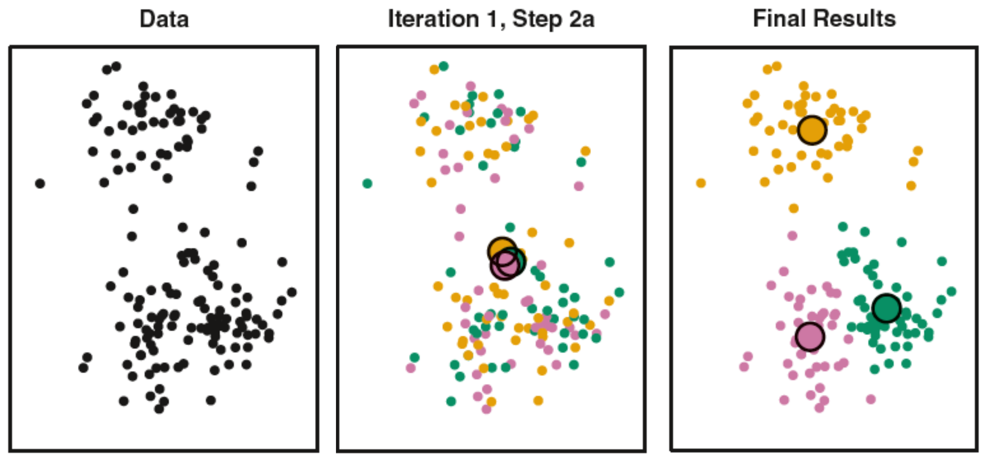
Topics on Machine Learning

ECE 301 Linear Systems

Machine Learning: An Overview

(James, Witten, Hastie, & Tibshirani, 2013)



- Unsupervised learning:
  - Learns from a set of unlabeled data to discover patterns, without human supervision.
  - We'll cover principal component analysis (PCA).
- Supervised learning:
  - Learns an input–output mapping based on labeled data.
  - We'll cover linear regression and neural networks.

Strawberry

Bathing cap

Flute

Traffic light

(Li and Russakovsky, 2013)

Machine Learning Topics and Learning Objectives

- Topic 1: Linear algebra
  - Explain linear algebra concepts such as linear independence, vector space, and orthogonal basis
  - Conduct eigendecomposition for symmetric matrices using Matlab
- Topic 2: Principal component analysis (unsupervised learning)
  - Explain the two equivalent goals of PCA
  - Implement the PCA algorithm and visualize the results
- Topic 3: Linear regression and prediction (supervised learning)
  - Interpret regression problem mathematically and geometrically
  - Apply linear regression to learning problems without overfit
- Topic 4: Convolutional neural network (CNN)
  - Describe the structure of CNN
  - Build and train simple CNNs using a deep learning package

Linear Algebra

Learning objectives

- Explain linear algebra concepts such as linear independence, vector space, and orthogonal basis
- Conduct eigendecomposition for symmetric matrices using Matlab

(Refer to ECE 220's textbook for a review on vector and matrix. A comprehensive treatment of linear algebra can be found in Scheffe's appendices, available on the library's course reserves.)

Linear Algebra Review: Vector

- Vector: an <u>ordered</u> n-tuple.

  Row vector: $\mathbf{x} = \begin{bmatrix} x_1, & x_2, & \ldots, & x_n \end{bmatrix}$

  Column vector: $\mathbf{x} = \begin{bmatrix} x_1, & x_2, & \ldots, & x_n \end{bmatrix}^T$

  (Assume all vectors are column from now on.)

- Vector properties:

  $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$ (commutative)

  $\mathbf{x} + (\mathbf{y} + \mathbf{z}) = (\mathbf{x} + \mathbf{y}) + \mathbf{z}$ (associative)

  $c \begin{bmatrix} x_1, \ldots, x_n \end{bmatrix} = \begin{bmatrix} cx_1, \ldots, cx_n \end{bmatrix}$ (scaling)

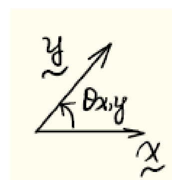
- Norm/length: $\|\mathbf{x}\| = (\mathbf{x}^T \mathbf{x})^{\frac{1}{2}}$, e.g., $\mathbf{x} = \begin{bmatrix} 3, 4 \end{bmatrix}^T$, $\|\mathbf{x}\| = 5$.

Linear Algebra Review: Vector (cont'd)

- Inner product of $\mathbf{x}$ and $\mathbf{y}$:

$$\mathbf{x}^T\mathbf{y} = [x_1, \cdots, x_n] \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^{n} x_i y_i = \mathbf{y}^T\mathbf{x}.$$

- $\mathbf{x}^T\mathbf{y} = \|\mathbf{x}\| \cdot \|\mathbf{y}\| \cos\theta_{\mathbf{x},\mathbf{y}}$



Ex: $\mathbf{x} = [1, 0],\ \mathbf{y} = [1, 1]$

$\cos\theta_{\mathbf{x},\mathbf{y}} = \frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} = \frac{1\cdot 1 + 0\cdot 1}{\sqrt{1^2+0^2}\cdot\sqrt{1^2+1^2}} = \frac{\sqrt{2}}{2}$

$\Rightarrow \theta_{\mathbf{x},\mathbf{y}} = 45^\circ$

- Def: $\mathbf{x}$ and $\mathbf{y}$ are orthogonal if $\mathbf{x}^T\mathbf{y} = 0$.

- Remark: When $\mathbf{x}^T\mathbf{y} = 0$, $\cos^{-1}\left(\frac{0}{\|\mathbf{x}\|\|\mathbf{y}\|}\right) = \frac{\pi}{2}(2k+1)$.

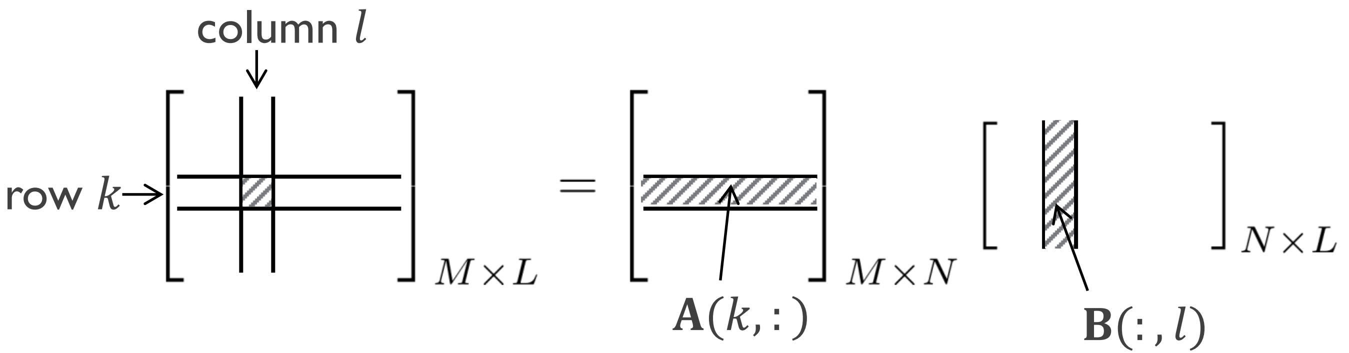
Linear Algebra Review: Matrix

- Matrix: $\mathbf{A} = [a_{kl}] = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ a_{21} & \cdots & a_{2N} \\ \vdots & \ddots & \vdots \\ a_{M1} & \cdots & a_{MN} \end{bmatrix} \in \mathbb{R}^{M \times N}$, M rows, N columns.
- Addition: $\mathbf{A} + \mathbf{B} = [a_{kl} + b_{kl}] = \mathbf{B} + \mathbf{A}$
- Scaling: $c\mathbf{A} = [ca_{kl}]$ Ex: $2 \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 1 & 2 \end{bmatrix}$
- Transpose "T": $\mathbf{A}^T = [a_{kl}]^T = [a_{lk}]$ Ex: $\begin{bmatrix} 1 & -1 & 1 \\ 2 & 4 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 2 \\ -1 & 4 \\ 1 & 6 \end{bmatrix}$
- Special matrices: $\mathbf{0}_{M \times N} = [0]_{M \times N}$, $\mathbb{1}_{M \times N} = [1]_{M \times N}$,

Identity matrix $\mathbf{I}_M = \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix} = \mathrm{diag}(\mathrm{ones}(M, 1))$.

Linear Algebra Review: Matrix (cont'd)

- Matrix Multiplication: $\mathbf{C} = \mathbf{AB}$, where $c_{kl} = \sum_{q=1}^{N} a_{kq} b_{ql} = \mathbf{A}(k,:)\mathbf{B}(:,l)$



Note: In general, $\mathbf{AB} \neq \mathbf{BA}$. Why?

Ex: $\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 3 & 4 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} 3+5 & 4+6 \\ -3+5 & -4+6 \end{bmatrix} = \begin{bmatrix} 8 & 10 \\ 2 & 2 \end{bmatrix}$

- Def: $\mathbf{A}^{-1} = \mathbf{B}$ if ① $\mathbf{A}$ is square, and ② $\mathbf{AB} = \mathbf{I} = \mathbf{BA}$.

Linear Algebra Review: Matrix (cont'd)

- For 2-by-2 matrices: $\mathbf{A}^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \frac{1}{\det \mathbf{A}} \begin{bmatrix} d & -c \\ -b & a \end{bmatrix}^T = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

Ex: $\mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{A} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$

$$\mathbf{A}^{-1}(\mathbf{A}\mathbf{x}) = \mathbf{A}^{-1}\mathbf{b} \Longrightarrow \mathbf{x} = \mathbf{A}^{-1}\mathbf{b} = \tfrac{1}{2} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$$

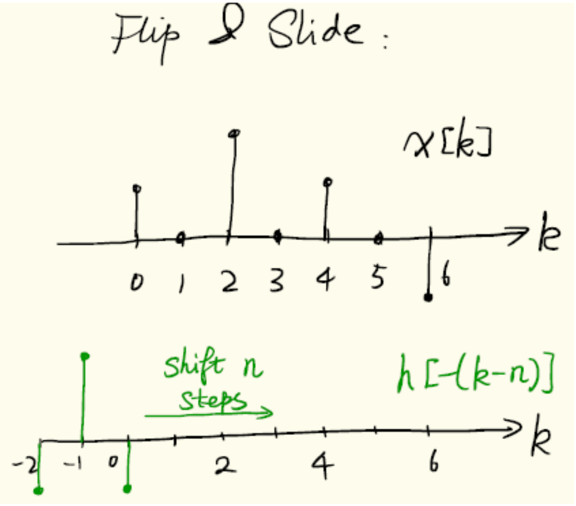
Motivation: Linear Algebra for Discrete Convolution

Ex: $x[n] = \{\underset{\substack{\uparrow \\ x[0]}}{1}, 0, 2, 0, 1, 0, -1\}$, length = 7, $h[n] = \{\underset{\substack{\uparrow \\ h[0]}}{-1}, 2, -1\}$. length = 3 $y[n] = x[n] * h[n] = ?$ length = ?

Matrix-vector form:

$$\begin{bmatrix} -1 \\ 2 \\ -3 \\ 4 \\ -3 \\ 2 \\ 0 \\ -2 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & & & \cdots & & & 0 \\ 2 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ \vdots & & -1 & 2 & -1 & & \vdots \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \\ & & & & & -1 & 2 \\ 0 & & & \cdots & & & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 2 \\ 0 \\ 1 \\ 0 \\ -1 \end{bmatrix}$$

$\mathbf{y} \in \mathbb{R}^9$ $\quad$ $\mathbf{H} \in \mathbb{R}^{9\times 7}$ $\quad$ $\mathbf{x} \in \mathbb{R}^7$



Linear Independence of a Set of Vectors

- Given $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$. Defs:

$$\alpha_1 \mathbf{v}_1 + \cdots + \alpha_n \mathbf{v}_n = \mathbf{0} \Rightarrow \alpha_i = 0, \forall i$$ (linearly independent)

$$\alpha_1 \mathbf{v}_1 + \cdots + \alpha_n \mathbf{v}_n = \mathbf{0} \Rightarrow \text{not all } \alpha_i = 0$$ (linearly dependent)

- For "linearly dependent" case (when $\alpha_1 \neq 0$), we may write:

$$\mathbf{v}_1 = \beta_2 \mathbf{v}_2 + \cdots + \beta_n \mathbf{v}_n$$ Why?

- Ex: $\mathbf{v}_1 = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}^T, \ \mathbf{v}_2 = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}^T.$

$$\alpha_1 \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} + \alpha_2 \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \mathbf{0} \quad \Rightarrow \begin{cases} \alpha_1 + \alpha_2 = 0 \\ 2\alpha_1 + 0 = 0 \\ \alpha_1 + \alpha_2 = 0 \end{cases} \Rightarrow \begin{cases} \alpha_1 = 0 \\ \alpha_2 = 0 \end{cases} \Rightarrow \text{linearly independent}$$

Linear Independence of a Set of Vectors (cont'd)

- Ex: $\mathbf{v}_1 = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}^T$, $\mathbf{v}_4 = \begin{bmatrix} -2 & -4 & -2 \end{bmatrix}^T$.

$$\mathbf{v}_4 = -2\mathbf{v}_1 \Rightarrow \text{linearly dependent}$$

- Ex: $\mathbf{v}_1 = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}^T$, $\mathbf{v}_2 = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}^T$, $\mathbf{v}_3 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$.

$$\mathbf{v}_1 = \mathbf{v}_2 + 2\mathbf{v}_3 \Rightarrow \text{linearly dependent}$$

Vector Space

- Def: Vector space: A set, V of all vectors that are linear combination of $\{\mathbf{v}_i\}_{i=1}^n$, i.e.,

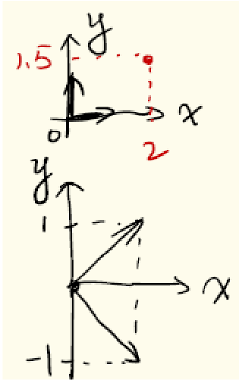
$$\boxed{V = \left\{ \mathbf{v} = \sum_{i=1}^{n} \alpha_i \mathbf{v}_i,\ \alpha_i \in \mathbb{R} \right\}.}$$

$\mathbf{v}_i$'s are said to span the vector space, i.e., $V = \mathrm{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$.

- Ex:

$$V^{(1)} = \left\{ \alpha_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix},\ \alpha_i \in \mathbb{R} \right\} = \mathbb{R}^2$$

$$V^{(2)} = \left\{ r_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + r_2 \begin{bmatrix} 1 \\ -1 \end{bmatrix},\ r_i \in \mathbb{R} \right\} = \mathbb{R}^2$$



Basis for Vector Space

- Def: A basis for V is a set of linearly independent vectors that span V.
- Ex: Q1. What is V? Q2. Are vectors linearly independent?

$$\left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\} \qquad \left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$$

$$\left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\} \qquad \left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right\}$$

Basis for Vector Space

- Def: A basis for V is a set of linearly independent vectors that span V.
- Ex: Q1. What is V? Q2. Are vectors linearly independent?

$$\left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\}$$ yes

$$\left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$$ yes

$$\left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$$ yes

$$\left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right\}$$ no

Dimension of Vector Space

- Def: The dimension of vector space V is the number of vectors in any/a basis for V (or the # of independent vectors in V).
- Column/row rank: The dimension of column/row vector space, respectively.
- Ex: What's the column rank of matrix

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 0 \\ 2 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} ?$$

It's just another way to ask: what's the dimension of vector space

$$V = \left\{ \mathbf{v} = \alpha_1 \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} + \alpha_2 \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} + \alpha_3 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \ \alpha_i \in \mathbb{R} \right\} ?$$

Dimension of Vector Space (cont'd)

- Approach 1: By observation, we notice that any (and only) two pairs of vectors spanned V are linearly independent. Hence, we can immediately write out at least three bases:

$$\left\{\begin{bmatrix}1\\2\\1\end{bmatrix},\begin{bmatrix}1\\0\\1\end{bmatrix}\right\} \text{ or } \left\{\begin{bmatrix}1\\2\\1\end{bmatrix},\begin{bmatrix}0\\1\\0\end{bmatrix}\right\} \text{ or } \left\{\begin{bmatrix}1\\0\\1\end{bmatrix},\begin{bmatrix}0\\1\\0\end{bmatrix}\right\}$$

  Hence, the column rank of $\mathbf{X}$ or dimension of vector space V is 2.

- Approach 2: Define the three vectors to be $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$, respectively.

$$\begin{aligned} V &= \left\{\mathbf{v} = \alpha_1(\mathbf{v}_2 + 2\mathbf{v}_3) + \alpha_2\mathbf{v}_2 + \alpha_3\mathbf{v}_3\right\} \\ &= \left\{\mathbf{v} = (\alpha_1 + \alpha_2)\mathbf{v}_2 + (2\alpha_1 + \alpha_3)\mathbf{v}_3\right\}. \end{aligned}$$

$\mathbf{v}_2 \perp \mathbf{v}_3 \Rightarrow$ they are linearly independent. So the dim/rank is 2.

Projection of a Vector on a Unit Vector

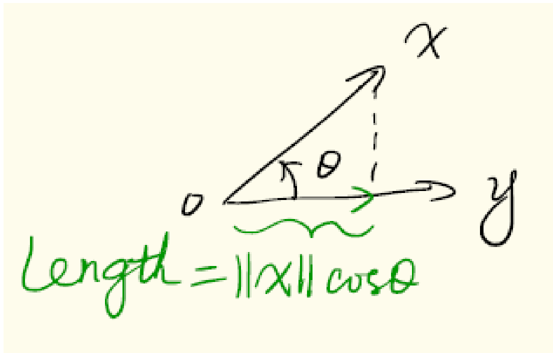
- Project a vector $\mathbf{x}$ on a unit vector $\mathbf{u}$:
  - Projection length is $\mathbf{u}^T\mathbf{x}$. (a number, with sign)
  - Projected vector is $(\mathbf{u}^T\mathbf{x})\mathbf{u}$. (a scaled vector along $\mathbf{u}$)

- Proof (projection length):

$$\mathbf{x}^T\mathbf{y} = \|\mathbf{x}\| \, \|\mathbf{y}\| \cos\theta$$

Divide both sides by $\|\mathbf{y}\|$ and let $\mathbf{u} = \mathbf{y}/\|\mathbf{y}\|$:

$$\mathbf{x}^T\left(\frac{\mathbf{y}}{\|\mathbf{y}\|}\right) = \|\mathbf{x}\| \cos\theta = \mathbf{u}^T\mathbf{x}.$$



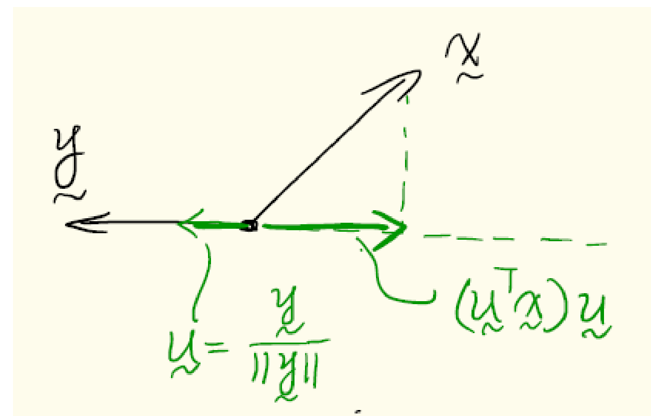
Projection One Vector on Another

- Project a vector $\mathbf{x}$ on a vector $\mathbf{y}$:
  - Projection length is $\boldsymbol{y}^T\mathbf{x}/||\boldsymbol{y}||$. (a number, with sign)
  - Projected vector is $(\boldsymbol{y}^T\mathbf{x})\boldsymbol{y}/||\boldsymbol{y}||^2$. (a scaled vector along $\mathbf{y}$)

- Proof (projected vector):

$$\text{Projection of } \mathbf{x} \text{ onto } \mathbf{y} = \left(\mathbf{u}^T\mathbf{x}\right)\mathbf{u}$$

Placing $\mathbf{u}$ by $\mathbf{y}/\left\|\mathbf{y}\right\|$, we obtain :



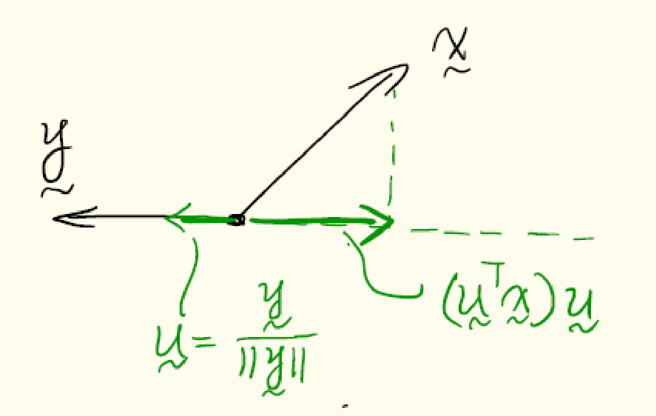
Projection One Vector on Another

- Project a vector $\mathbf{x}$ on a vector $\mathbf{y}$:
  - Projection length is $\boldsymbol{y}^T\mathbf{x}/||\boldsymbol{y}||$. (a number, with sign)
  - Projected vector is $(\boldsymbol{y}^T\mathbf{x})\boldsymbol{y}/||\boldsymbol{y}||^2$. (a scaled vector along $\mathbf{y}$)
- Proof (projected vector):

$$\text{Projection of } \mathbf{x} \text{ onto } \mathbf{y} = \left(\mathbf{u}^T\mathbf{x}\right)\mathbf{u}$$

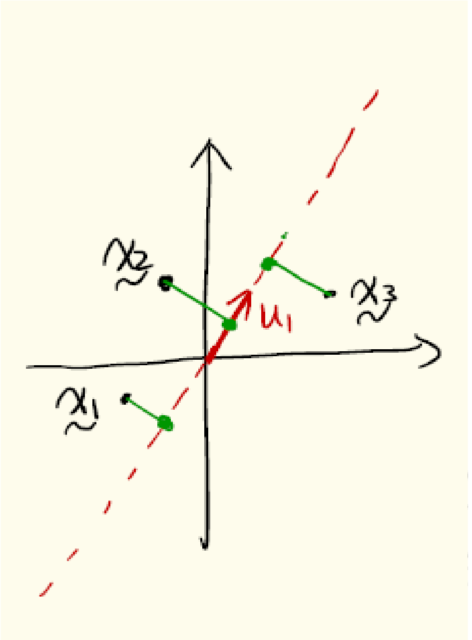
$$\text{Placing } \mathbf{u} \text{ by } \mathbf{y}/\left\|\mathbf{y}\right\|, \text{ we obtain :}$$

$$= \left[\left(\frac{\mathbf{y}}{\|\mathbf{y}\|}\right)^T \mathbf{x}\right]\frac{\mathbf{y}}{\|\mathbf{y}\|} = \left(\mathbf{y}^T\mathbf{x}\right)\mathbf{y}\Big/\left\|\mathbf{y}\right\|^2.$$



Projection of a Vector on a Unit Vector

- Example:



$$\mathbf{u}_1 = \left[\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right]^T$$
$$\mathbf{x}_1 = \left[-1, -\tfrac{1}{2}\right]^T$$
$$\mathbf{x}_2 = \left[-\tfrac{1}{2}, 1\right]^T$$
$$\mathbf{x}_3 = [2, 1]^T$$

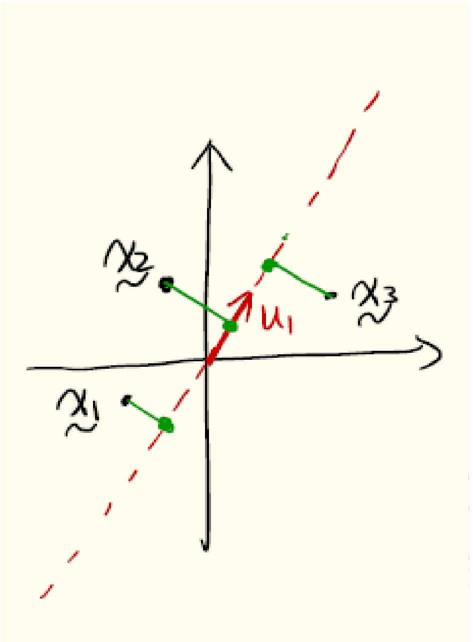
$$z_{11} = \mathbf{x}_1^T \mathbf{u}_1 =$$

$$z_{21} = \mathbf{x}_2^T \mathbf{u}_1 =$$

$$z_{31} = \mathbf{x}_3^T \mathbf{u}_1 =$$

Projection of a Vector on a Unit Vector

- Example:



$$\mathbf{u}_1 = \left[\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right]^T$$
$$\mathbf{x}_1 = \left[-1, -\tfrac{1}{2}\right]^T$$
$$\mathbf{x}_2 = \left[-\tfrac{1}{2}, 1\right]^T$$
$$\mathbf{x}_3 = [2, 1]^T$$

$$z_{11} = \mathbf{x}_1^T \mathbf{u}_1 = (-1) \cdot \frac{\sqrt{2}}{2} + \left(-\tfrac{1}{2}\right) \cdot \frac{\sqrt{2}}{2}$$

$$z_{21} = \mathbf{x}_2^T \mathbf{u}_1 = \frac{\sqrt{2}}{4}$$

$$z_{31} = \mathbf{x}_3^T \mathbf{u}_1 = \tfrac{3}{2}\sqrt{2}$$

Orthonormal Basis

- Def: A basis $\{\mathbf{a}_1, \ldots, \mathbf{a}_r\}$ for V is called orthonormal if r vectors are (i) pairwise orthogonal and (ii) have unit norms.
- Ex: Given a vector space

$$V = \left\{ \mathbf{v} = \alpha_1 \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} + \alpha_2 \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} + \alpha_3 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix},\ \alpha_i \in \mathbb{R} \right\}$$

$$\left\{ \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \right\} \qquad \left\{ \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right\} \qquad \left\{ \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right\} \qquad \left\{ \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \right\}$$

Orthonormal Basis

- Def: A basis $\{\mathbf{a}_1, \dots, \mathbf{a}_r\}$ for V is called orthonormal if r vectors are (i) pairwise orthogonal and (ii) have unit norms.
- Ex: Given a vector space

$$V = \left\{ \mathbf{v} = \alpha_1 \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} + \alpha_2 \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} + \alpha_3 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \ \alpha_i \in \mathbb{R} \right\}$$

$$\left\{ \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \right\}$$

Basis
Not orthogonal
Not unit vectors

$$\left\{ \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right\}$$

Basis
Not orthogonal
Not unit vectors

$$\left\{ \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right\}$$

Basis w/ orthogonal vectors.
Can normalize $[1 \ 0 \ 1]^T$ to obtain an orthonormal basis.

$$\left\{ \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \right\}$$

Not even a basis.
Why???

Orthogonal Matrix (or Orthonormal Matrix)

- Def: A square matrix $\mathbf{P}$ is orthogonal if and only if its columns (or rows) constitute an orthonormal basis.
- Properties:
  - $\mathbf{P}^{\mathrm{T}}\mathbf{P} = \mathbf{P}\mathbf{P}^{\mathrm{T}} = \mathbf{I}$
  - $\mathbf{P}^{-1} = \mathbf{P}^{\mathrm{T}}$
- Ex:

$$\mathbf{P} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1/\sqrt{2} & 1/\sqrt{2} \\ 1 & 0 & 0 \\ 0 & 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \quad \mathbf{P}^{\mathbf{T}} = \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \mathbf{v}_3^T \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1/\sqrt{2} & 0 & 1/\sqrt{2} \\ 1/\sqrt{2} & 0 & -1/\sqrt{2} \end{bmatrix}$$

$$\mathbf{P}\mathbf{P}^T = \begin{bmatrix} 0 + \left(\frac{1}{\sqrt{2}}\right)^2 + \left(\frac{1}{\sqrt{2}}\right)^2 & 0 & \frac{1}{2} - \frac{1}{2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{I}$$

(Direct evaluation)

$$\mathbf{P}^T\mathbf{P} = \begin{bmatrix} \mathbf{v}_1^T \\ \hdashline \mathbf{v}_2^T \\ \hdashline \mathbf{v}_3^T \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 & \vdots & \mathbf{v}_2 & \vdots & \mathbf{v}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1^T\mathbf{v}_1 & 0 & 0 \\ 0 & \mathbf{v}_2^T\mathbf{v}_2 & 0 \\ 0 & 0 & \mathbf{v}_3^T\mathbf{v}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{I}$$

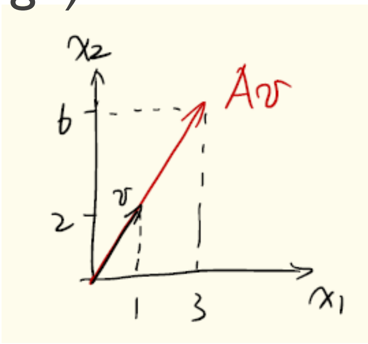
(Block trick)

Eigenvector and Eigenvalue

- Def: Let $\mathbf{A}$ be an n-by-n matrix. A nonzero vector $\mathbf{v}$ is called an eigenvector of $\mathbf{A}$ if $\mathbf{Av} = \lambda\mathbf{v}$. Here, λ is called an eigenvalue of $\mathbf{A}$, and $\mathbf{v}$ is eigenvector corresponding to eigenvalue λ.
- Prefix "eigen" means "characteristic."
- The characteristic is of $\mathbf{A}$, not of $\mathbf{v}$.
- Physical interpretation: $\mathbf{v}$ is invariant to operator $\mathbf{A}$, which means that $\mathbf{A}$ acts on $\mathbf{v}$ can only change its length (and sign) but not orientation.
- Ex: Let $\mathbf{A} = \begin{bmatrix} 3 & 0 \\ 8 & -1 \end{bmatrix}$, $\mathbf{v} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$.

  Since $\mathbf{Av} = \begin{bmatrix} 3 \cdot 1 + 0 \cdot 2 \\ 8 \cdot 1 - 1 \cdot 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \end{bmatrix} = 3\mathbf{v}$,

  the eigenvalue $\lambda = 3$.



Eigendecomposition for Symmetric Matrices

- Def: A square matrix $\mathbf{A}$ is symmetric if $\mathbf{A} = \mathbf{A}^T$.
- Thm: A p-by-p symmetric matrix $\mathbf{R}$ can be diagonalized by an orthogonal matrix $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_p]$. The following statements are equivalent:

1. $\mathbf{R} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T$

$$= \left[\begin{array}{c:c:c} \mathbf{v}_1 & \cdots & \mathbf{v}_p \end{array}\right] \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_p \end{bmatrix} \left[\begin{array}{c} \mathbf{v}_1^T \\ \hdashline \vdots \\ \hdashline \mathbf{v}_p^T \end{array}\right]$$

$$= [\lambda_1 \mathbf{v}_1 \cdots \lambda_p \mathbf{v}_p] \begin{bmatrix} \mathbf{v}_1^T \\ \vdots \\ \mathbf{v}_p^T \end{bmatrix}$$

$$= \sum_{i=1}^{p} \lambda_i \mathbf{v}_i \mathbf{v}_i^T$$

2. $\mathbf{R}\mathbf{V} = \mathbf{V}\boldsymbol{\Lambda} = [\mathbf{v}_1 \cdots \mathbf{v}_p] \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_p \end{bmatrix}$

3. $\mathbf{R}\mathbf{v}_i = \lambda_i \mathbf{v}_i, \quad i = 1, \ldots, p.$

Eigendecomposition Using Matlab

- Ex: Use Matlab to decompose matrix $\mathbf{R} = \begin{bmatrix} 1 & 4 & 5 \\ 4 & -3 & 0 \\ 5 & 0 & 7 \end{bmatrix}$
- Source code:

```
R = [1 4 5; 4 -3 0; 5 0 7];
[V, Lambda] = eig(R);  % use built-in function for eigendecomposition

for j = 1 : size(Lambda, 1)
  if norm(R * V(:, j) - Lambda(j, j) * V(:, j)) < 1e-5  % verify result
    disp(['Eigenvector-value pair ' int2str(j) ' verified.'])
  end
end
```

Can you numerically verify the 3 equivalent expressions on the previous slide?

- Output:

```
V =
    0.5952    0.6072    0.5263
   -0.7707    0.6167    0.1601
   -0.2274   -0.5009    0.8351
```

```
Lambda =
   -6.0892         0         0
         0    0.9383         0
         0         0   10.1509
```

Eigendecomposition by Hand (optional)

- Thm: Eigenvalues are roots of the *characteristic polynomial* $\det(\mathbf{A} - \lambda\mathbf{I})$.
- Ex: $\mathbf{A} = \begin{bmatrix} 5 & 4 \\ 1 & 2 \end{bmatrix}$

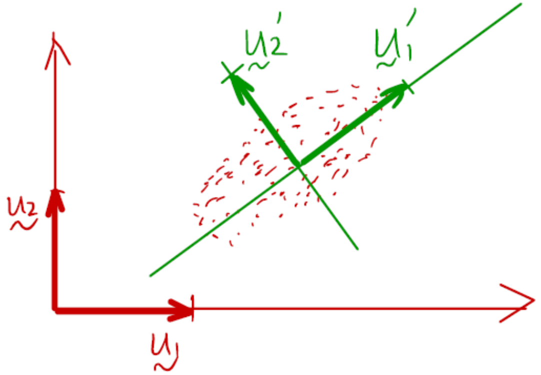
$$0 = \det\left(\begin{bmatrix} 5 & 4 \\ 1 & 2 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) = \det\left(\begin{bmatrix} 5-\lambda & 4 \\ 1 & 2-\lambda \end{bmatrix}\right)$$
$$= \lambda^2 - 7\lambda + 6 \Rightarrow \lambda_1 = 6,\ \lambda_2 = 1.$$

$$\text{For } \lambda_1 = 6, \quad (\mathbf{A} - \lambda_1 \mathbf{I})\,\mathbf{v} = 0 \Leftrightarrow \begin{cases} -v_1 + 4v_2 = 0 \\ v_1 - 4v_2 = 0 \end{cases} \Rightarrow \mathbf{v} = \begin{bmatrix} 4 \\ 1 \end{bmatrix}$$

$$\text{For } \lambda_2 = 1, \quad (\mathbf{A} - \lambda_2 \mathbf{I})\,\mathbf{v} = 0 \Leftrightarrow \begin{cases} 4v_1 + 4v_2 = 0 \\ v_1 + v_2 = 0 \end{cases} \Rightarrow \mathbf{v} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Nonunique solutions for underdetermined systems

Principal Component Analysis (Unsupervised Learning)



Learning objectives

- Explain the two equivalent goals of PCA
- Implement the PCA algorithm and visualize the results

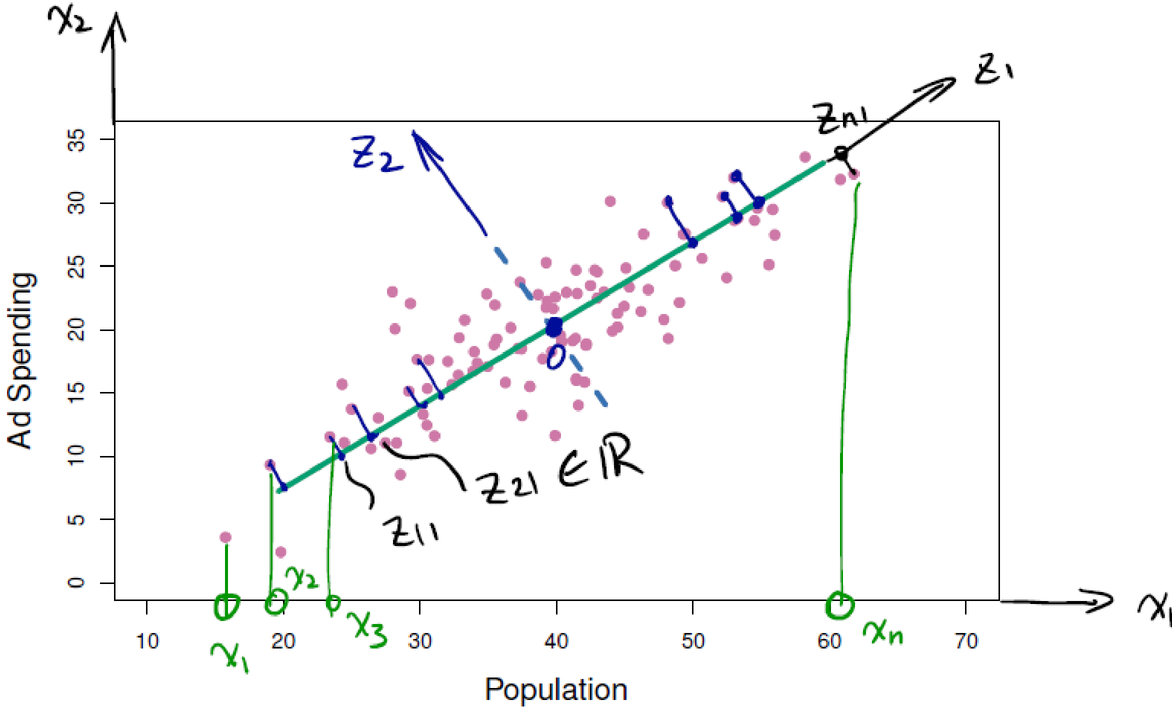
(Ref: 10.2 of James et al. 2013, 12.2 of Murphy 2012. Extra ref: 12.1 of Bishop 2006.)

Unsupervised Learning

- Def: Learns from a set of unlabeled data to discover interesting patterns.
  - Visualize the data in an informative way.
  - Discover subgroups among observations/variables.
- Examples:
  - Movies grouped by ratings and behavioral data from viewers.
  - Groups of shoppers characterized by browsing & purchasing histories.
  - Subgroups of breast cancer patients grouped by gene expressions.
  - Tweets grouped by latent topics inferred from the use of words.

PCA: Two Equivalent Goals

- Goals, i.e., cost/loss/objective functions, of PCA: (1) maximize variance, and (2) minimize error.



(James, Witten, Hastie, & Tibshirani, 2013)

PCA Objective 1: Maximizing Variance

- Maximize variance: Project data onto a lower-dimensional subspace while maximizing the variance of the projected data.

- Details:

$\{\mathbf{x}_i\}_{i=1}^n, \quad \mathbf{x}_i \in \mathbb{R}^p$	A dataset of n data points
$\mathbf{u}_1 \; : \; \lVert \mathbf{u}_1 \rVert^2 = 1$	Unit vector / direction $\mathbf{u}_1$ (to figure out!)
$z_{i1} = \mathbf{u}_1^T \mathbf{x}_i$	Projection of $\mathbf{x}_i$ along $\mathbf{u}_1$

- Naming:
  - z_{i1}—score, coefficient, transformed coefficient, weight, projected values, …
  - $\mathbf{u}_1$—loading, (1st) principal component vector, …

$$\text{Spread} = \frac{1}{n-1}\sum_{i=1}^{n}\left(z_{i1} - \overline{z_1}\right)^2 \quad :$$

Sample variance measures spread of the projected data along $\mathbf{u}_1$.

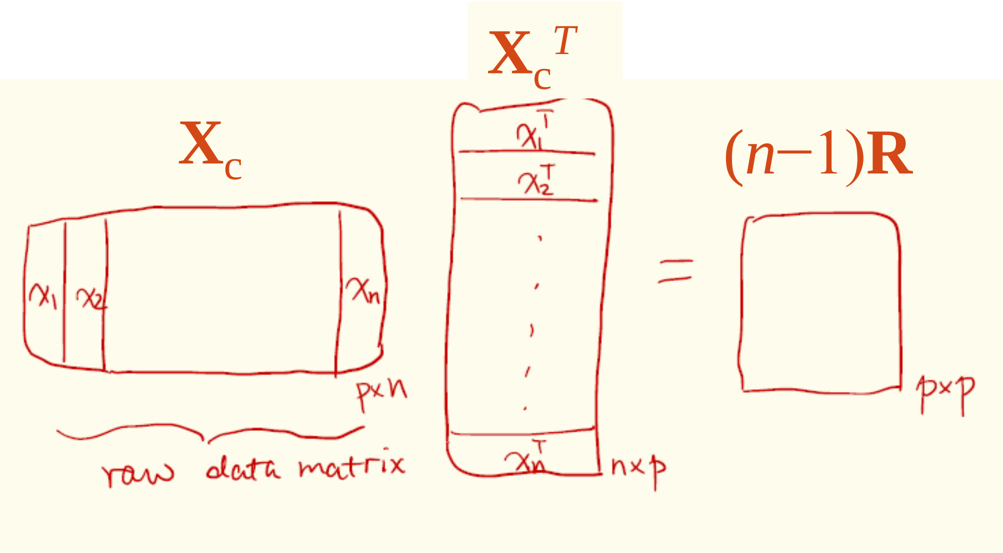
where $\overline{z_1} \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} z_{i1}$ is the *sample mean*.

$$= \frac{1}{n-1}\sum_{i=1}^{n}\left(\mathbf{u}_1^T\mathbf{x}_i - \mathbf{u}_1^T\bar{\mathbf{x}}\right)^2$$

$$= \frac{1}{n-1}\sum_{i=1}^{n}\mathbf{u}_1^T\left(\mathbf{x}_i - \bar{\mathbf{x}}\right)\left(\mathbf{x}_i - \bar{\mathbf{x}}\right)^T\mathbf{u}_1$$

$$= \mathbf{u}_1^T \underbrace{\left[\frac{1}{n-1}\sum_{i=1}^{n}\left(\mathbf{x}_i - \bar{\mathbf{x}}\right)\left(\mathbf{x}_i - \bar{\mathbf{x}}\right)^T\right]}_{\mathbf{R},\ \text{sample covariance}} \mathbf{u}_1$$

Matrix form for calculating $\mathbf{R}$:

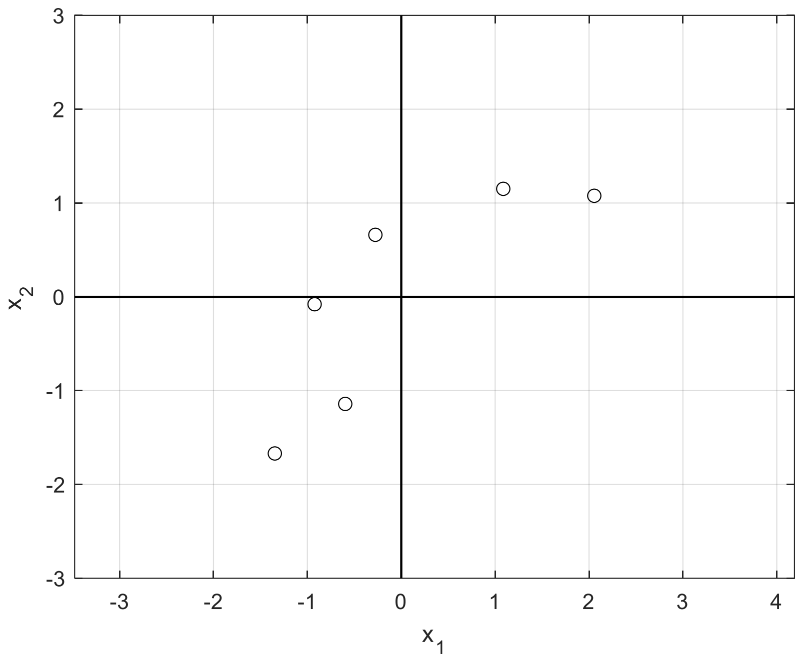


(Assuming all $\mathbf{x}_i$ are already "centered," i.e., $\mathbf{x}_i \leftarrow \mathbf{x}_i - \bar{\mathbf{x}},\ \forall i$.)

Source code:

```
plot(X_c(1, :), X_c(2, :), 'ko');
xlabel('x_1'); ylabel('x_2');
axis([-3 3 -3 3]); axis equal;
hold on;

R = (X_c * X_c') / (n-1);
```



Output:

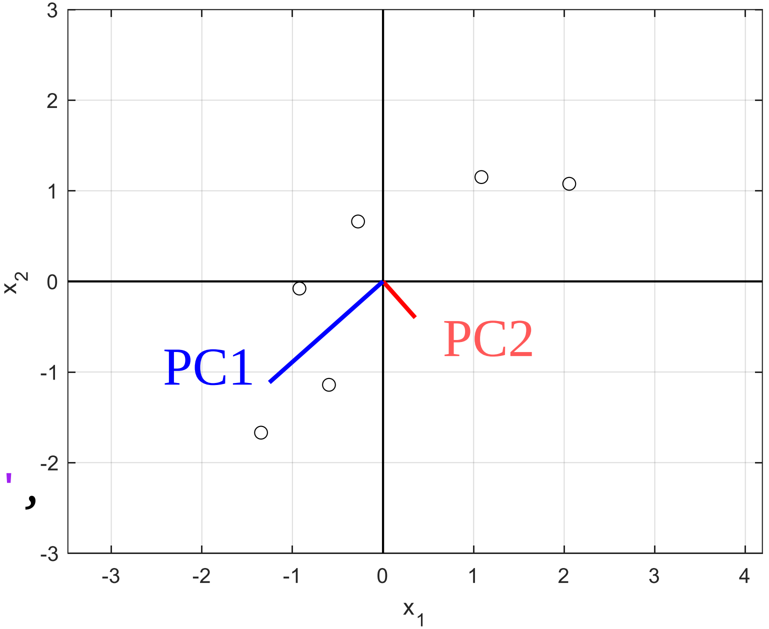
```
X_c =
   -0.9229    1.0864   -1.3466    2.0556   -0.5967   -0.2758
   -0.0789    1.1515   -1.6695    1.0773   -1.1415    0.6611

R =
    1.7006    1.2570
    1.2570    1.4040
```

Source code:

```
[U, Lambda] = eig(R);
eigenvalues = diag(Lambda);
color_arr = ['r', 'b'];

for k = 1 : size(U, 2)
  u = U(:, k);
  len = sqrt(eigenvalues(k));
  plot([0 len*u(1)], [0 len*u(2)], 'LineWidth',
'color', color_arr(k));
end
```



Output:

```
U =
      0.6644   -0.7474
     -0.7474   -0.6644
```

PC2 PC1

```
Lambda =
      0.2865         0
           0    2.8181
```

λ_2 λ_1

(Optional)

$$\text{maximize}_{\mathbf{u}} \quad \mathbf{u}^T\mathbf{R}\mathbf{u} \qquad \text{subject to } \|\mathbf{u}\| = 1$$

Use Lagrange, we have $J(\mathbf{u}) = \mathbf{u}^T\mathbf{R}\mathbf{u} + \lambda\left(1 - \mathbf{u}^T\mathbf{u}\right)$. Taking the gradient $\nabla_{\mathbf{u}}$ (i.e., a vector of partial derivatives, $[\frac{\partial}{\partial u_1}, \dots, \frac{\partial}{\partial u_p}]^T$) for $J(\mathbf{u})$ and set it to the $\mathbf{0}$ vector

$$\nabla_{\mathbf{u}} J(\mathbf{u}) = 2\mathbf{R}^T\mathbf{u} + \lambda(-2\mathbf{u}) = \Big|_{\mathbf{u}=\hat{\mathbf{u}}} \quad \mathbf{0},$$

we obtain $\mathbf{R}\hat{\mathbf{u}} = \lambda\hat{\mathbf{u}}$. Left multiply $\hat{\mathbf{u}}^T$ to both sides, we have

$$\hat{\mathbf{u}}^T\mathbf{R}\hat{\mathbf{u}} = \hat{\mathbf{u}}^T\lambda\hat{\mathbf{u}} = \lambda\|\hat{\mathbf{u}}\|^2 = \lambda.$$

The cost function is then simplified to finding the largest λ, or largest eigenvalue of $\mathbf{R}$. $\hat{\mathbf{u}}$ is the eigenvector that corresponds to the largest eigenvalue.

Vector calculus cheat sheet (p. 521–527): https://www.cs.cmu.edu/~epxing/Class/10701-08s/recitation/mc.pdf

PCA: Forward Transform and Reconstruction

i) Analysis/Forward Transform:

Also known as Karhunen–Loeve Transform (KLT)

$$\begin{bmatrix} z_{i1} \\ z_{i2} \\ \vdots \\ z_{in} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1^T \\ \hline \mathbf{u}_2^T \\ \hline \vdots \\ \hline \mathbf{u}_n^T \end{bmatrix} \begin{bmatrix} \\ \mathbf{x}_i \\ \\ \end{bmatrix}$$

$$\mathbf{z}_i = \mathbf{U}^T \mathbf{x}_i$$

ii) Synthesis/Reconstruction:

$$\mathbf{x}_i = \mathbf{U}\mathbf{z}_i = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_n \end{bmatrix} \begin{bmatrix} z_{i1} \\ \vdots \\ z_{in} \end{bmatrix} = \sum_{k=1}^{n} z_{ik} \mathbf{u}_k$$

Analysis example:

$$\underbrace{\begin{bmatrix} -1.58 \\ -0.14 \end{bmatrix}}_{\mathbf{z}_i} = \underbrace{\begin{bmatrix} -0.75 & 0.66 \\ -0.66 & -0.75 \end{bmatrix}^T}_{\mathbf{U}^\mathbf{T}} \underbrace{\begin{bmatrix} 1.09 \\ 1.15 \end{bmatrix}}_{\mathbf{x}_i}$$

Synthesis example:

$$\underbrace{\begin{bmatrix} 1.09 \\ 1.15 \end{bmatrix}}_{\mathbf{x}_i} = \underbrace{\begin{bmatrix} -0.75 & 0.66 \\ -0.66 & -0.75 \end{bmatrix}}_{\mathbf{U}} \underbrace{\begin{bmatrix} -1.58 \\ -0.14 \end{bmatrix}}_{\mathbf{z}_i}$$

$$= -1.58 \begin{bmatrix} -0.75 \\ -0.66 \end{bmatrix} - 0.14 \begin{bmatrix} 0.66 \\ -0.75 \end{bmatrix}$$

$$= \begin{bmatrix} 1.19 \\ 1.04 \end{bmatrix} + \begin{bmatrix} -0.09 \\ 0.11 \end{bmatrix}$$

Contribution from PC2 is **small**

Reconstruction Using Dominant PCs

(Murphy 2012)

Raw image

PC1 of MNIST

Recon w/ 2 PCs

Recon w/ 10 PCs

PC2

PC3

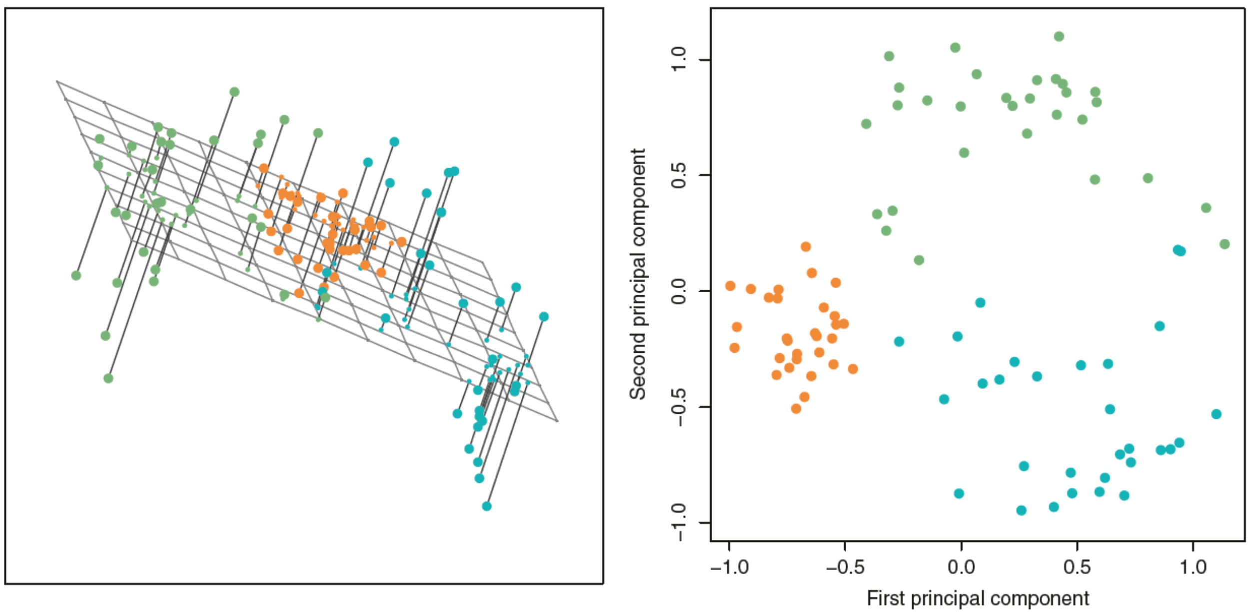
Recon w/ 100 PCs

Recon w/ 506 PCs

- Each image of 50x50 is stacked into a column vector of length 2,500.
- Sample covariance matrix will be of size 2,500x2,500.
- Eigenvectors/principal components (PCs) of length 2,500 are **reshaped** to 50x50 for display. May call them "eigen-images."

PCA Objective 2: Minimizing Error

- Approximate the data points using a presentation in a lower-dimensional subspace.



(James, Witten, Hastie, & Tibshirani, 2013)

(Optional)

Assume $\mathbf{x}_i$'s are centered, i.e., $\mathbf{x}_i \leftarrow \mathbf{x}_i - \bar{\mathbf{x}}$, $\forall i$.

$$J(\mathbf{u}_1, z_{i1}) = \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i - z_{i1}\mathbf{u}_1\|^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - z_{i1}\mathbf{u}_1)^T(\mathbf{x}_i - z_{i1}\mathbf{u}_1)$$

$$= \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i^T\mathbf{x}_i - 2z_{i1}\mathbf{x}_i^T\mathbf{u}_1 + z_{i1}^2\mathbf{u}_1^T\mathbf{u}_1)$$

$$\frac{\partial}{\partial z_{j1}}J = \frac{1}{n}(-2\mathbf{x}_j^T\mathbf{u}_1 + 2z_{j1}\underbrace{\mathbf{u}_1^T\mathbf{u}_1}_{=1}) = \Big|_{z_{j1}=\hat{z}_{j1}} 0 \Rightarrow \hat{z}_{j1} = \mathbf{u}_1^T\mathbf{x}_j$$

(Does this result look familiar?)

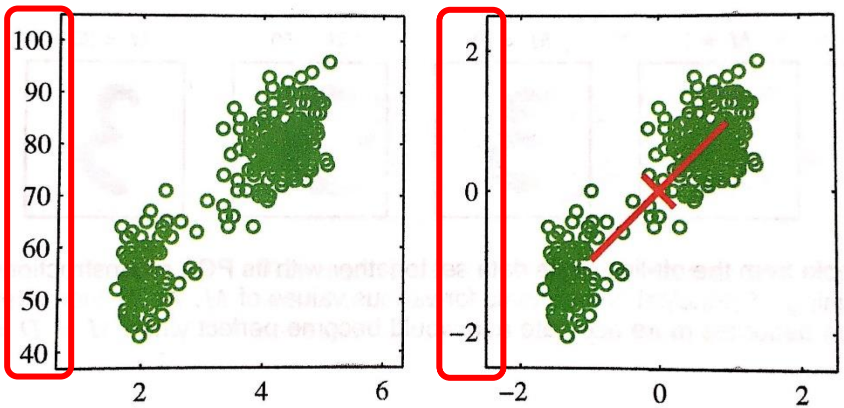
$$J = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i^T\mathbf{x}_i - 2z_{i1}^2 + z_{i1}^2) \quad \text{(skip the hat of } z_{i1} \text{ for simplicity)}$$

$$\min_{\mathbf{u}_1} J = \max_{\mathbf{u}_1}\sum_{i=1}^{n} z_{i1}^2 = \underline{\text{maximize the spread!}}$$

Same as the Objective 1

PCA's Caveat: Proper Standardization May be Needed

◆ If coordinates of $\mathbf{x}_j = [x_{1,j}, \ldots, x_{p,j}]^T$ have different **units**, maximal variance direction may be biased toward $x_{i,j}$ with largest magnitude.



(Bishop 2006)

- Why is standardization needed in this case?
- Do the hand-written digit and face recognition need standardization?

◆ When proper standardization of coordinate/variable/feature i is needed:

$$\tilde{x}_{ij} = \frac{x_{ij} - \bar{x}_{i\cdot}}{\sqrt{\frac{1}{n}\sum_{j=1}^{n}(x_{ij} - \bar{x}_{i\cdot})^2}}, \qquad i = 1, \ldots, p.$$

Should standardize along the feature/horizontal direction rather than within each data point.

PCA: Applications and Beyond

- PCA is lightweight yet powerful. Should be tried before applying more sophisticated tools.

- Modern replacement of PCA:
  - Data visualization: t-SNE, UMAP.
  - Dimensionality reduction: Nonlinear dimensionality reduction algorithms.
  - Lossy data compression: Data-independent transforms tailored for data following certain statistical behaviors.
  - Feature extraction: Topic modeling (unsupervised), CNN self-learned feature extraction (supervised).

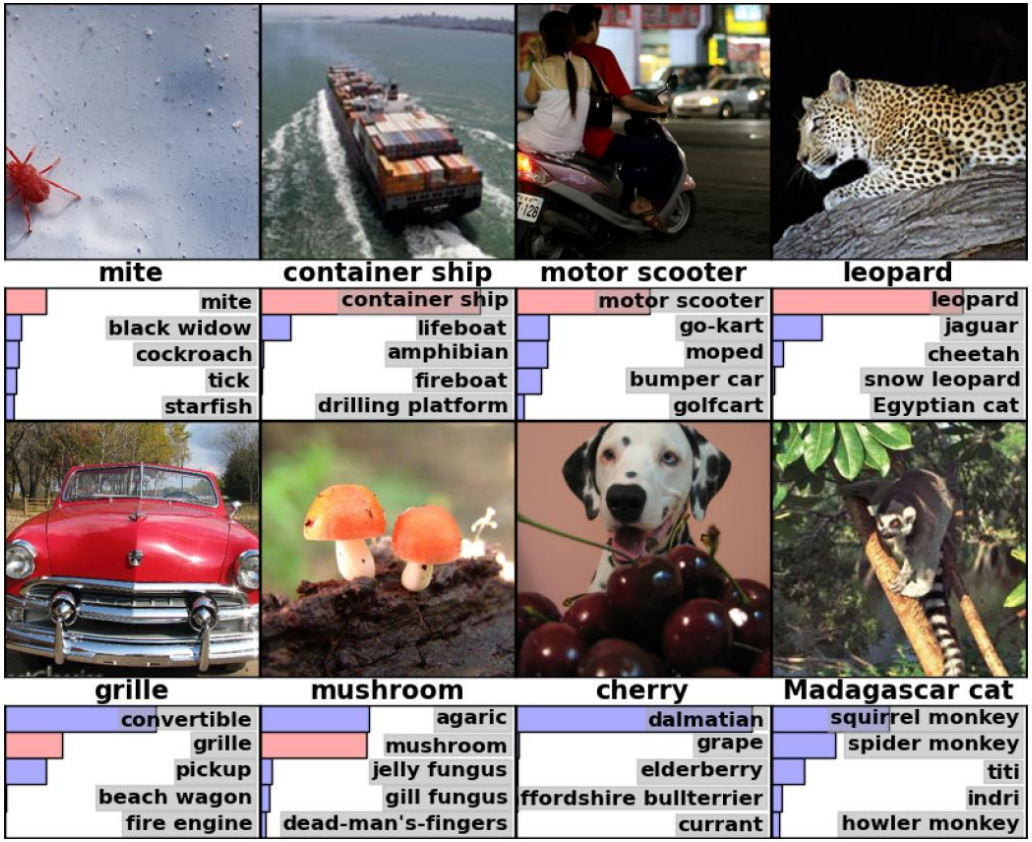
Linear Regression and Prediction (Supervised Learning)

Learning objectives

- Interpret regression problem mathematically and geometrically
- Apply linear regression to learning problems without overfit

(A comprehensive treatment of basic linear regression can be found in Scheffe Ch1, available on the library's course reserves.)

Supervised Learning: Classification



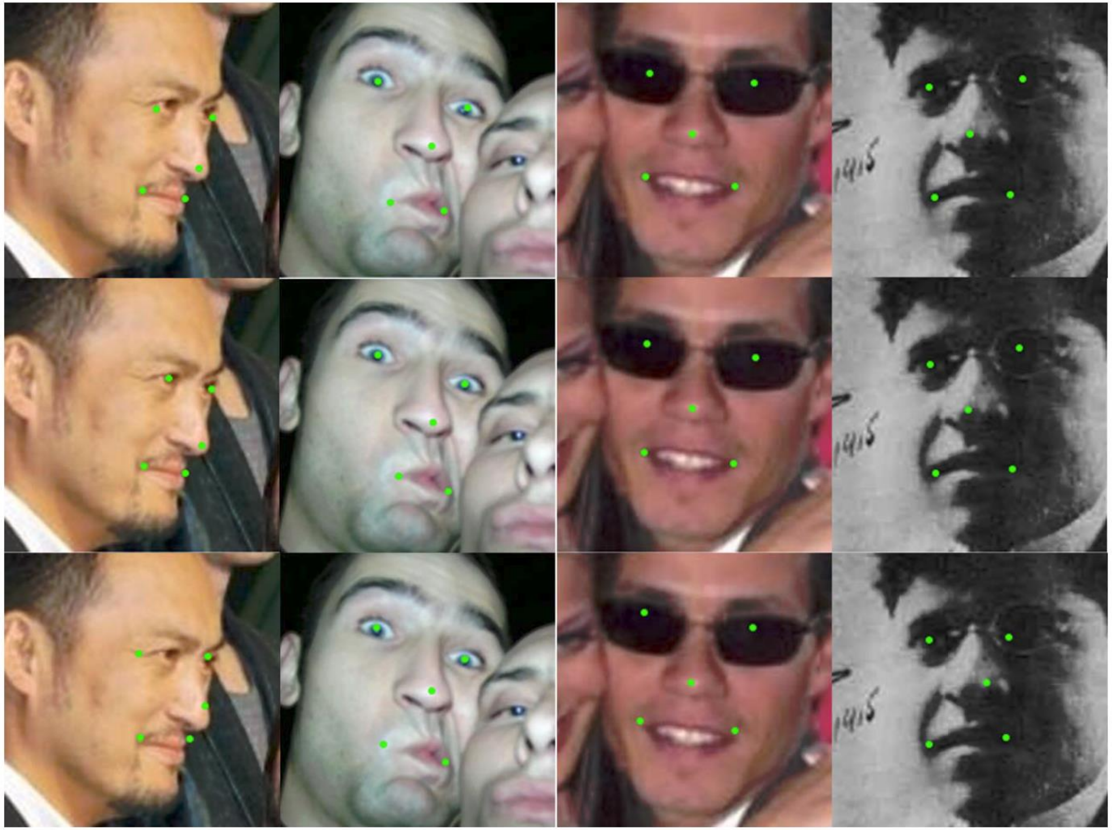
<u>Goal of classification:</u>
Assign a categorical/ qualitative label, or a class, to an given input.

← Given an image, it returns the class label.

Optionally, provide a "confidence score."

Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, ImageNet Classification with Deep Convolutional Neural Networks, *NIPS*, 2012.

Supervised Learning: Regression



<u>Goal of regression:</u>
Assign a number to each input.

Loosely, ML people also call it "label."

🡐 Given a facial image, it returns the 2D location for each key point of the face.

Yi Sun, Xiaogang Wang, Xiaoou Tang, Deep Convolutional Network Cascade for Facial Point Detection, *CVPR*, 2013.

Supervised Learning: Definition

- **Terminologies:**
  - Training data: $\mathcal{D}_{\text{tr}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$
  - Test data: $\mathcal{D}_{\text{te}} = \{(\mathbf{x}_i, y_i)\}_{i=n+1}^{n+m}$
  - Learned model: $y = f(\mathbf{x})$

- **Goal**: Given a set of training data $\mathcal{D}_{\text{tr}}$ as the inputs, we would like to compute a learned model $y = f(\mathbf{x})$ such that it can generate accurate predicted outputs

$$\hat{y}_i = f(\mathbf{x}_i), \quad i = n+1, \ldots, n+m,$$

from a set of new inputs $\{\mathbf{x}_i\}_{i=n+1}^{n+m}$ of the test data $\mathcal{D}_{\text{te}}$ whose labels $\{y_i\}_{i=n+1}^{n+m}$ have never been taken into account when the model is computed.

Quantifying the Accuracy of Prediction

- Quantify the accuracy of the learned model by a *loss function* (or cost/objective function), based on predicted output, $\hat{y}_i$, and the true output, y_i, namely, $L(\hat{\mathbf{y}}, \mathbf{y})$

- A typical choice for the loss function for a continuous-valued output is the *mean squared error*:

$$L(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{m} \sum_{i=n+1}^{n+m} (\hat{y}_i - y_i)^2$$

- Key ML assumption: Test data shouldn't have been seen before (at the training stage), or there will be overfit.

Simplest Example: Linear Model

Data: $(x_i, Y_i), \quad i = 1, \ldots, n$

Model: $Y_i = \beta_0 + \beta_1 x_i + e_i$

Simplest Example: Linear Model

Data: $(x_i, Y_i), \quad i = 1, \ldots, n$

random $\mathbb{E}[e_i] = 0$

Model: $Y_i = \beta_0 + \beta_1 x_i + e_i$

intercept

noise: measurement noise, biological variation

dependent var.
/observation

independent var./predictor

$\boldsymbol{\theta} = [\beta_0, \beta_1]^T$ is the parameter vector/weights.

$\mathbb{E}[Y_i] = \beta_0 + \beta_1 x_i =$ linear combination of unknowns β_0 and β_1 with known coefficient 1 and x_i.

Linear Model in Matrix-Vector Form

$$\boxed{\begin{aligned} &Y_i = \beta_0 + \beta_1 x_i + e_i, \\ &i = 1, \ldots, n. \end{aligned}}$$

$$\mathbf{Y} = \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix}_{n\times 1}, \quad \mathbf{X} = \begin{bmatrix} \underbrace{\begin{matrix} 1 \\ \vdots \\ 1 \end{matrix}}_{\mathbb{1}} & \underbrace{\begin{matrix} x_1 \\ \vdots \\ x_n \end{matrix}}_{\mathbf{x}} \end{bmatrix}_{n\times 2}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}_{2\times 1}, \quad \mathbf{e} = \begin{bmatrix} e_1 \\ \vdots \\ e_n \end{bmatrix}_{n\times 1}$$

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}$$ "Matrix–vector form"

data matrix

Linear Model with Multiple Predictors / Features

- Multiple (Linear) Regression Model:

$$Y_i = \sum_{j=1}^{p} x_{ij}\beta_j + e_i, \quad i = 1, \ldots, n.$$

$$\mathbf{Y}_{n\times 1} = \mathbf{X}_{n\times p}\boldsymbol{\beta}_{p\times 1} + \mathbf{e}_{n\times 1}$$

vector of random elements

Linear Regression Example

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + e_i, \quad i = 1, \ldots, 50.$$

Y_i : grade

x_{i1} : time spent on HW

x_{i2} : time spent on review

$$\begin{bmatrix} Y_1 \\ \vdots \\ Y_{50} \end{bmatrix} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} \\ \vdots & \vdots & \vdots \\ 1 & x_{50,1} & x_{50,2} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} e_1 \\ \vdots \\ e_{50} \end{bmatrix}$$

How to estimate model parameters β_0, β_1, and β_2? Least-Squares!

Linear Regression Example

$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + e_i, \quad i = 1, \ldots, 50.$$

Y_i : grade

x_{i1} : time spent on HW

x_{i2} : time spent on review

$$\begin{bmatrix} Y_1 \\ \vdots \\ Y_{50} \end{bmatrix} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} \\ \vdots & \vdots & \vdots \\ 1 & x_{50,1} & x_{50,2} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} e_1 \\ \vdots \\ e_{50} \end{bmatrix}$$

How to estimate model parameters β_0, β_1, and β_2? Least-Squares!

Least-Squares for Parameter Estimation

Problem Setup: $\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}$, where $\mathbf{X} \triangleq [\boldsymbol{\xi}_1, \cdots, \boldsymbol{\xi}_p]$.

Estimate $\boldsymbol{\beta}$ such that $J(\boldsymbol{\beta}) = \|\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\|^2$ is minimized.

$$\text{or } J(\boldsymbol{\beta}) = \sum_{i=1}^{n} (Y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2$$

This is called the *least-squares* procedure.

Least-Squares via Vector Calculus

Recall: $J(\boldsymbol{\beta}) = \|\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\|^2$

Method 1: $\nabla_{\boldsymbol{\beta}} J(\boldsymbol{\beta}) = \Big|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}} 0,$

$$\nabla_{\boldsymbol{\beta}} J(\boldsymbol{\beta}) = 2\left[-\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})\right] = \Big|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}} \mathbf{0}$$

$$\boxed{\mathbf{X}^T\mathbf{Y} = \mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}}} \qquad \boxed{\mathbf{X}^T(\mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}}) = \mathbf{0}}$$

(Error orthogonal to data)

<u>Normal Equation (N.E.)</u>

Vector calculus cheat sheet (p. 521–527): https://www.cs.cmu.edu/~epxing/Class/10701-08s/recitation/mc.pdf

Least-Squares via Partial Differentiation (optional)

If linear algebra is not used, the derivation can be much more involved:

Method 2 :

$\boxed{\text{Recall: } J(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(Y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2}$

$$\frac{\partial J}{\partial \beta_k} = \sum_{i=1}^{n} 2\Big(Y_i - \sum_{j=1}^{p} x_{ij}\beta_j\Big) \underbrace{\frac{\partial}{\partial \beta_k}\Big(-\big(\cdots + x_{ik}\beta_k + \cdots\big)\Big)}_{-x_{ik}}$$

$$= \big|_{\beta_j=\hat{\beta}_j} 0, \quad k = 1, \cdots, p$$

$$\Longleftrightarrow \sum_i Y_i x_{ik} = \sum_i \sum_j x_{ij}\hat{\beta}_j x_{ik} \Longleftrightarrow \boxed{\mathbf{X}^T\mathbf{Y} = \mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}}}$$

Normal Equation (N.E.)

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

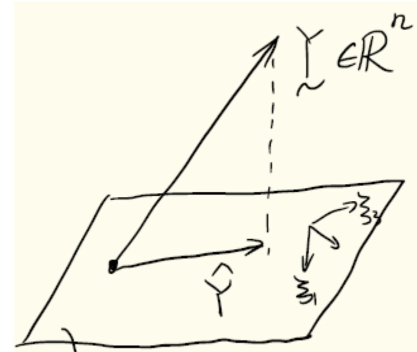
where $\mathbf{X}^T\mathbf{Y} = \left[\sum_{i=1}^{n} x_{ik}Y_i\right]_{p\times 1}$, $\mathbf{X}^T\mathbf{X} = \left[\sum_{i=1}^{n} x_{ij}x_{ik}\right]_{p\times p}$

$$\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}} = \left[\sum_{j=1}^{p}\left(\sum_{i=1}^{n} x_{ij}x_{ik}\right)\hat{\beta}_j\right]_{p\times 1}$$

Geometric Interpretation of Least-Squares (LS)

- Lemma: The LS procedure finds a vector $\widehat{\boldsymbol{\beta}}$ such that
  - $\widehat{\mathbf{Y}} = \mathbf{X}\widehat{\boldsymbol{\beta}}$ is as close as possible to $\mathbf{y}$, or
  - $(\mathbf{Y} - \widehat{\mathbf{Y}}) \perp \mathcal{C}(\mathbf{X})$.
- Note $\mathcal{C}(\mathbf{X}) = \{\mathbf{Xb}, \mathbf{b} \in \mathbb{R}^p\}$

$$
\begin{aligned}
& (\mathbf{Y} - \hat{\mathbf{Y}}) \perp \mathcal{C}(\mathbf{X}) \\
\iff & (\mathbf{Y} - \hat{\mathbf{Y}}) \perp \mathbf{Xb}, \quad \forall \mathbf{b} \in \mathbb{R}^p \\
\iff & \boldsymbol{\xi}_j^T (\mathbf{Y} - \hat{\mathbf{Y}}) = 0, \quad j = 1, \cdots, p \\
\iff & [\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_p]^T (\mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}}) = \mathbf{0} \\
\iff & \mathbf{X}^T \mathbf{Y} = \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\beta}}
\end{aligned}
$$



$\mathcal{C}(X) = \{X\underset{\sim}{b}, \underset{\sim}{b} \in \mathbb{R}^p\}$

$X \overset{\text{def}}{=} [\underset{\sim}{\xi_1}, \underset{\sim}{\xi_2}, \cdots, \underset{\sim}{\xi_p}]$

$\dim(\mathcal{C}(X)) = r \le p$

Properties of Least-Square Estimate

If $\operatorname{rank}(\mathbf{X}) \triangleq r = p$ ① $\boxed{\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}}$ is unique soluntion.

$$\mathbb{E}[\hat{\boldsymbol{\beta}}] = \mathbb{E}[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}] = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{X}\boldsymbol{\beta}) = \boldsymbol{\beta} \text{ (unbiased)}$$

② $\hat{\mathbf{Y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \underbrace{\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T}\mathbf{Y} = \mathbf{H}\mathbf{Y}$

$\mathbf{H}$: "hat" matrix, or "orthogonal projector." $\mathbf{H}^n = \mathbf{H}$. Why?

Ex: Linear Model for Learning and Prediction

- Training data (3 data points / a *random sample* of size 3):
  - *Feature*/*predictor* 1: (2, 1, 1). Feature/predictor 2: (1, 2, 1).
  - Labels: (1, 1, 1).
- Test data (2 data points / a random sample of size 2):
  - Feature 1: (1.2, 1.8). Feature 2: (0.9, 1.3).
  - Labels: (0.9, 0.8).
- Tasks:
  a) Learn a linear model without intercept.
  b) Using drawing to illustrate the data and learned model.
  c) Evaluate the mean squared errors (MSEs) of training and testing.

a) $\mathbf{X} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 1 & 1 \end{bmatrix}$ $\quad \mathbf{Y} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ $\quad \mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}$ $\quad (\mathbf{X}, \mathbf{Y})$: training data

Estimated/ trained model parameters:

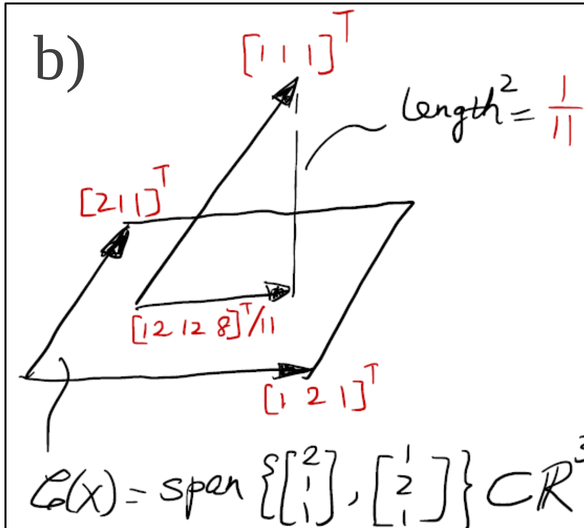
$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}$$

$$= \left(\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix}\begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 1 & 1 \end{bmatrix}\right)^{-1}\left(\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\right)$$

$$= \begin{bmatrix} 6 & 5 \\ 5 & 6 \end{bmatrix}^{-1}\begin{bmatrix} 4 \\ 4 \end{bmatrix} = \begin{bmatrix} 6 & -5 \\ -5 & 6 \end{bmatrix} \cdot \frac{1}{11} \cdot \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

$$= \frac{4}{11}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

b)



Predicted output based on training data:

$$\hat{\mathbf{Y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 1 & 1 \end{bmatrix}\frac{4}{11}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{11}\begin{bmatrix} 12 \\ 12 \\ 8 \end{bmatrix} \neq \mathbf{Y}\text{, or}$$

$$\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T = \frac{1}{11}\begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 6 & -5 \\ -5 & 6 \end{bmatrix}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix} = \frac{1}{11}\begin{bmatrix} 10 & -1 & 3 \\ -1 & 10 & 3 \\ 3 & 3 & 2 \end{bmatrix}$$

$$\hat{\mathbf{Y}} = \mathbf{H}\mathbf{Y} = \frac{1}{11}\begin{bmatrix} 12 \\ 12 \\ 8 \end{bmatrix}$$

c) Training error (in MSE):

$$\frac{1}{3}\sum_{i=1}^{3}\left(y_i - \mathbf{x}_i^T\hat{\boldsymbol{\beta}}\right)^2 = \frac{1}{3}\|\mathbf{Y}-\hat{\mathbf{Y}}\|^2 = \frac{1}{3}\|\mathbf{Y}-\mathbf{X}\hat{\boldsymbol{\beta}}\|^2$$

$$= \frac{1}{3}\cdot\frac{1}{11^2}\left\|\begin{bmatrix} 12-11 \\ 12-11 \\ 8-11 \end{bmatrix}\right\|^2 = \frac{1}{3}\cdot\frac{1}{11^2}(1+1+9) = \frac{1}{3}\cdot\frac{1}{11} = 0.03$$

Testing error (in MSE):

$$\mathbf{X}_{\text{test}} = \begin{bmatrix} 1.2 & 0.9 \\ 1.8 & 0.3 \end{bmatrix} \quad \mathbf{Y}_{\text{test}} = \begin{bmatrix} 0.9 \\ 0.8 \end{bmatrix} \quad (\mathbf{X}_{\text{test}}, \mathbf{Y}_{\text{test}}):$$ testing data

$$\frac{1}{2}\sum_{i=4}^{5}\left(y_i - \mathbf{x}_i^T\hat{\boldsymbol{\beta}}\right)^2 = \frac{1}{2}\|\mathbf{Y}_{\text{test}}-\hat{\mathbf{Y}}_{\text{test}}\|^2 = \frac{1}{2}\|\mathbf{Y}_{\text{test}}-\mathbf{X}_{\text{test}}\hat{\boldsymbol{\beta}}\|^2$$

$$= \frac{1}{2}\left\|\begin{bmatrix} 0.9 \\ 0.8 \end{bmatrix} - \begin{bmatrix} 1.2 & 0.9 \\ 1.8 & 0.3 \end{bmatrix}\left(\frac{4}{11}\begin{bmatrix} 1 \\ 1 \end{bmatrix}\right)\right\|^2 = \frac{1}{2}\left\|\begin{bmatrix} 0.14 \\ 0.04 \end{bmatrix}\right\|^2 = 0.01$$

Testing error is usually larger than training error.

Convolutional Neural Network (CNN)

Learning objectives

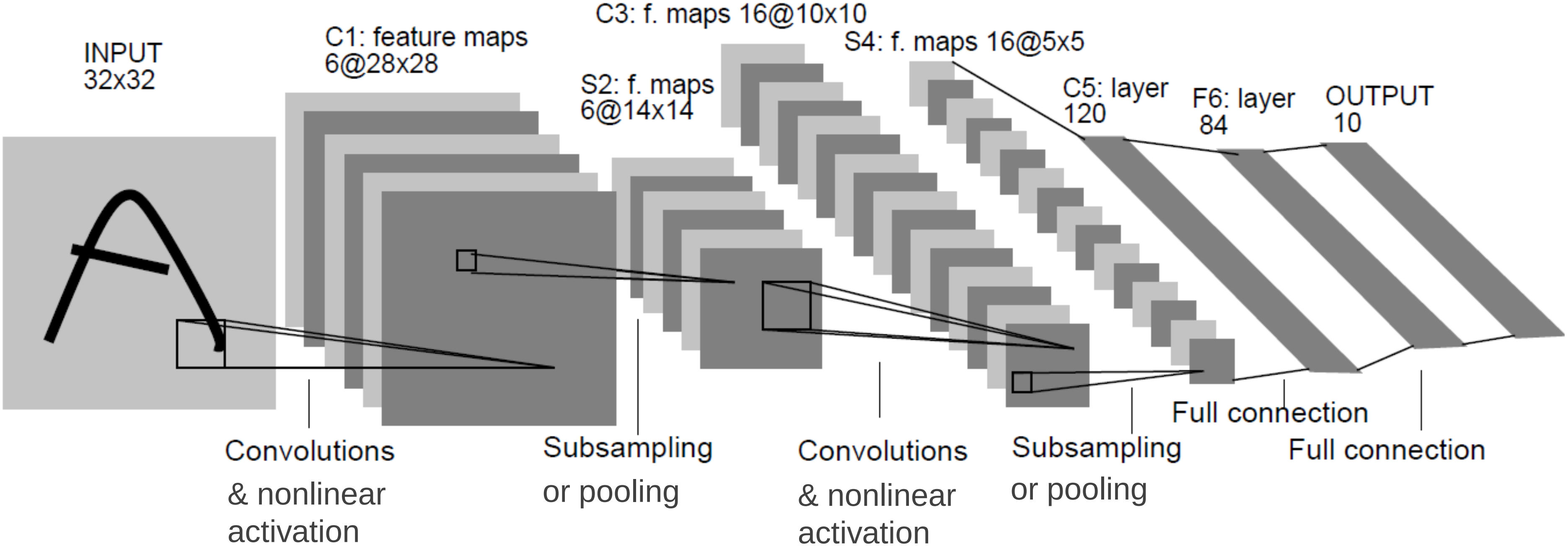
- Describe the structure of CNN
- Build and train simple CNNs using a deep learning package

(Ref: Ch 9 of Goodfellow et al. 2016)

Some slides were adapted from Stanford's CS231n by Fei-Fei Li et al.: http://cs231n.stanford.edu/

Convolutional Neural Network (CNN)

The **single** most important technology that fueled the rapid development of ***deep learning*** and ***big data*** in the past decade.



LeCun, Bottou, Bengio, Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proc. IEEE*, **1998.**

Why is Deep Learning so Successful?

1. **Improved model:** convolutional layer, more layers ("deep"), simpler activation (i.e., ReLU), skip/residual connection (i.e., ResNet), attention (i.e., Transformer)
2. **Big data:** huge dataset, transfer learning
3. **Powerful computation:** graphical processing units (GPUs)

- Example of big data: ImageNet (22K categories, 15M images)

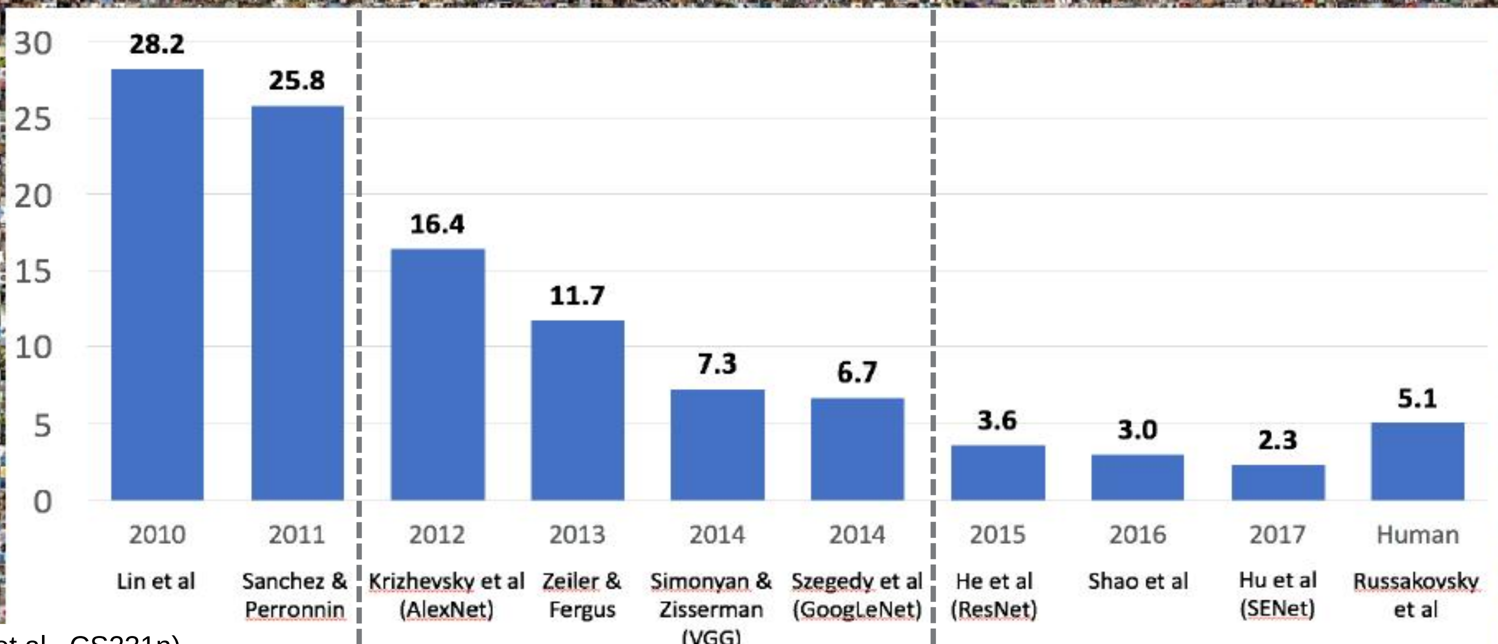
Deng, Dong, Socher, Li, Li & Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," *IEEE CVPR*, 2009.

IMAGENET Large Scale Visual Recognition Challenge

The Image Classification Challenge:

1,000 object classes

1,431,167 images



(Fei-Fei Li et al., CS231n)

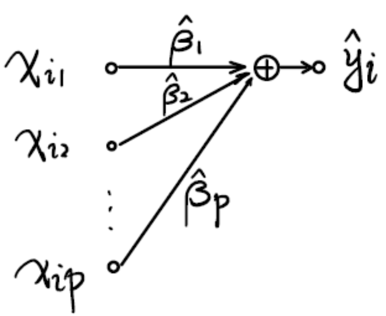
Linear Model to Neural Network

Recall linear model w/ multiple predictors / features / inputs.

$$\underbrace{y_i}_{\text{true output}} = \sum_{j=1}^{p} x_{ij}\beta_j + e_i = \underbrace{[\beta_1, \dots, \beta_p]}_{\text{true weights}} \begin{bmatrix} x_{i1} \\ \vdots \\ x_{ip} \end{bmatrix} + e_i \quad , i = 1, \dots, n.$$

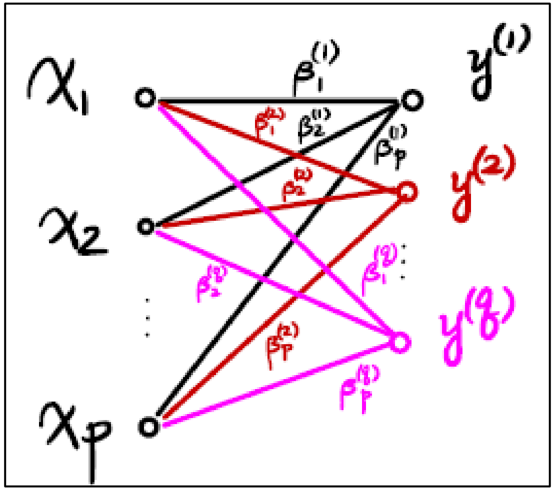
$$\underbrace{\hat{y}_i}_{\text{predicted output}} = \sum_{j=1}^{p} x_{ij}\hat{\beta}_j = \underbrace{[\hat{\beta}_1, \dots, \hat{\beta}_p]}_{\text{estimated weights}} \begin{bmatrix} x_{i1} \\ \vdots \\ x_{ip} \end{bmatrix} \quad , i = n+1, \dots, n+m.$$

Graphically we have:



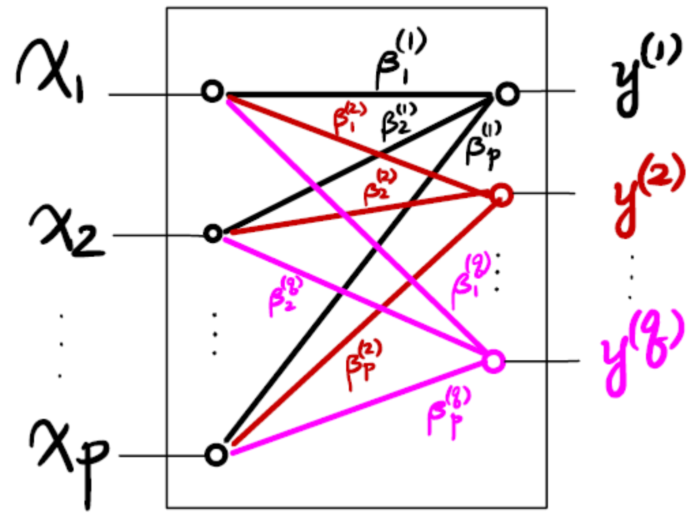
① Use multiple linear models

② Simplify the notations.



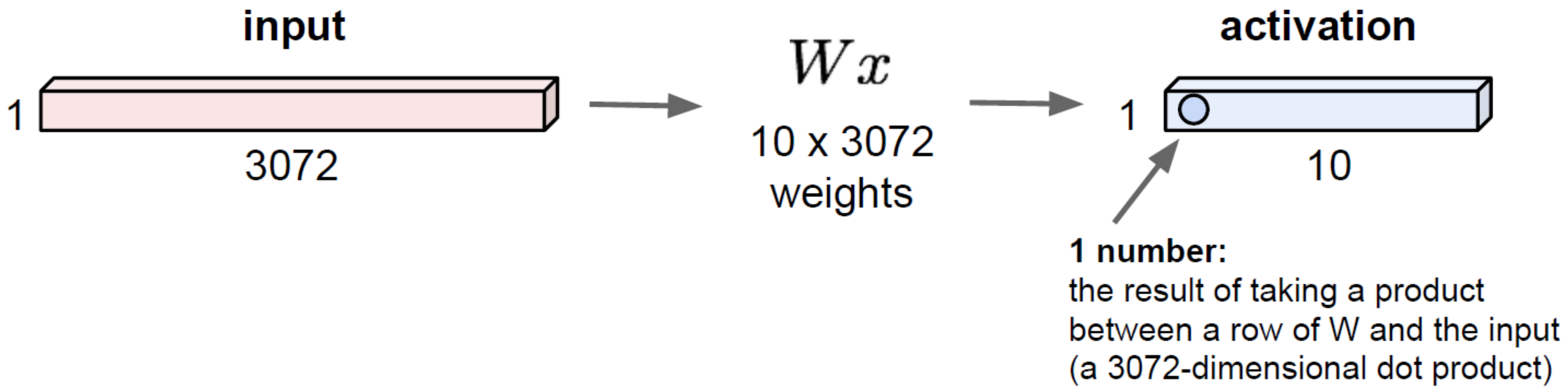
Fully-Connected Layer for 1D Signal

$$\underbrace{\begin{bmatrix} y^{(1)} \\ \hline y^{(2)} \\ \hline \vdots \\ \hline y^{(q)} \end{bmatrix}}_{\substack{\text{layer output,} \\ \underset{\sim}{y} \in \mathbb{R}^{q}}} = \underbrace{\begin{bmatrix} \beta_1^{(1)} & \beta_2^{(1)} & \cdots & \beta_p^{(1)} \\ \hline \beta_1^{(2)} & \beta_2^{(2)} & \cdots & \beta_p^{(2)} \\ \hline \vdots & & & \vdots \\ \hline \beta_1^{(q)} & \beta_2^{(q)} & \cdots & \beta_p^{(q)} \end{bmatrix}}_{\substack{\text{dense weight matrix} \\ B \in \mathbb{R}^{q \times p}}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix}}_{\substack{\text{layer input,} \\ \underset{\sim}{x} \in \mathbb{R}^{p}}}$$



Fully-Connected Layer for RGB Image

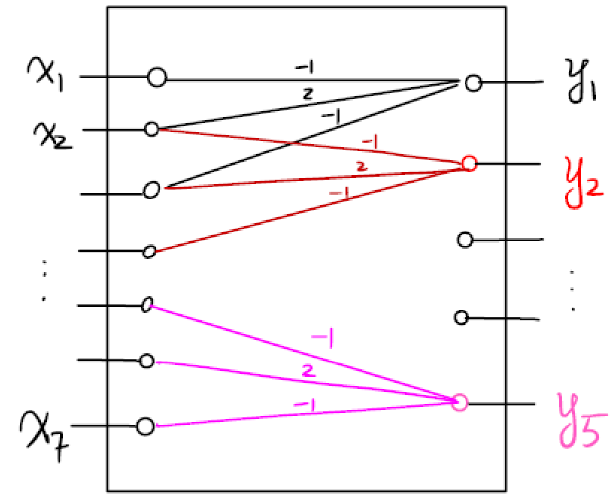
32x32x3 image -> stretch to 3072 x 1



Convolutional Layer for 1D Signal

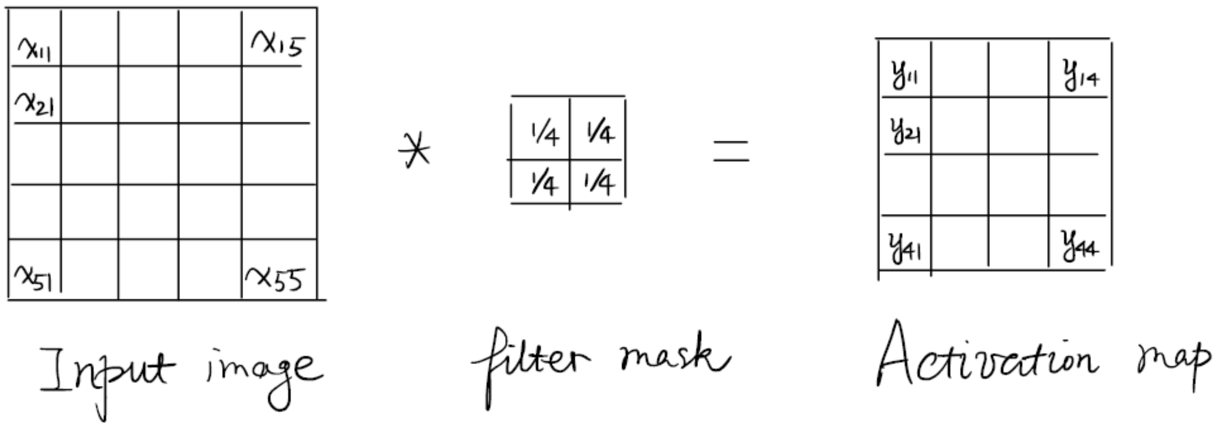
$$\begin{bmatrix} y_1 \\ \vdots \\ y_5 \end{bmatrix} = \begin{bmatrix} -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ \vdots & & -1 & 2 & -1 & & \vdots \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_7 \end{bmatrix}$$

Sparse weight matrix

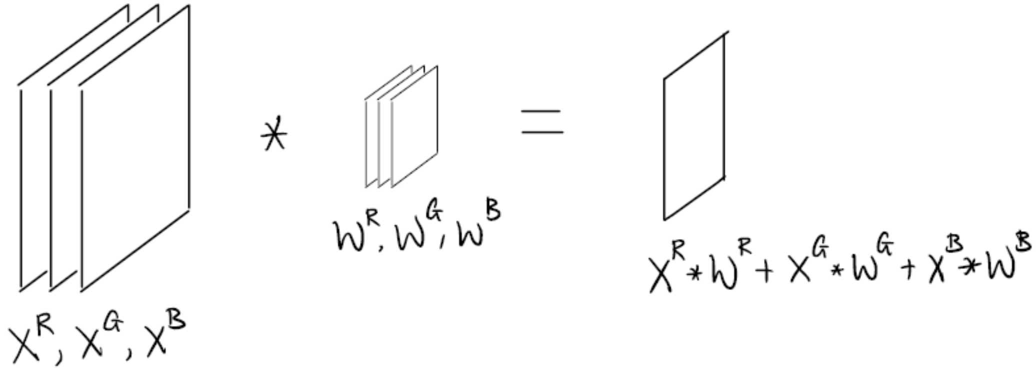


Input	x_1 | x_2 | x_3 | x_4 | x_5 | x_6 | x_7	length 7
Convolution/filter mask	$*$ -1 | 2 | -1 $\longrightarrow$	length 3
Output (w/o boundary elements)	y_1 | y_2 | y_3 | y_4 | y_5	length $7-(3-1)=5$

Convolutional Layer for 2D Matrix/Image



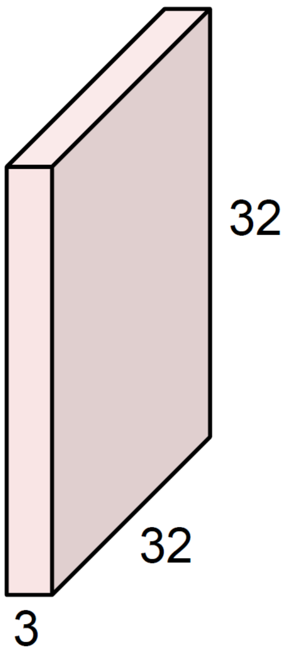
2D Convolution



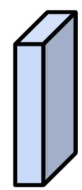
Multiple color channels need multiple filter masks

Convolutional Layer for RGB Image

32x32x3 image

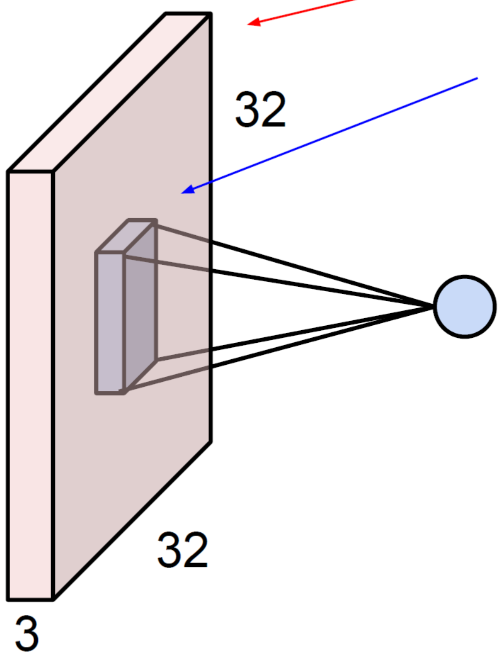


5x5x3 filter



Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

(Fei-Fei Li et al., CS231n)



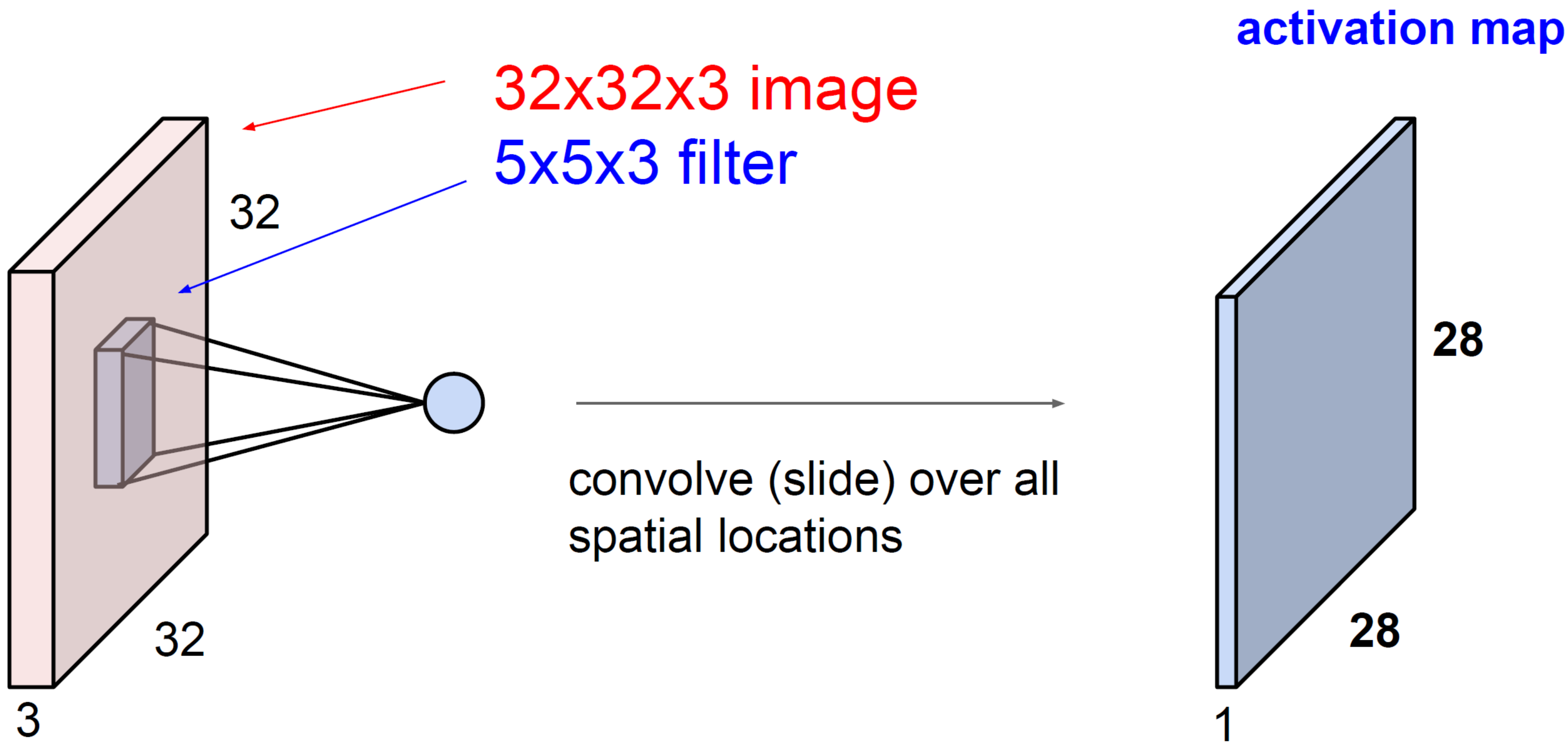
32x32x3 image

5x5x3 filter w

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

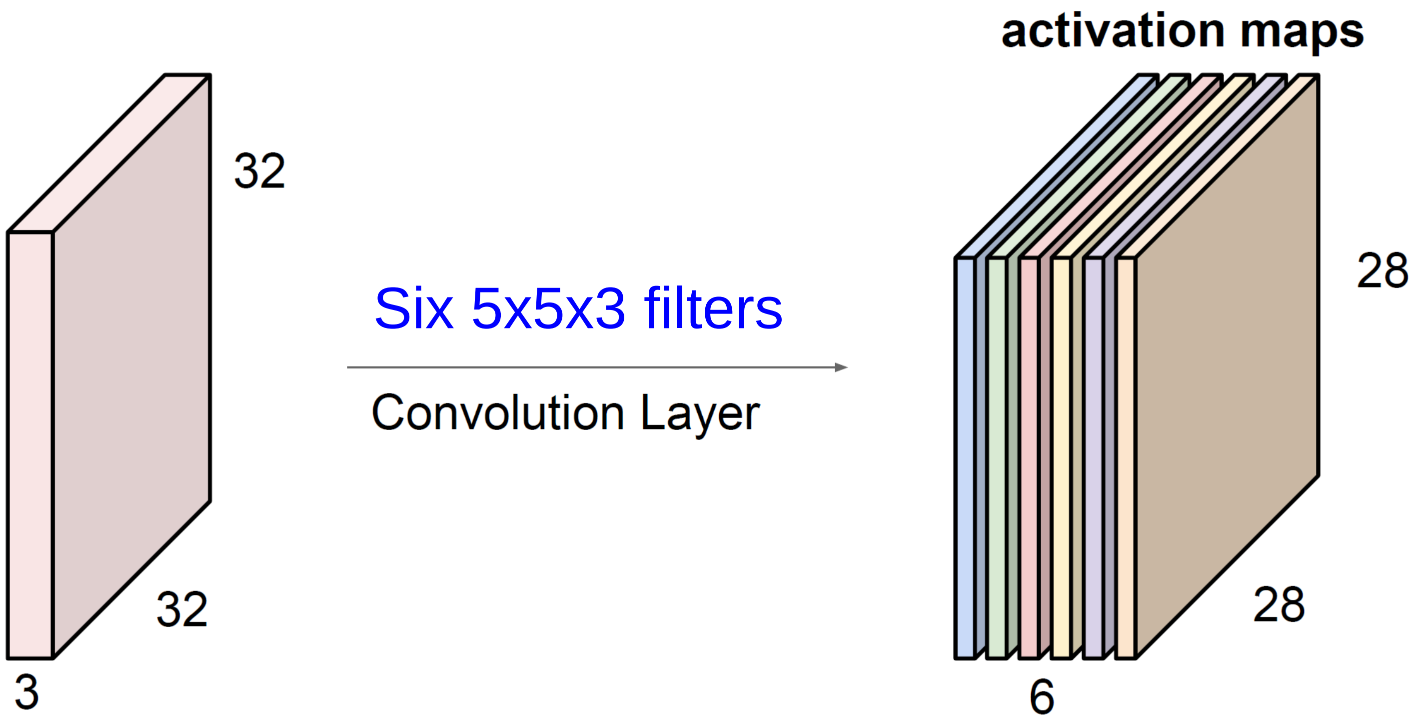
$$w^T x + b$$

A closer look at spatial dimensions:



(Fei-Fei Li et al., CS231n)

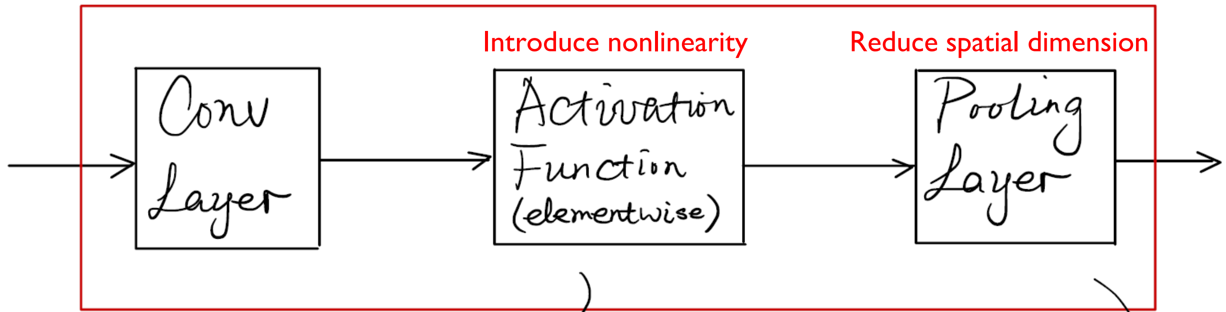
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

(Fei-Fei Li et al., CS231n)

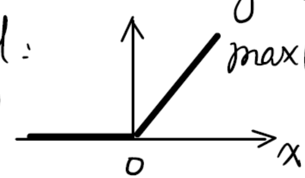
Building Block for Modern CNN



(Rectified Linear Unit)

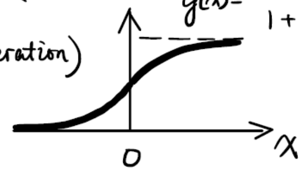
ReLU: (modern)

$g(x) = \max(0, x)$



Logistic: (prior generation)

$g(x) = \frac{1}{1+e^{-x}}$



Ex: $\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$: $\{x_{ij}\}_{i,j=1}^{2}$

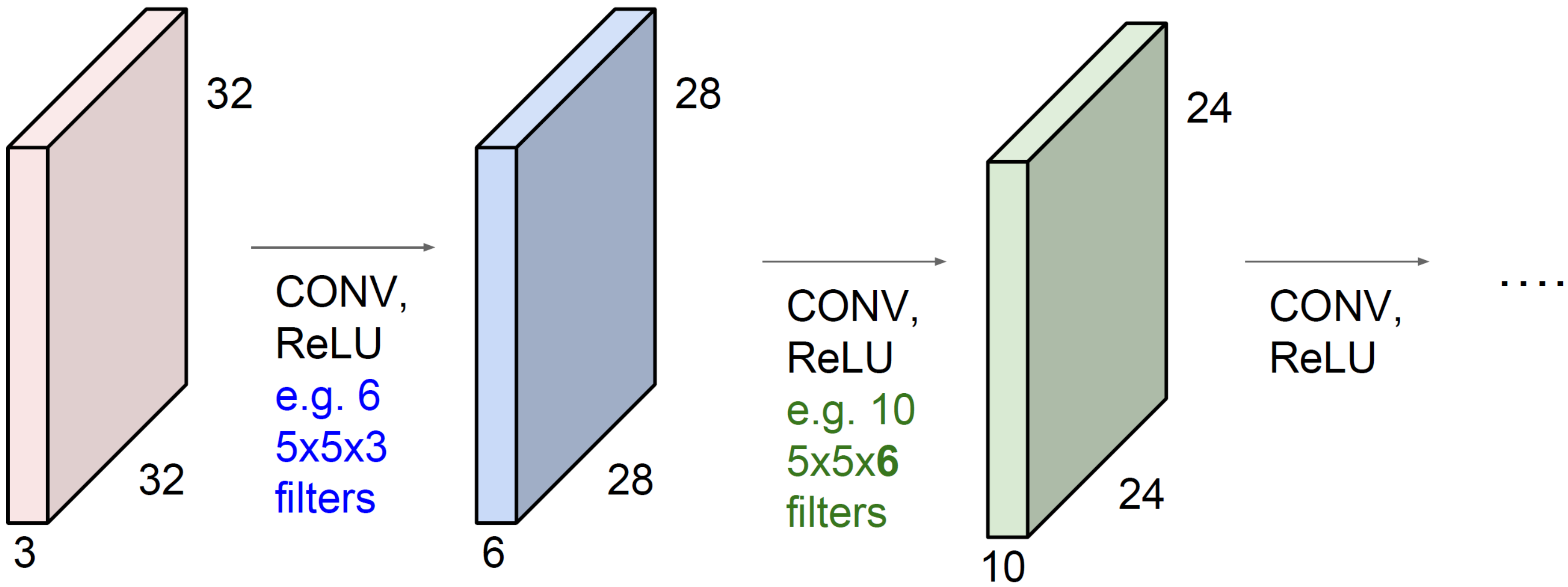
Max pooling:

$$g(\{x_{ij}\}) = \max(\{x_{ij}\})$$

Average pooling:

$$g(\{x_{ij}\}) = \frac{1}{|\{x_{ij}\}|} \sum_{i,j} x_{ij}$$

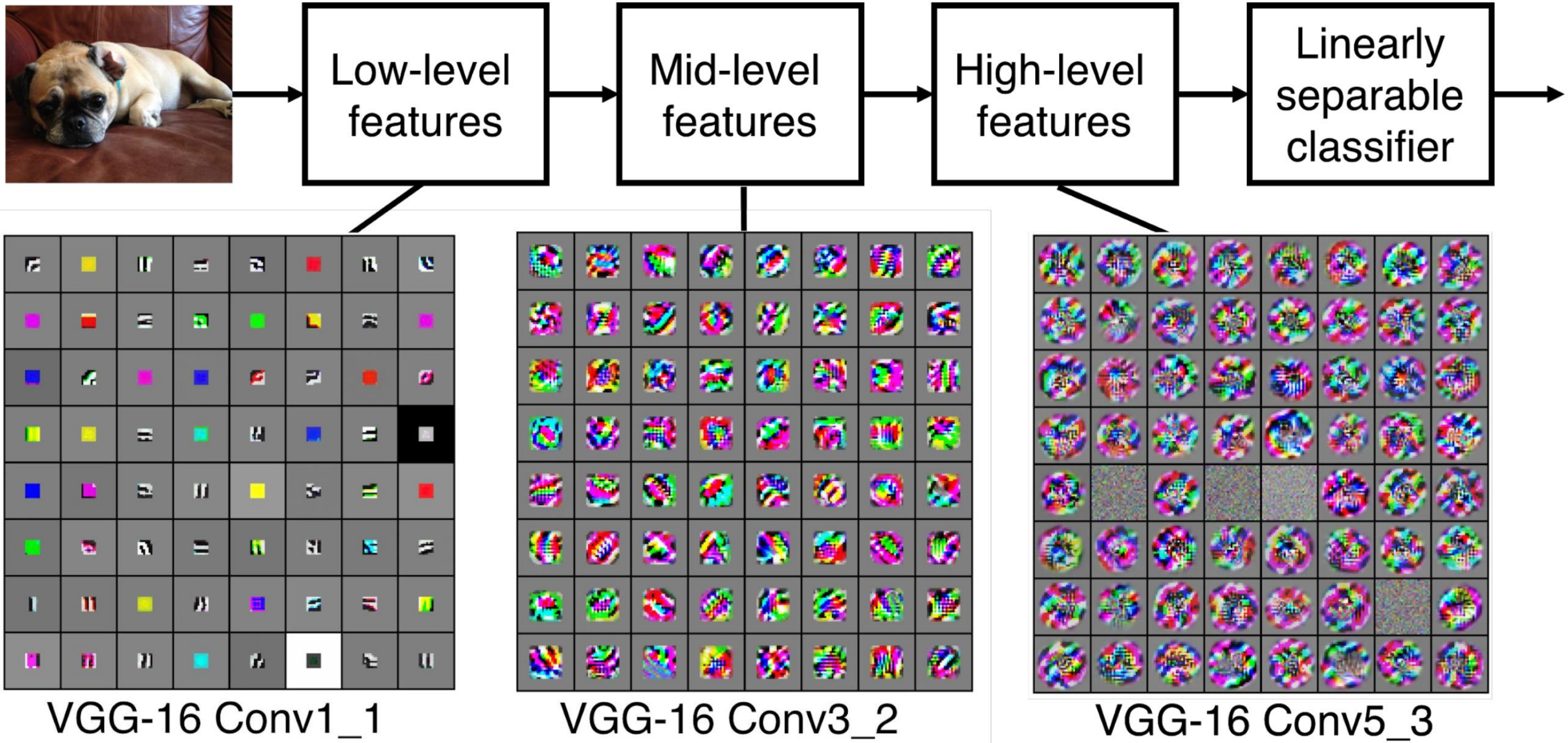
CNN is composed of a sequence of convolutional layers, interspersed with activation functions (ReLU, in most cases).



(Fei-Fei Li et al., CS231n)

[Zeiler and Fergus 2013]

Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].



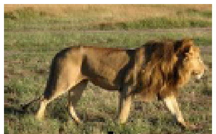
(Fei-Fei Li et al., CS231n)

IMAGENET Large Scale Visual Recognition Challenge

AlexNet / ResNet

Year 2010	Year 2012 (AlexNet)	Year 2014		Year 2015 (ResNet)
NEC-UIUC	SuperVision	GoogLeNet	VGG	MSRA

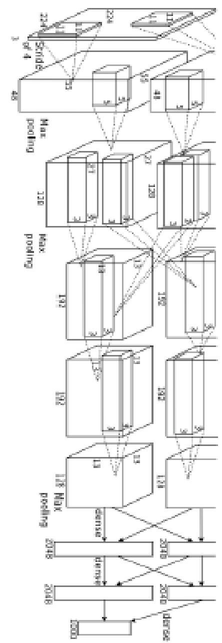
Year 2010 — NEC-UIUC



Dense descriptor grid: HOG, LBP
↓
Coding: local coordinate, super-vector
↓
Pooling, SPM
↓
Linear SVM

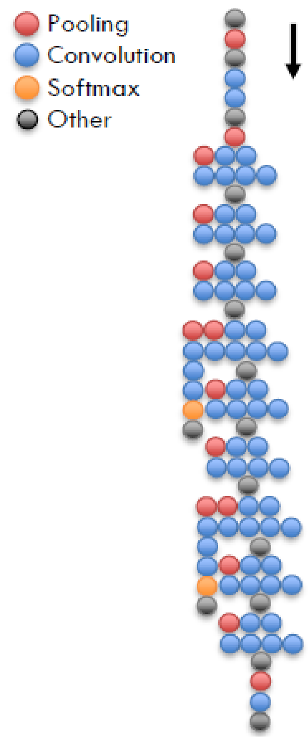
[Lin CVPR 2011]

Year 2012 — SuperVision (AlexNet)



[Krizhevsky NIPS 2012]

Year 2014 — GoogLeNet



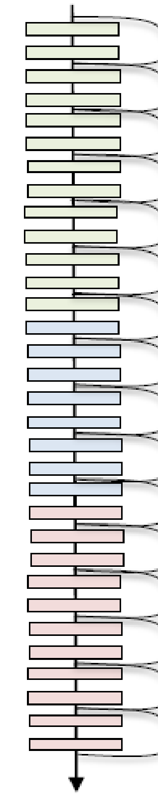
[Szegedy arxiv 2014]

Year 2014 — VGG

Image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
fc-4096
fc-4096
fc-1000
softmax

[Simonyan arxiv 2014]

Year 2015 — MSRA (ResNet)



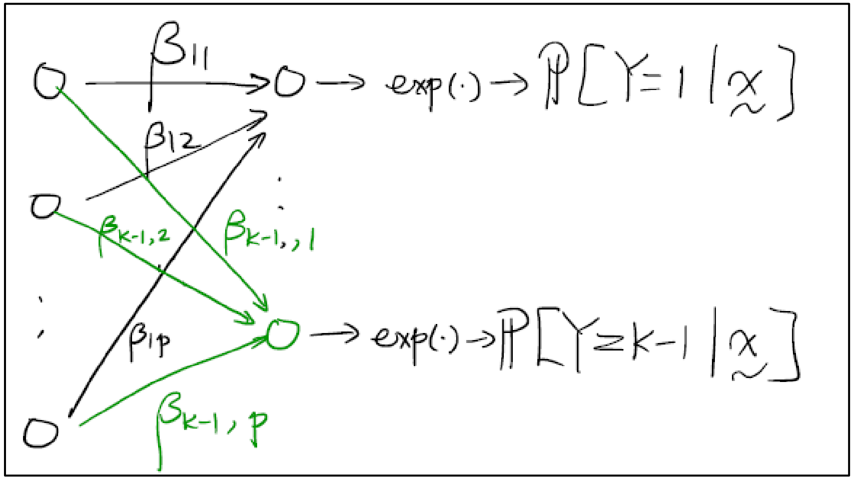
[He ICCV 2015]

(Fei-Fei Li et al., CS231n)

One Last Thing: When Output is Categorical

- A **softmax layer** is needed:
- Softmax function:

$$\sigma_i(\underset{\sim}{z}) = \frac{e^{\beta z_i}}{\sum_{j=1}^{K} e^{\beta z_j}}$$



- Ex:

$$K=2 . \quad \sigma_1 = \frac{e^{\beta z_1}}{e^{\beta z_1} + e^{\beta z_2}} = \frac{1}{1 + e^{\beta(z_2 - z_1)}}$$

When β very large,

$z_2 > z_1$ leads to $\begin{cases} \sigma_1 = 0 \\ \sigma_2 = 1 \end{cases}$

Winner takes all!

Machine Learning (ML) and Data Science (DS)

- Follow-up machine learning / data science courses:
  - ECE 411 Intro to Machine Learning
  - ECE 542 Neural Nets and Intro to Deep Learning
  - ECE 592-61 Data Science
  - ECE 759 Pattern Recognition and Machine Learning
  - ECE 763 Computer Vision
  - ECE 792-41 Statistical Foundations for Signal Processing & Machine Learning
  - Any courses/videos on YouTube, Coursera, etc.
- Data science competitions: kaggle.com
- Programming languages for ML/DS: Python, R, Matlab