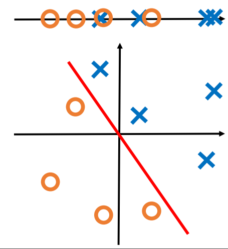
Dimensionality reduction

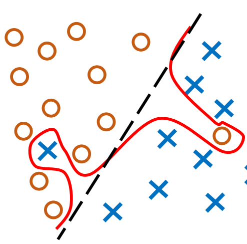
CISC 5800
Professor Daniel Leeds

The benefits of extra dimensions



- **Finds existing complex separations between classes**

The risks of too-many dimensions



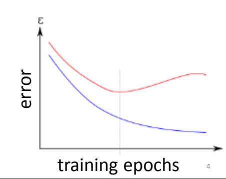
- **High dimensions with kernels over-fit the outlier data**
- Two dimensions ignore the outlier data

Training vs. testing

- **Training**: learn parameters from set of data in each class
- **Testing**: measure how often classifier correctly identifies new data

- More training reduces classifier error ε
  - More gradient ascent steps
  - More learned feature

- Too much training causes worse testing error – overfitting



Goal: High Performance, Few Parameters

- "Information criterion": performance/parameter trade-off

- Variables to consider:
  - **L** likelihood of train data after learning
  - **k** number of parameters (e.g., number of features)
  - **m** number of points of training data

- Popular information criteria:
  - Akaike information criterion **AIC**: log(L) - k
  - Bayesian information criterion **BIC**: log(L) - 0.5 k log(m)

Decreasing parameters

- Force parameter values to 0
  - L1 regularization
  - Support Vector selection
  - Feature selection/removal

- Consolidate feature space
  - Component analysis

Feature removal

- Start with feature set: F={x_1, …, x_k}
- Find classifier performance with set F: perform(F)
- Loop
  - Find classifier performance for removing feature x_1, x_2, …, x_k: argmax_i perform(F-x_1)
  - Remove feature that causes least decrease in performance: F=F-x_i

Repeat, using AIC or BIC as termination criterion

AIC: log(L) - k

BIC: log(L) - 0.5 k log(m)

AIC testing: log(L)-k

Features	k (num features)	L (likelihood)	AIC
F	40	0.1	-42.3
F-{x_3}	39	0.03	-41.5
F-{x_3,x_{24}}	38	0.005	-41.3
F-{x_3,x_{24},x_{32}}	37	0.001	-40.9
F-{x_3,x_{24},x_{32},x_{15}}	36	0.0001	-41.2

Feature selection

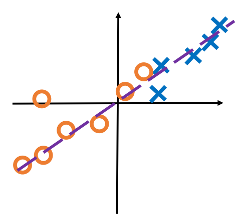
- Find classifier performance for just set of 1 feature: argmax_i perform({x_i})
- Add feature with highest performance: F={x_i}
- Loop
  - Find classifier performance for adding one new feature: argmax_i perform(F+{x_i})
  - Add to F feature with highest performance increase: F=F+{x_i}

Repeat, using AIC or BIC as termination criterion

AIC: log(L) - k

BIC: log(L) - 0.5 k log(m)

Defining new feature axes

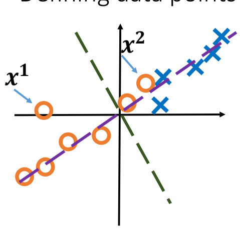


- Identify a common trend

$$\boldsymbol{u}_1 = \begin{bmatrix} 0.91 \\ 0.45 \end{bmatrix}$$

- Map data onto new dimension $\boldsymbol{u}_1$

Defining data points with new axes



$$x^1 = -1 \times \boldsymbol{u}_1 + (-0.5) \times \boldsymbol{u}_2$$

$$x^2 = \mathbf{0.5} \times \boldsymbol{u}_1 + (-0.1) \times \boldsymbol{u}_2$$

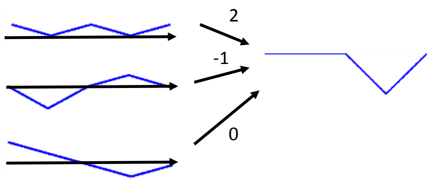
Component analysis

Each data point $\boldsymbol{x}^i$ in D can be reconstructed as sum of components $\boldsymbol{u}$:

- $\boldsymbol{x}^i = \sum_{q=1}^{T} z_q^i \boldsymbol{u}_q$
- z_q^i is weight on q^{th} component to reconstruct data point $\mathbf{x}^i$

Component analysis: examples

Components   Data



Component analysis: examples

"Eigenfaces" – learned from set of face images

u: nine components

x^i: data reconstructed

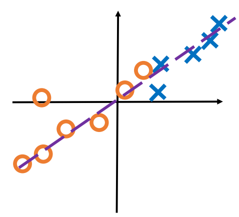
Types of component analysis

$$\boldsymbol{x^i} = \sum_{q=1}^{T} z_q^i \boldsymbol{u_q}$$

Learn new axes from data sets: common "components"

- Principal component analysis (PCA):
  - Best reconstruction of each data point $\boldsymbol{x}^i$ with first t components
  - Each component perpendicular to all others: $(\boldsymbol{u_i})^T \boldsymbol{u_j} = 0 \quad \forall i \neq j$
- Independent component analysis (ICA):
  - Minimize number of components to describe each x^i
  - Can focus on different components for different x^i
- Non-negative matrix factorization (NMF):
  - All data $\boldsymbol{x}^i$ non-negative
  - All components and weights non-negative $\boldsymbol{u}_j \geq 0,\ z_q^i \geq 0 \quad \forall i, q$

Principle component analysis (PCA)



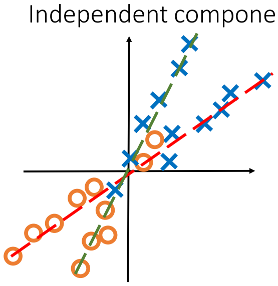
Start with
- D = {$\boldsymbol{x^1}$,…,$\boldsymbol{x^n}$} , data 0-center
- Component index: q=1

Loop
- Find direction of highest variance: $\boldsymbol{u_q}$
  - Ensure $|\boldsymbol{u_q}| = 1$
- Remove $\mathbf{u_q}$ from data:
  $D = \left\{\boldsymbol{x^1} - (\boldsymbol{x^1})^T \boldsymbol{u_q}, \cdots, \boldsymbol{x^n} - (\boldsymbol{x^n})^T \boldsymbol{u_q}\right\}$

We **require** $(\boldsymbol{u_i})^T \boldsymbol{u_j} = 0 \quad \forall i \neq j$

Thus, we guarantee $z_j^i = \boldsymbol{u}_j^T \boldsymbol{x}^i$

Independent component analysis (ICA)



Start with
- D = {$\boldsymbol{x^1}$,…,$\boldsymbol{x^n}$} , data 0-center

Find group(s) for each data point

Find direction for each group $\boldsymbol{u_q}$
- Ensure $|\boldsymbol{u_q}| = 1$

We do **not** require $(\boldsymbol{u_i})^T \boldsymbol{u_j} = 0 \quad \forall i \neq j$

Thus, we cannot guarantee $z_j^i = \boldsymbol{u}_j^T \boldsymbol{x}^i$

Evaluating components

- Components learned in order of descriptive power
- Compute reconstruction error for all data by using first v components:

$$error = \sum_i \left(\sum_j \left(x_j^i - \sum_{q=1}^{v} a_q^i \boldsymbol{u}_{q,j} \right)^2 \right)$$