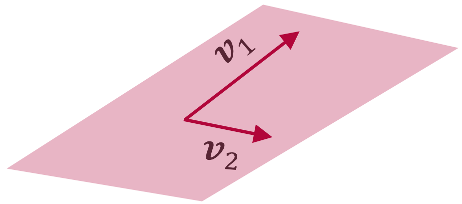
- To write a linear model $f(x; \boldsymbol{w}) = \sum_{i=1}^{D} w_i \cdot \phi_i(x)$ compactly, we use two *vectors*
  1. A vector $\boldsymbol{\phi} = \begin{bmatrix} \phi_1(x) \\ \vdots \\ \phi_D(x) \end{bmatrix}$ of D feature values for the input x
  2. A vector $\boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_D \end{bmatrix}$ of D weightings ("weight" parameters)
- **Vector Space**. *A vector space is a set V that satisfies the following axioms: There exists a null element $\mathbf{0} \in V$ such that for all $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in V$ and scalars $a, b \in \mathbb{R}$*
  1. **Associativity**: $\boldsymbol{u} + (\boldsymbol{v} + \boldsymbol{w}) = (\boldsymbol{u} + \boldsymbol{v}) + \boldsymbol{w}$
  2. **Commutativity**: $\boldsymbol{u} + \boldsymbol{v} = \boldsymbol{v} + \boldsymbol{u}$
  3. **Identity element (of vector addition)**: $\boldsymbol{v} + \mathbf{0} = \boldsymbol{v}$
  4. **Inverse element**: $\boldsymbol{v} + (-\boldsymbol{v}) = \mathbf{0}$
  5. **Identity element (of scalar multiplication)**: $1 \cdot \boldsymbol{v} = \boldsymbol{v}$
  6. **Distributivity (w.r.t. vector addition)**: $a \cdot (\boldsymbol{u} + \boldsymbol{v}) = a \cdot \boldsymbol{u} + a \cdot \boldsymbol{v}$
  7. **Distributivity (w.r.t. scalar addition)**: $(a + b) \cdot \boldsymbol{v} = a \cdot \boldsymbol{v} + b \cdot \boldsymbol{v}$

Bernhard Bolzano (1781 – 1848)

Linear Independence, Span and Basis







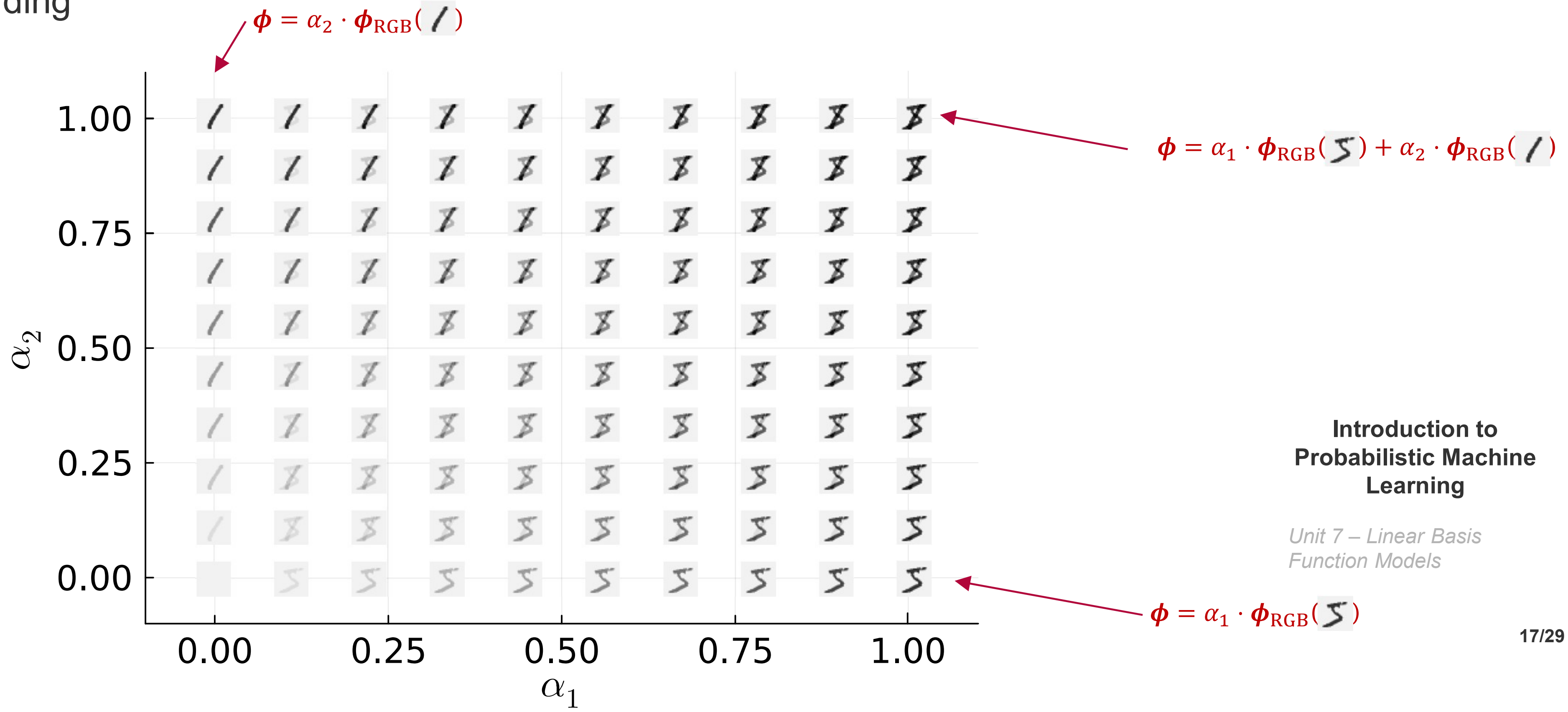
- **Linear combination**. *Given a set* $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ *of a vector space* V*, a* linear combination *is defined by (*$a_1, a_2, \ldots, a_n$ *are called coefficients)*

$$a_1 \cdot \boldsymbol{v}_1 + a_2 \cdot \boldsymbol{v}_2 + \cdots + a_n \cdot \boldsymbol{v}_n$$

- **Linear independence**. *A set* $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ *is called* linearly independent *if no* $\boldsymbol{v}_i$ can be written as a linear combination of $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{i-1}, \boldsymbol{v}_{i+1}, \ldots, \boldsymbol{v}_n$.
- **Span**. *Given a set* $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ *of a vector space* V*, the* span $\mathrm{span}(\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n)$ *is the set of all* linear combinations of $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$.
  - **Example**:
    - $\mathrm{span}(\boldsymbol{v}_1)$ is the line through the origin along $\boldsymbol{v}_1$
    - $\mathrm{span}(\boldsymbol{v}_1, \boldsymbol{v}_2)$ is the plane through the origin along $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$
- **Basis**. *A subset* $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n$ *of a vector space* V *is called a* basis *if its span equals the whole vector space, that is,* $\mathit{span}(\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n) = V$.
- **Dimensionality**. *All bases of a vector space* V *have the same cardinality called the* dimensionality *of the vector space,* $\dim(V)$.

Example: Linear Independence & Span

- Consider two 28x28 pictures and and their $28^2 = 784$-dimensional $\boldsymbol{\phi}_{\mathrm{RGB}}$ embedding

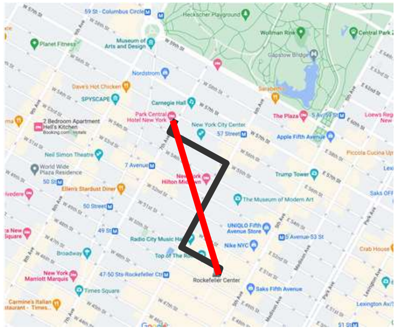


Introduction to Probabilistic Machine Learning

Unit 7 – Linear Basis Function Models

Distances and Scalar Products

- So far, a vector space is a set but there is no notion of **distance**!
  - In New York, the **walking distance** between two points is much longer than the **flying distance**!



- **Metric space**. *A metric space is a vector space V together with a metric $d\colon V \times V \to \mathbb{R}^+$ that has the following properties for all $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$:*

$$d(\boldsymbol{x}, \boldsymbol{y}) = 0 \Leftrightarrow \boldsymbol{x} = \boldsymbol{y}$$
$$d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$$
symmetry
$$d(\boldsymbol{x}, \boldsymbol{y}) \leq d(\boldsymbol{x}, \boldsymbol{z}) + d(\boldsymbol{z}, \boldsymbol{y})$$
triangle inequality

- **Scalar product**. *Given a vector space a scalar product $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y} \in \mathbb{R}$ has the following properties for all $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$:*

$$\boldsymbol{x}^T\boldsymbol{x} = 0 \Leftrightarrow \boldsymbol{x} = \boldsymbol{0}$$
$$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y} = \boldsymbol{y}^{\mathrm{T}}\boldsymbol{x}$$
symmetry
$$(a \cdot \boldsymbol{x} + b \cdot \boldsymbol{y})^{\mathrm{T}}\boldsymbol{z} = a \cdot \boldsymbol{x}^{\mathrm{T}}\boldsymbol{z} + b \cdot \boldsymbol{y}^{\mathrm{T}}\boldsymbol{z}$$
linearity

- *Given a scalar product, the function $d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{y})}$ is a metric!*

Overview

Linear Mappings and Matrices

- **Linear mapping**. *A function $f: V \rightarrow W$ is a* linear mapping *between vector space V and W if for any two vectors $\boldsymbol{u}, \boldsymbol{v} \in V$ and any scalar $c \in \mathbb{R}$:*

$$f(\boldsymbol{u}+\boldsymbol{v}) = f(\boldsymbol{u}) + f(\boldsymbol{v})$$
$$f(c \cdot \boldsymbol{u}) = c \cdot f(\boldsymbol{u})$$

  - Addition and scalar multiplication can be applied before or after the map!
  - It follows that $f(\mathbf{0}) = f(0 \cdot \boldsymbol{v}) = 0 \cdot f(\boldsymbol{v}) = \mathbf{0}$!

- **Theorem**. *Every linear mapping $f: V \rightarrow W$ between V and W of dimension n and m can be represented via a matrix multiplication with an $m \times n$ matrix $\boldsymbol{A}$. The rank of $\boldsymbol{A}$ is the dimensionality of the image of W, that is $\operatorname{rank}(\boldsymbol{A}) = \dim(W)$.*
  - **Proof**. If $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_n\}$ is a basis for V and $\{\boldsymbol{w}_1, \dots, \boldsymbol{w}_m\}$ is a basis for W then we know

because $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n$ is a basis

$$f(\boldsymbol{v}_j) = a_{1,j} \cdot \boldsymbol{w}_1 + \cdots + a_{m,j} \cdot \boldsymbol{w}_m$$

because $\boldsymbol{w}_1, \dots, \boldsymbol{w}_m$ is a basis

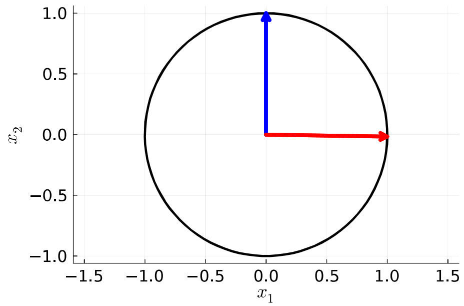
$$f(\boldsymbol{v}) = f(c_1 \boldsymbol{v}_1 + \cdots + c_n \boldsymbol{v}_n) = c_1 \cdot f(\boldsymbol{v}_1) + \cdots + c_n \cdot f(\boldsymbol{v}_n)$$

because f is a linear mapping

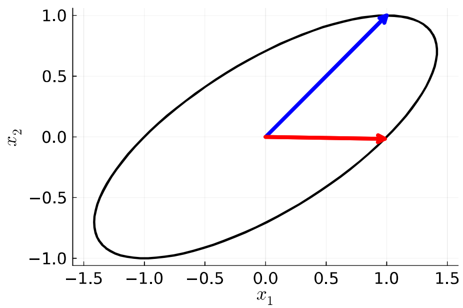
$$= \left(\sum_{j=1}^{n} c_j \cdot a_{1,j}\right) \cdot \boldsymbol{w}_1 + \cdots + \boxed{\left(\sum_{j=1}^{n} c_j \cdot a_{m,j}\right)} \cdot \boldsymbol{w}_m$$

$[\boldsymbol{A}\boldsymbol{c}]_m$

  - For $V = \mathbb{R}^n$ and $W = \mathbb{R}^m$, the columns of $\boldsymbol{A}$ are the images of the basis vectors in $\mathbb{R}^n$.

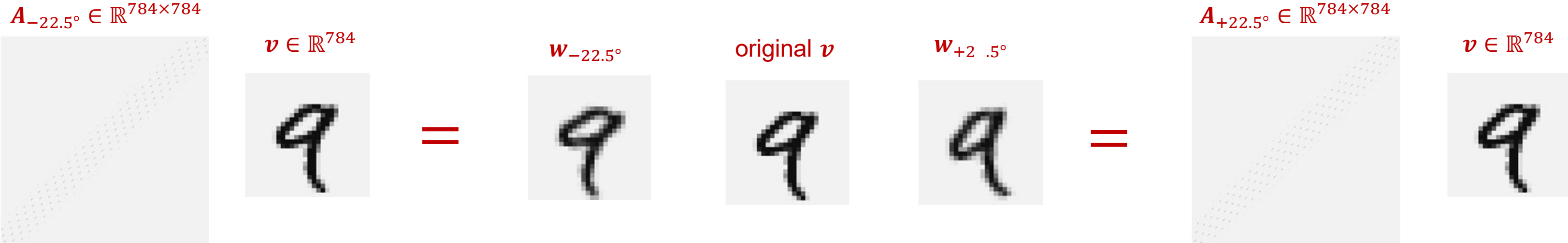


$$f(\boldsymbol{x}) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \boldsymbol{x}$$



Example: Linear Mappings and Matrices

- Consider a 28x28 picture  and its $28^2 = 784$-dimensional $\boldsymbol{\phi}_{\mathrm{RGB}}$ embedding
- Then the rotation of the image content can be expressed as a linear mapping from 784-dimensional onto itself
  - By computing the four fractional source pixel that contributed to a target pixel



- The following movie was produced by 120 linear mappings of 3° applied to the same 784-dimensional $\boldsymbol{\phi}_{\mathrm{RGB}}$ embedding of the source image

Matrix Types and Properties

- **Square Matrix**. *A matrix* $\boldsymbol{A} \in \mathbb{R}^{n \times m}$ *where* $m = n$ *is called a* square matrix*.*
  - It parameterizes mappings of a space onto itself.
- **Diagonal Matrix**. *A square matrix* $\boldsymbol{A}$ *is called* diagonal *if* $\boldsymbol{A}_{ij} = 0$ *for all* $i \neq j$*.*
  - Geometrically, a diagonal matrix is an axis scaling (mapping).
  - A special diagonal matrix is $\mathbf{A}_{ii} = 1$ which is also called identity matrix $\boldsymbol{I}$.
- **Orthogonal Matrix**. *A square matrix* $\boldsymbol{A}$ *is called orthogonal if* $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}} = \boldsymbol{I}$*.*
  - Geometrically, an orthogonal matrix is a rotation and mirroring (mapping).
- **Symmetric Matrix**. *A square matrix* $\boldsymbol{A}$ *is called* symmetric *if and only if* $\boldsymbol{A} = \boldsymbol{A}^{\mathrm{T}}$*.*
- **Positive semi-definite matrix**. *A symmetric matrix* $\boldsymbol{A}$ *is called* positive definite *if and only if*

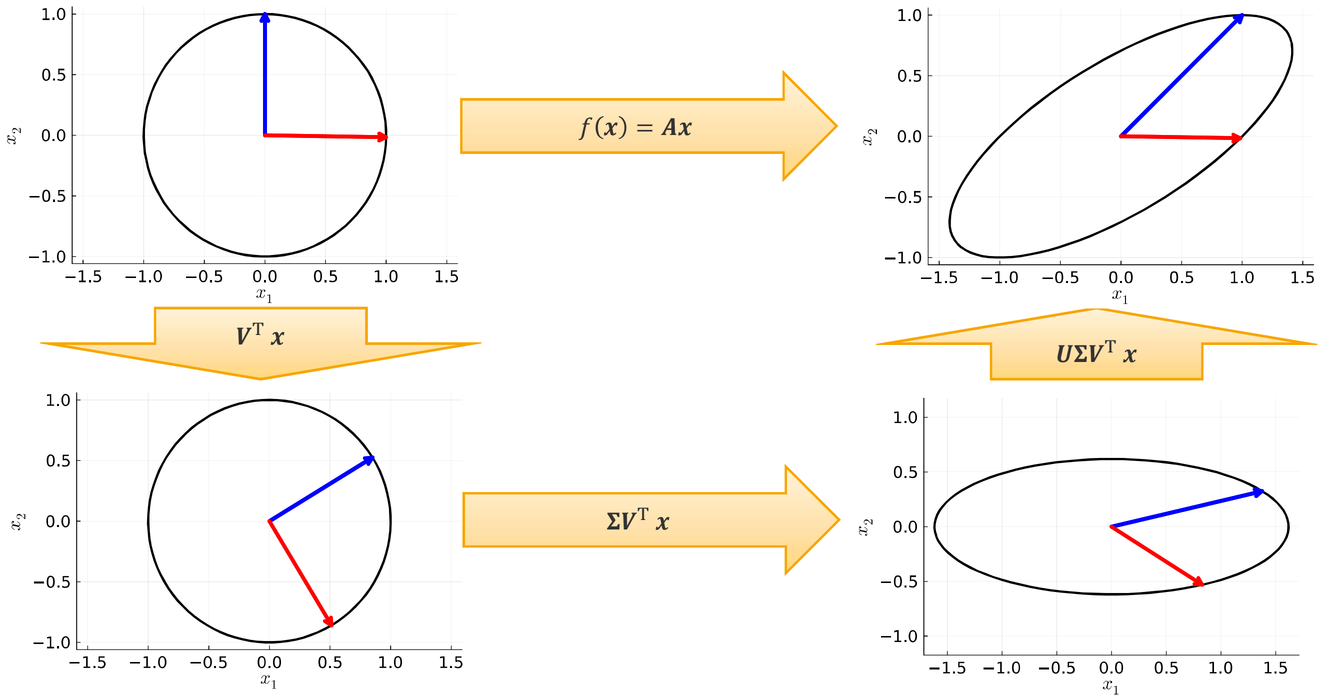
$$\forall \boldsymbol{x} \neq \mathbf{0}: \boldsymbol{x}^{\mathrm{T}} \boldsymbol{A} \boldsymbol{x} > 0$$

  - Positive definiteness means that no axis is inverted or removed in the mapping

Singular Value Decomposition

- **Singular Value Decomposition**. *Any matrix* $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ *has a decomposition into three matrices* $\boldsymbol{U} \in \mathbb{R}^{n\times k}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{k\times m}$ *and* $\boldsymbol{V} \in \mathbb{R}^{m\times m}$ *such that* $\boldsymbol{U}$ *and* $\boldsymbol{V}$ *are orthogonal and* $\boldsymbol{\Sigma}$ *is only non-zero on diagonal elements (where* $k = \mathrm{rank}(\boldsymbol{A})$*)*

$$\boldsymbol{A} = \boldsymbol{U\Sigma V}^{\mathrm{T}}$$



Singular Value Decomposition Example

- Imagine we arrange the 784-dimensional RGB vectors for 1000 images of digits in the rows of a matrix $\boldsymbol{A}$. Then the SVD has the following property

$$\boldsymbol{A} = \underbrace{\boldsymbol{U}}_{\boldsymbol{U} \in \mathbb{R}^{1000\times 784}}\underbrace{\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}}_{= \boldsymbol{B} \in \mathbb{R}^{784\times 784}} = \boldsymbol{U}\boldsymbol{B} = [\boldsymbol{u}_1 \quad \cdots \quad \boldsymbol{u}_{784}] \cdot \begin{bmatrix} \boldsymbol{b}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{b}_{784}^{\mathrm{T}} \end{bmatrix}$$

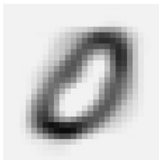
First 5 rows of $\boldsymbol{B}$

$\boldsymbol{b}_1$ $\boldsymbol{b}_2$ $\boldsymbol{b}_3$ $\boldsymbol{b}_4$ $\boldsymbol{b}_5$

- Reconstruction with $\widehat{\boldsymbol{A}} = \underbrace{[\boldsymbol{u}_1 \quad \cdots \quad \boldsymbol{u}_k]}_{\in \mathbb{R}^{1000\times k}} \cdot \underbrace{\begin{bmatrix} \boldsymbol{b}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{b}_k^{\mathrm{T}} \end{bmatrix}}_{\in \mathbb{R}^{k\times 784}}$ for the first k singular values

Original $\boldsymbol{a}_1$

Reconstruction $\hat{\boldsymbol{a}}_1$



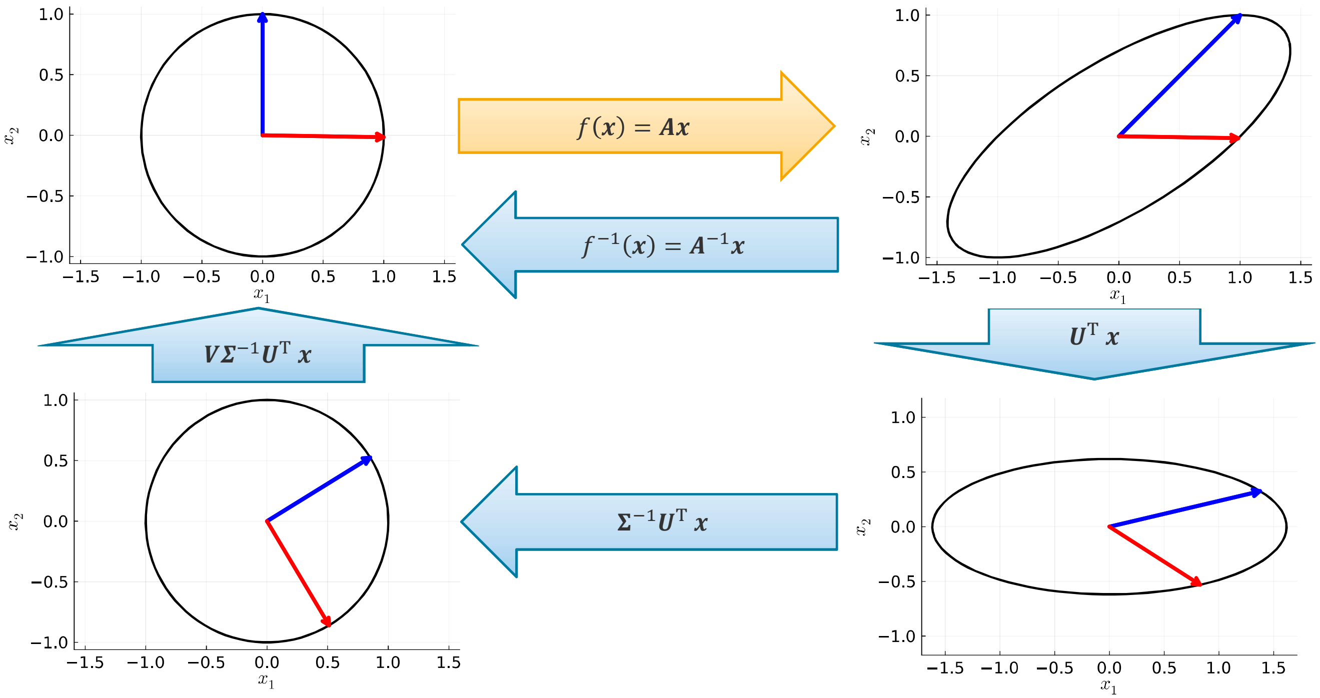
Inverse of a Matrix and SVD

- **Inverse**. *The inverse $\boldsymbol{A}^{-1}$ of a full-rank square matrix $\boldsymbol{A}$ has the property that*

$$\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}$$

  - One way to compute it is with the singular value decomposition

$$\boldsymbol{A}^{-1} = \boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^{\mathrm{T}}$$



Cholesky Decomposition

Hasso Plattner Institut
Digital Engineering · Universität Potsdam

- **Numerical Challenge**: Given a symmetric, positive-definite matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and a vector $\boldsymbol{b} \in \mathbb{R}^n$ find the solution $\boldsymbol{x}$ such that
$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$
- **Naïve Solution**: Invert the matrix $\boldsymbol{A}$ and compute
$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$$
  - **Challenges**: If $\boldsymbol{A}$ has some singular values close to zero, this is numerically unstable!
- **Cholesky Decomposition**: Every symmetric positive-definite matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ has a unique decomposition into a lower-triangular matrix $\boldsymbol{L} \in \mathbb{R}^{n \times n}$
$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$$
  - **Advantage**: Finding $\boldsymbol{y}$ such that $\boldsymbol{L}\boldsymbol{y} = \boldsymbol{b}$ can be done in $O(n^2)$ without an inverse simply using back-substitution (written as $\boldsymbol{y} = \boldsymbol{L} \backslash \boldsymbol{b}$)!
- **Cholesky Solution**: Find the solution $\boldsymbol{x}$ for $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is a two-step algorithm
  1. Compute $\boldsymbol{y} = \boldsymbol{L} \backslash \boldsymbol{b}$
  2. Compute $\boldsymbol{x} = \boldsymbol{L}^{\mathrm{T}} \backslash \boldsymbol{y}$

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \Leftrightarrow \boldsymbol{L}\boxed{\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x}} = \boldsymbol{b}$$

$$\downarrow$$

$$\boldsymbol{L}\boldsymbol{y} = \boldsymbol{b}$$

André-Louis Cholesky
(1875 – 1918)

$$\begin{pmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$y_1 = \frac{b_1}{L_{11}}$$

$$y_2 = \frac{b_2 - L_{21} \cdot y_1}{L_{22}}$$

$$y_3 = \frac{b_3 - L_{31} \cdot y_1 - L_{32} \cdot y_2}{L_{33}}$$

Cholesky Decomposition (ctd)

$$\boldsymbol{LL}^{\mathrm{T}} = \begin{pmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \cdot \begin{pmatrix} L_{11} & L_{21} & L_{31} \\ 0 & L_{22} & L_{32} \\ 0 & 0 & L_{33} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{21} & A_{31} \\ A_{21} & A_{22} & A_{32} \\ A_{31} & A_{32} & A_{33} \end{pmatrix}$$

1. $L_{11}^2 = A_{11}$ $\qquad L_{11} = \sqrt{A_{11}}$
2. $L_{11} \cdot L_{21} = A_{21}$ $\qquad L_{21} = \frac{A_{21}}{L_{11}}$
3. $L_{21}^2 + L_{22}^2 = A_{22}$ $\qquad L_{22} = \sqrt{A_{22} - L_{21}^2}$
4. $L_{11} \cdot L_{31} = A_{31}$ $\qquad L_{31} = \frac{A_{31}}{L_{11}}$
5. $L_{21} \cdot L_{31} + L_{22} \cdot L_{32} = A_{32}$ $\qquad L_{32} = \frac{A_{32} - L_{21} \cdot L_{31}}{L_{22}}$
6. $L_{31}^2 + L_{32}^2 + L_{33}^2 = A_{33}$ $\qquad L_{33} = \sqrt{A_{33} - L_{31}^2 - L_{32}^2}$

```julia
function cholesky(A::Matrix)
    # check that the matrix is square
    if (size(A)[1] != size(A)[2])
        error("matrix must be square")
    end
    n = size(A)[1]

    # create a zero matrix
    L = zeros(n, n)

    # run the Cholesky decomposition
    for i = 1:n
        for j = 1:i
            sum = 0
            for k = 1:(j-1)
                sum += L[i, k] * L[j, k]
            end

            L[i, j] = (i == j) ? sqrt(A[j, j] - sum) : (A[i, j] - sum) / L[j, j]
        end
    end

    return (L)
end
```

Summary

- **Modelling Data**
  - Both textual and image data can be represented at different levels of granularity
  - Images should ideally be represented in terms of feature of location and frequency: Gabor wavelets/filters are excellent candidate functions for such features
- **Linear Basis Functions Models**
  - Non-linear prediction models can be formed with non-linear basis functions
  - Linearity is in the parameters, *not* the input dimensions of the data
- **Linear Mappings and Matrices**
  - All linear mappings can be expressed via matrix products
  - The singular value decomposition is the rotation-scaling-rotation view on a mapping
  - The Cholesky decomposition is numerically stable to help solve for the parameters of a linear model and is crucial for solving least-squares problems

See you next week!