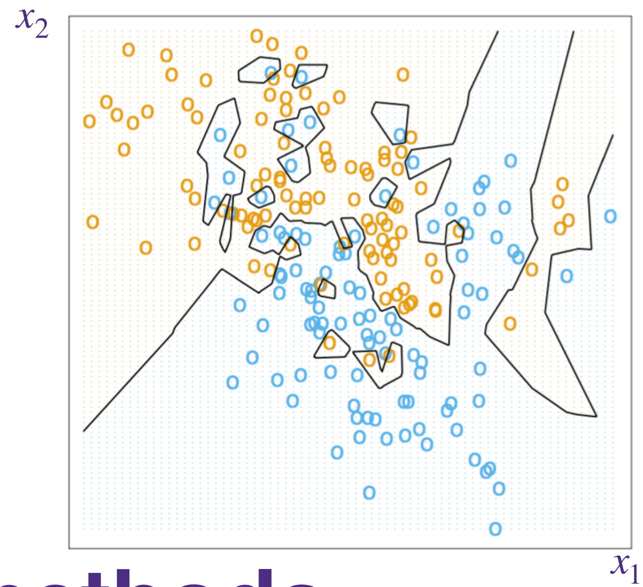
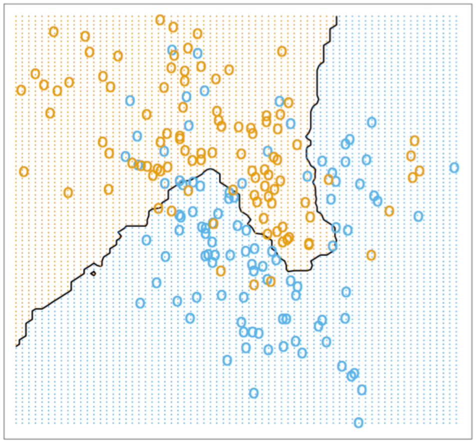


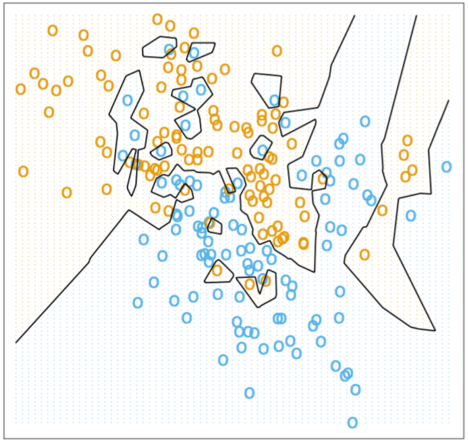
Nearest neighbor methods

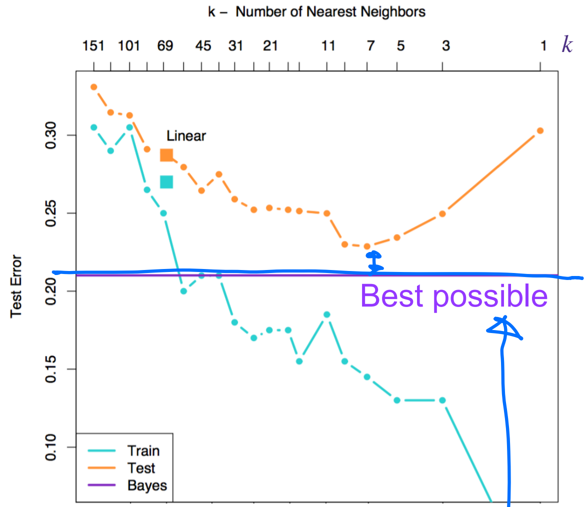
Recap of nearest neighbor methods

$k = 15$



$k = 1$





- **Principle** of designing nearest neighbor methods
  - Consider a “good” estimator that cannot be implemented (because it requires the knowledge of $P_{X,Y}(x, y)$)

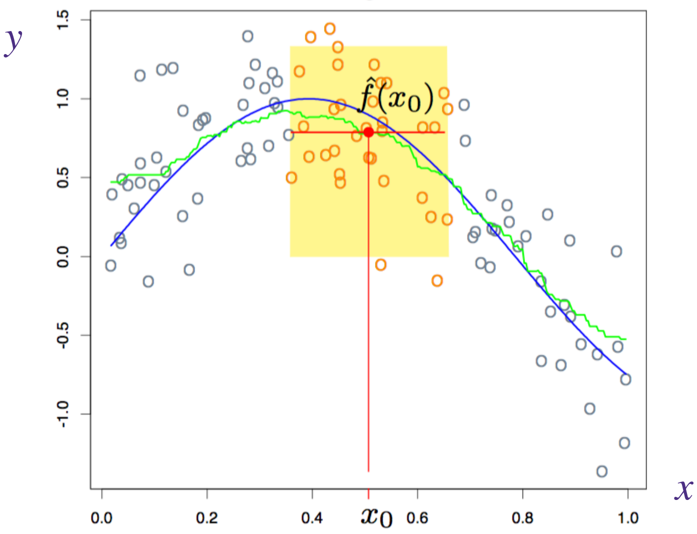
    e.g., for binary classification it is $\hat{y} = +1$ if $P(x, +1) > P(x, -1)$

    -1 if $P(x, +1) < P(x, -1)$
  - Replace $P_{X,Y}(x, y)$ by k_x^y **(i.e.# of nearest neighbors of label y)** among k-NNs

    e.g., $\hat{y} = +1$ if $k_x^+ > k_x^-$

    -1 if $k_x^+ < k_x^-$

Consider regression



- **Principle** of designing nearest neighbor methods
  - Consider a "good" estimator that cannot be implemented

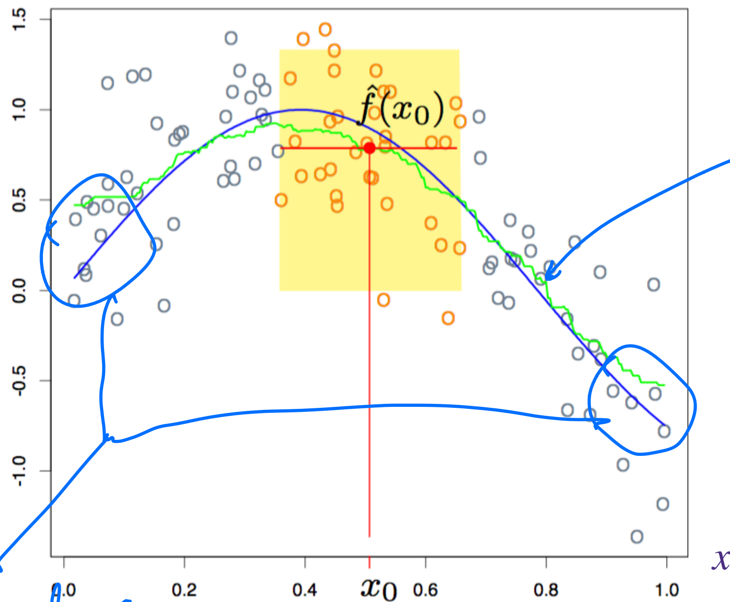
    e.g., for regression optimal predictor is $\hat{y} = \mathbb{E}[y \,|\, x] = \dfrac{\int y\, P_{X,Y}(x,y)\, dy}{\int 1\, P_{X,Y}(x,y)\, dy}$ } = $P(x)$

  - Replace $P_{X,Y}(x, y)$ by the empirical distribution among k-nearest neighbors

    e.g., $$\hat{y} = \frac{\sum_{j \in \text{nearest neighbor}} y_j}{\sum_{j \in \text{nearest neighbor}} 1} = \frac{\sum_{j \in \text{nearest neighbor}} y_j}{k}$$

Nearest neighbor regression

y $\{(x_i, y_i)\}_{i=1}^n$



discontinuous

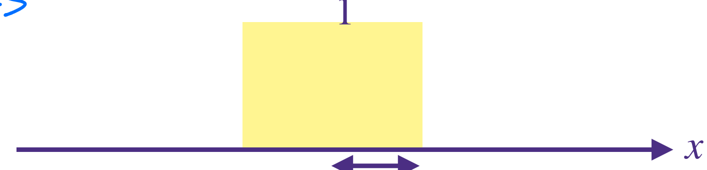
Boundary Bias

- k-nearest neighbor regressor is

$$\hat{f}(x) = \frac{1}{k} \sum_{j \in \text{nearest neighbor}} y_j$$

$$= \frac{\sum_{i=1}^n y_i \times \text{Ind}(x_i \text{ is a } k \text{ nearest neighbor})}{\sum_{i=1}^n \text{Ind}(x_i \text{ is a } k \text{ nearest neighbor})}$$

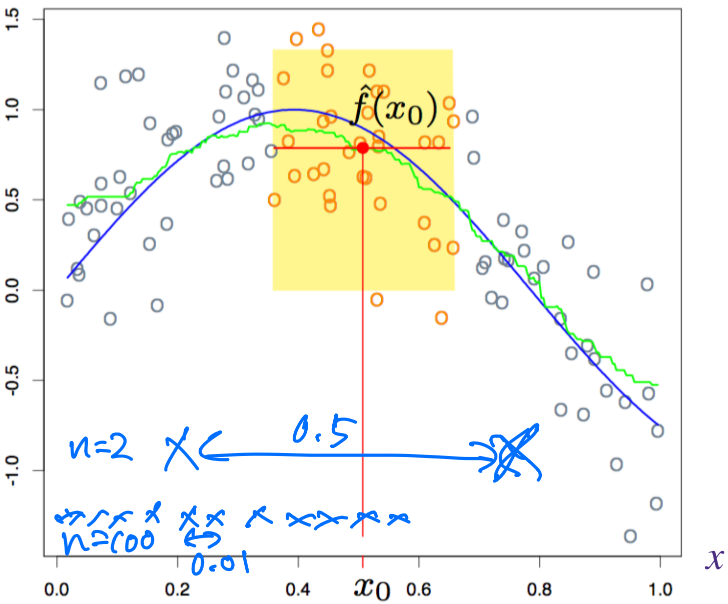
$\text{Ind}(x_i$ is a k nearest neighbor)



ρ_k = distance to the k-th NN

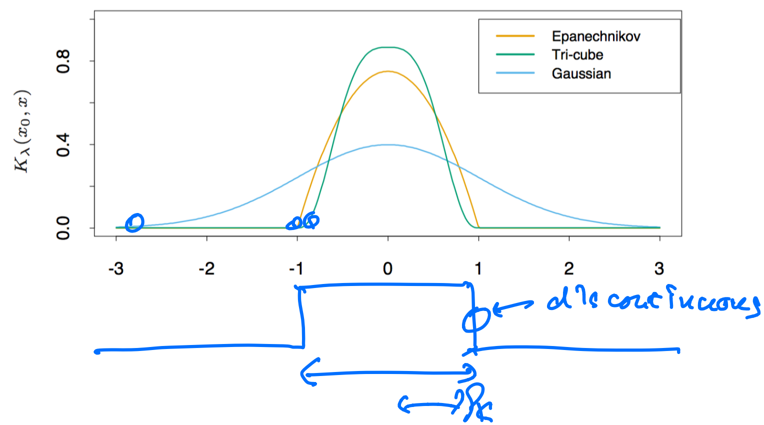
Nearest neighbor regression

$$\{(x_i, y_i)\}_{i=1}^n$$



Why are far-away neighbors weighted same as close neighbors!

smoothing: $K(x, y)$



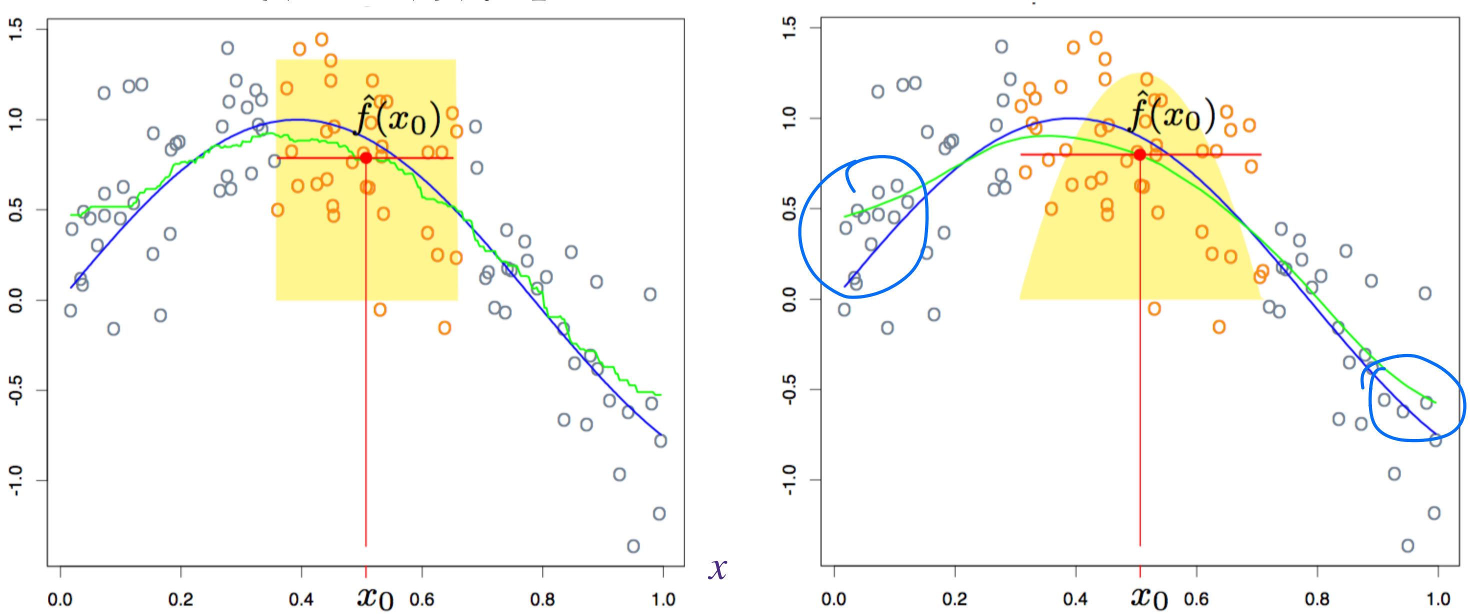
- k-nearest neighbor regressor is

$$\hat{f}(x_0) = \frac{\sum_{i=1}^n y_i \times \text{Ind}(x_i \text{ is a } k \text{ nearest neighbor})}{\sum_{i=1}^n \text{Ind}(x_i \text{ is a } k \text{ nearest neighbor})}$$

$$\widehat{f}(x_0) = \frac{\sum_{i=1}^n K(x_0, x_i) y_i}{\sum_{i=1}^n K(x_0, x_i)}$$

Nearest neighbor regression

y $\{(x_i, y_i)\}_{i=1}^n$



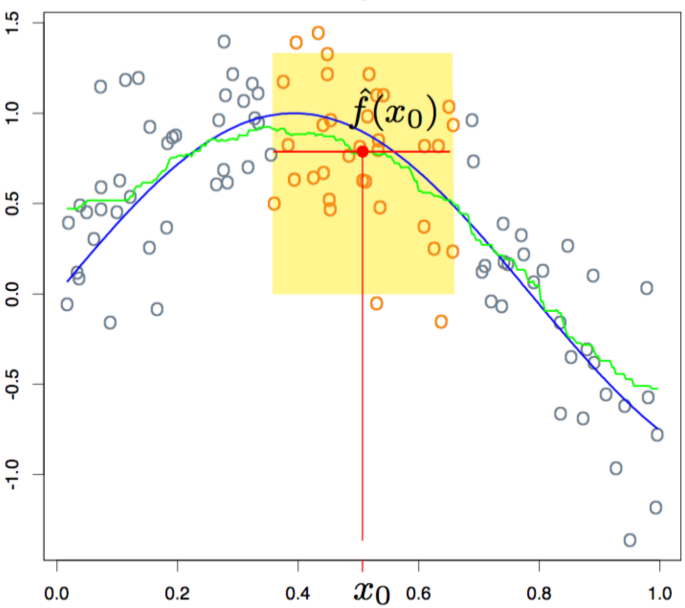
- k-nearest neighbor regressor is

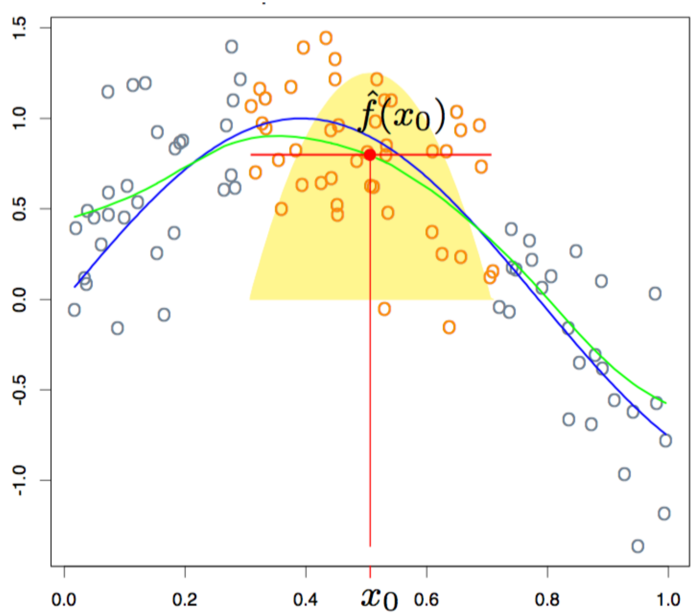
$$\hat{f}(x_0) = \frac{\sum_{i=1}^n y_i \times \text{Ind}(x_i \text{ is a } k \text{ nearest neighbor})}{\sum_{i=1}^n \text{Ind}(x_i \text{ is a } k \text{ nearest neighbor})}$$

$$\widehat{f}(x_0) = \frac{\sum_{i=1}^n K(x_0, x_i) y_i}{\sum_{i=1}^n K(x_0, x_i)}$$

Nearest neighbor regression

$$\{(x_i, y_i)\}_{i=1}^n$$





- k-nearest neighbor regressor is

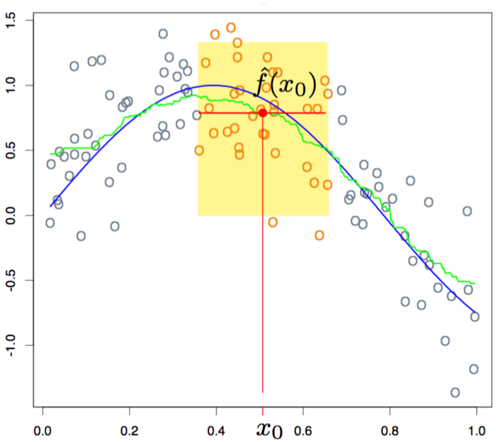
$$\hat{f}(x_0) = \frac{1}{k} \sum_{j \in \text{nearest neighbor}} y_j$$

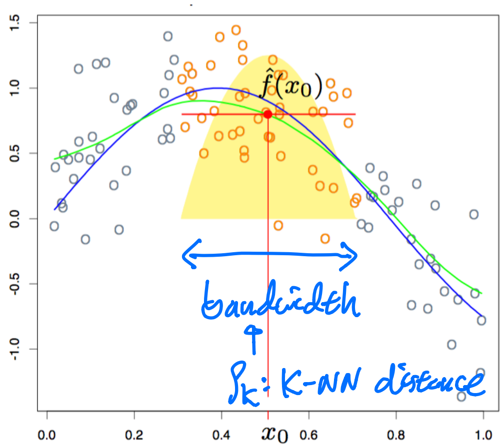
Why just average them?

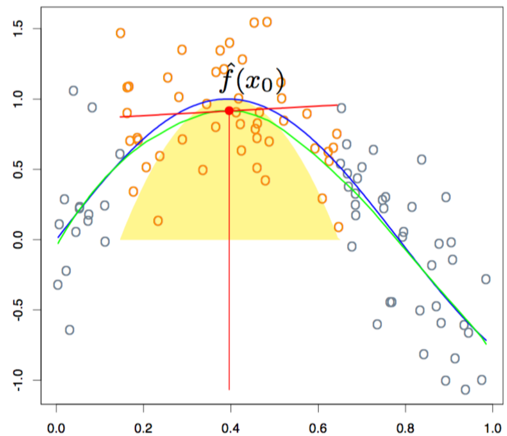
$$\widehat{f}(x_0) = \frac{\sum_{i=1}^n K(x_0, x_i) y_i}{\sum_{i=1}^n K(x_0, x_i)}$$

Nearest neighbor regression

$$\{(x_i, y_i)\}_{i=1}^n$$







$$\hat{f}(x_0) = \frac{\sum_{i=1}^n y_i \times \text{Ind}(x_i \text{ is a } k \text{ nearest neighbor})}{\sum_{i=1}^n \text{Ind}(x_i \text{ is a } k \text{ nearest neighbor})}$$

$$\hat{f}(x_0) = \frac{\sum_{i=1}^n K(x_0, x_i) y_i}{\sum_{i=1}^n K(x_0, x_i)}$$

$$\widehat{f}(x_0) = b(x_0) + w(x_0)^T x_0$$

$$w(x_0), b(x_0) = \arg\min_{w,b} \sum_{i=1}^n K(x_0, x_i)(y_i - (b + w^T x_i))^2$$

Local Linear Regression

Nearest Neighbor Overview

- Very simple to explain and implement
- No training! But finding nearest neighbors in large dataset at test can be computationally demanding (KD-trees help)
- You can use other forms of distance (not just Euclidean)
- Smoothing and local linear regression can improve performance (at the cost of higher variance)
- With a lot of data, "local methods" have strong, simple theoretical guarantees.
- Without a lot of data, neighborhoods aren't "local" and methods suffer (curse of dimensionality).

$|\theta| = d$ is fixed — parametric ⟷ non-parametric — $|\theta| \approx n$ not fixed

linear regression
NN
⋮

nearest-neighbor method
⋮ kernel method.

Questions?

Supervised Learning

$\{(X_i, Y_i)\}$

- Regression
  - $Y_i \in \{1, 2, 3 \ldots\}$
- classification
  - $Y_i \in \mathbb{R}$

Prediction

Unsupervised Learning

$\{X_i\}_{i=1}^{n}$

- Pattern
- similarities
- cluster
- dimensionality reduction
- representation

Principal Component Analysis

Motivation: dimensionality reduction

- it takes $n \times d$ memory to store data $\{x_i\}_{i=1}^n$ with $x_i \in \mathbb{R}^d$
- but many real data have repeated patterns
- can we represent each image compactly, but still preserve most of information, by exploiting similarities?

d pixels per image

n images

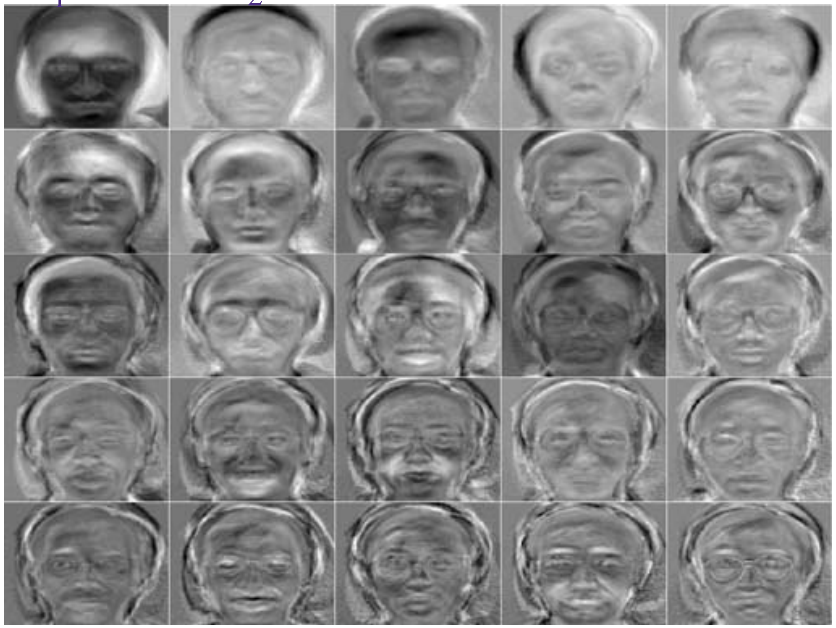
$d \times n$ real values to store the data

Principal component analysis finds a compact linear representation

- patterns that capture the distinct features of the samples is called **principal component** (to be formally defined later)
- we use $r = 25$ principal components

Principal components:

$u_1 \in \mathbb{R}^d$ u_2



Principal component analysis finds a compact linear representation

- patterns that capture the distinct features of the samples is called **principal component** (to be formally defined later)
- we use $r = 25$ principal components
- we can represent each sample as a **weighted linear combination** of the principal components, and just store the weights (as opposed to all pixel values)

Principal components:

$u_1 \in \mathbb{R}^d$ u_2

$$\approx \; a[1]u_1 + a[2]u_2 + \cdots + a[25]u_{25}$$

- Each image is now represented by $r = 25$ numbers $a = (a[1], \ldots, a[25])$
- To store n images, it requires memory of only $d \times r + r \times n \;\ll\; d \times n$

$u_1 \cdots u_r$ $\{a_i\}_{i=1}^{n}$

10 principal components give a pretty good reconstruction of a face

average face $\bar{x} + a[1]u_1$ $\bar{x} + a[1]u_1 + a[2]u_2$

$\bar{x}$ r = 1 r = 2 r = 3 r = 4

r = 10

Ground truths real face

Assumption

- Notice how we started with the average face $\bar{x} = \dfrac{1}{n}\sum_{i=1}^{n} x_i$
- PCA is applied to $\{x_i - \bar{x}\}_{i=1}^{n}$
- For simplicity, we will assume that x_i's are centered such that $\dfrac{1}{n}\sum_{i=1}^{n} x_i = 0$
- otherwise, without loss of generality,
  everything we do can be applied to the re-centered version of the data,
  i.e. $\{x_i - \bar{x}\}_{i=1}^{n}$, with $\bar{x} = \dfrac{1}{n}\sum_{i=1}^{n} x_i$

How do we define the principal components?

- Dimensionality reduction (for some $r \ll d$):
  we would like to have a set of orthogonal directions $u_1, \ldots, u_r \in \mathbb{R}^d$, with $\|u_j\|_2 = 1$ for all j, such that each data can be represented as linear combination of those direction vectors, i.e.

$$x_i \;\approx\; p_i \;=\; a_i[1]u_1 + \cdots + a_i[r]u_r$$

x_i

p_i

$$x_i = \begin{bmatrix} x_i[1] \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_i[d] \end{bmatrix} \longrightarrow a_i = \begin{bmatrix} a_i[1] \\ \vdots \\ a_i[r] \end{bmatrix}$$

How do we find the principal components?

- Dimensionality reduction (for some $r \ll d$):
  we would like to have a set of orthogonal directions $u_1, \ldots, u_r \in \mathbb{R}^d$, with $\|u_j\|_2 = 1$ for all j, such that each data can be represented as linear combination of those direction vectors, i.e.

$$x_i \approx p_i = a_i[1]u_1 + \cdots + a_i[r]u_r$$

$$x_i = \begin{bmatrix} x_i[1] \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_i[d] \end{bmatrix} \longrightarrow a_i = \begin{bmatrix} a_i[1] \\ \vdots \\ a_i[r] \end{bmatrix}$$

- those directions that minimize the average reconstruction error for a dataset is called the **principal components**

$\frac{1}{n}\sum_{i=1}^{n} \|x_i - p_i\|_2^2$

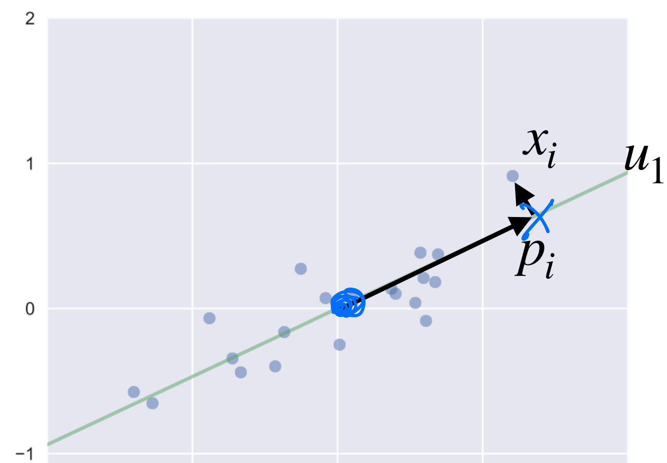
- given a choice of $u_1, \ldots, u_r$, the best representation p_i of x_i is the projection of the point onto the subspace spanned by u_j's, i.e.

$$a_i[j] = u_j^T x_i$$

length.

$$p_i = \sum_{j=1}^{r} \underbrace{(u_j^T x_i)}_{a_i[j]} u_j$$

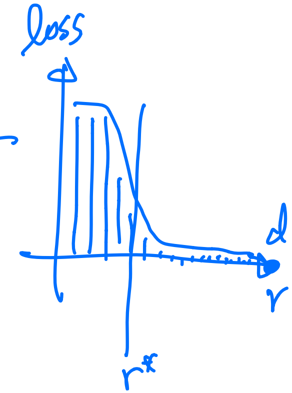
direction



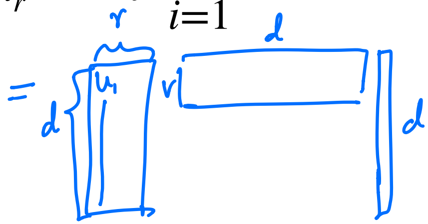
- we will use these without proving it

Principal components is the subspace that minimizes the reconstruction error

$$\underset{u_1,\ldots,u_r}{\text{minimize}} \ \frac{1}{n}\sum_{i=1}^{n} \|x_i - p_i\|_2^2$$



$$p_i = \sum_{j=1}^{r} (u_j^T x_i) u_j = \mathbf{U}\mathbf{U}^T x_i$$



where $\mathbf{U} = [u_1 \ \ u_2 \ \ \cdots \ \ u_r] \in \mathbb{R}^{d\times r}$

$$\underset{U}{\text{minimize}} \ \frac{1}{n}\sum_{i=1}^{n} \|x_i - \mathbf{U}\mathbf{U}^T x_i\|_2^2$$

$$\text{subject to} \ \mathbf{U}^T\mathbf{U} = \mathbf{I}_{r\times r}$$

normalization : $u_j^T u_j = 1$

orthogonality : $u_j^T u_l = 0$

Q. How do we solve this optimization?

Minimizing reconstruction error to find principal components

$$\underset{U}{\text{minimize}} \quad \frac{1}{n}\sum_{i=1}^{n} \|x_i - \mathbf{U}\mathbf{U}^T x_i\|_2^2$$

$$\text{subject to} \quad \mathbf{U}^T\mathbf{U} = \mathbf{I}_{r\times r}$$