Dimensionality Reduction, including by Feature Selection

www.cs.wisc.edu/~dpage/cs760

Goals for the lecture

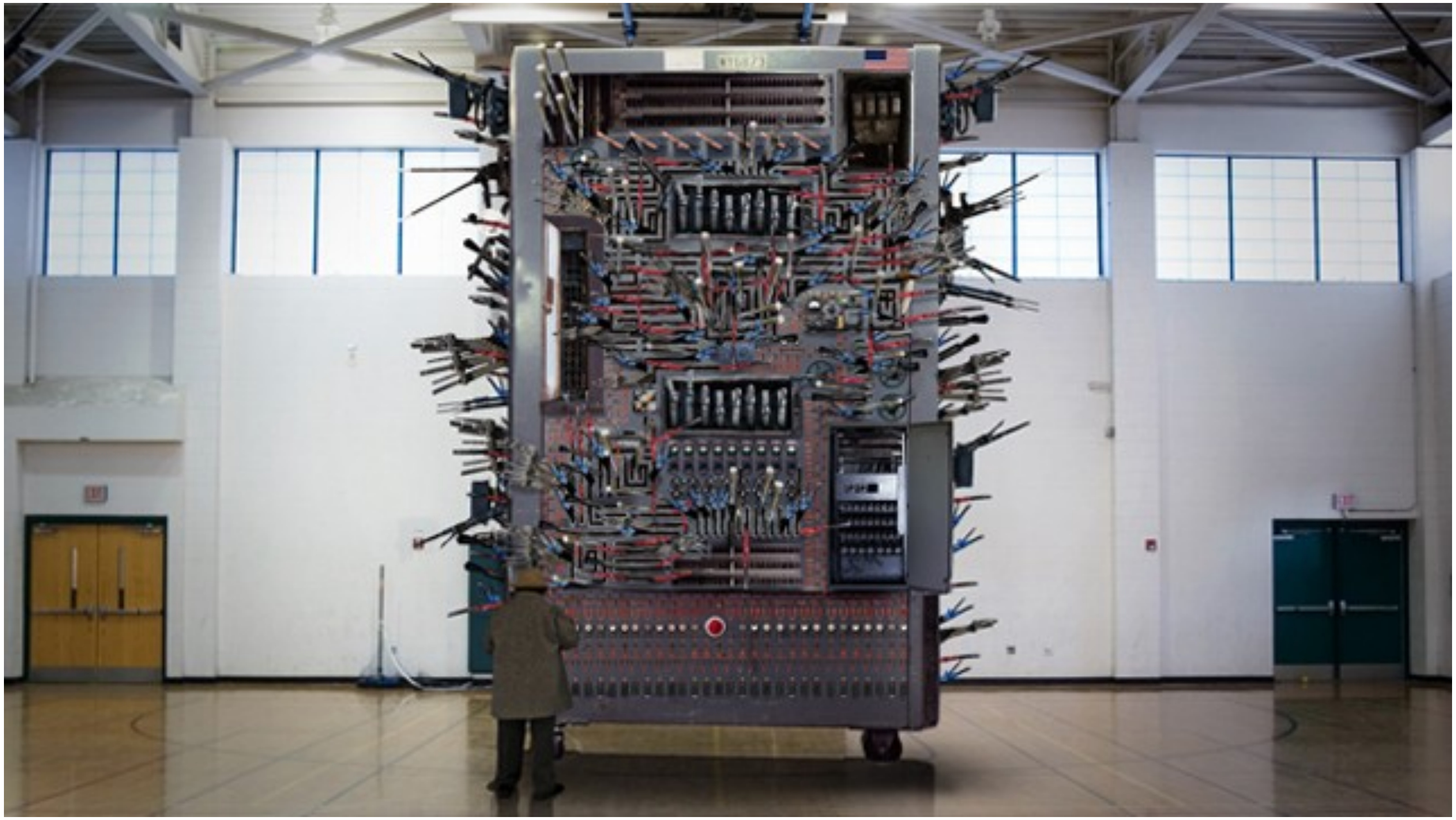
you should understand the following concepts

- filtering-based feature selection
- information gain filtering
- wrapper-based feature selection
- forward selection
- backward elimination
- recursive feature elimination (RFE)
- dimensionality reduction
- principal components analysis (PCA)
- remember that lasso also fits into this general area

Irrelevant and redundant features can lead to incomprehensible models and poor performance

the ONION®
America's Finest News Source

Florida To Experiment With New 600-Lever Voting Machines



Motivation for feature selection

1. We want models that we can interpret. We're specifically interested in which features are relevant for some task

2. We're interested in getting models with better predictive accuracy, and feature selection may help (can reduce overfitting)

3. We are concerned with efficiency. We want models that can be learned in a reasonable amount of time, and/or are compact and efficient to use

Motivation for feature selection

- some learning methods are sensitive to irrelevant or redundant features
  - k-NN
  - naïve Bayes
  - etc.

- other learning methods are ostensibly insensitive to irrelevant features (e.g. Weighted Majority) and/or redundant features (e.g. decision tree learners)

- empirically, feature selection is sometimes useful even with the latter class of methods [Kohavi & John, *Artificial Intelligence* 1997]

Information gain filtering

- select only those features that have significant information gain (mutual information with the class variable)

$$\text{InfoGain}(Y, X_i) = H(Y) - H(Y \mid X_i)$$

entropy of class variable (in training set)

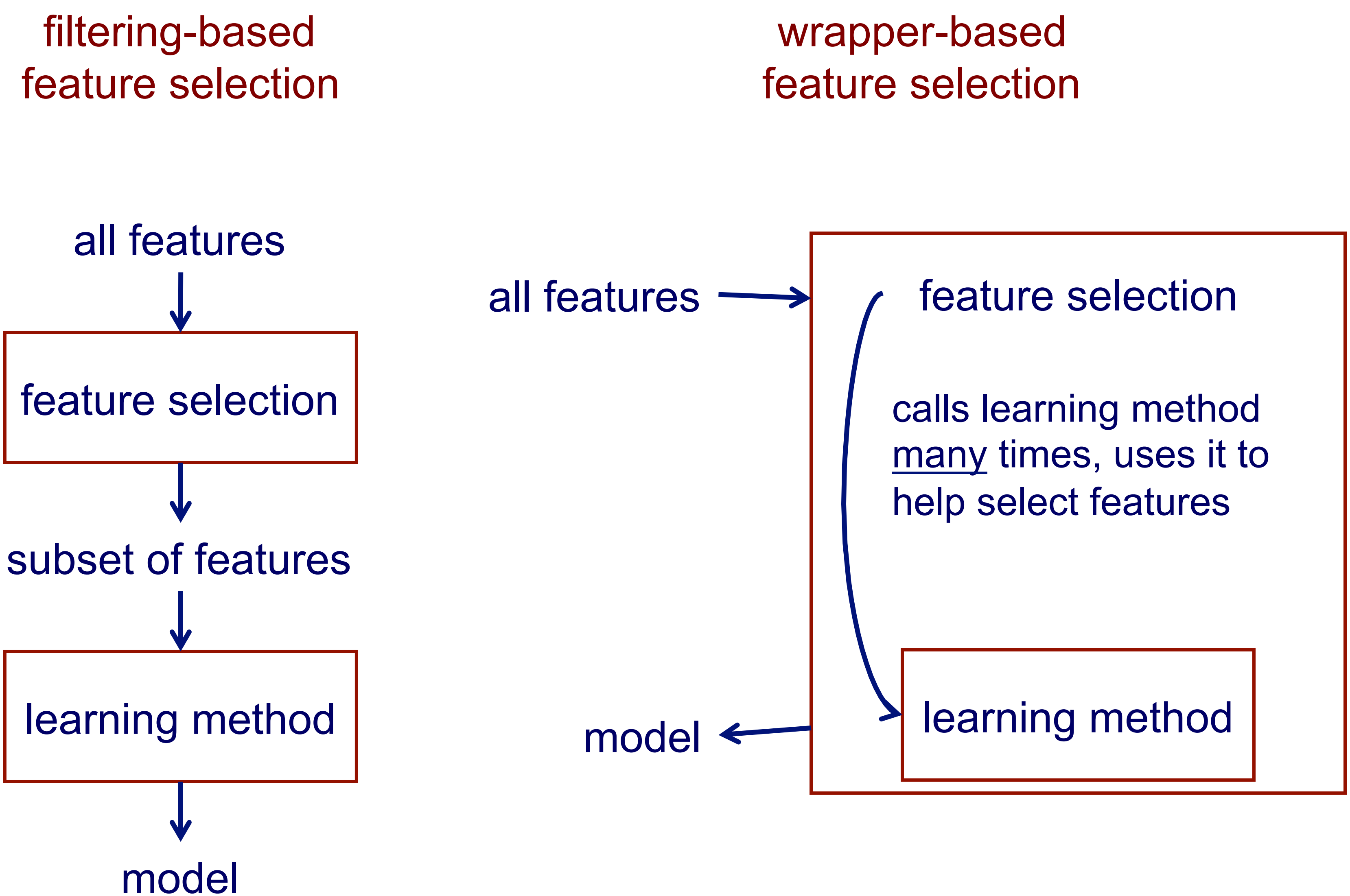
entropy of class variable given feature X_i

- unlikely to select features that are highly predictive only when combined with other features
- may select many redundant features

Other Filtering Methods

- gain ratio rather than info gain
- conditional mutual information (iterative, conditional on features selected so far)
- Ranking by Chi-square, correlation, p-value from significance test
- Add to previous methods an adjustment for *multiple comparisons problem* – if testing many features, some will look good by chance (false discovery rate, Bonferoni adjustment: multiply p-value by total features in data set). Doesn't change ranking but can affect how many features we keep.

Feature selection approaches



Feature selection as a search problem

state = set of features

start state = *empty* (forward selection)
or *full* (backward elimination)

operators

add/subtract a feature

scoring function

training or tuning-set or CV accuracy using learning method on a given state's feature set

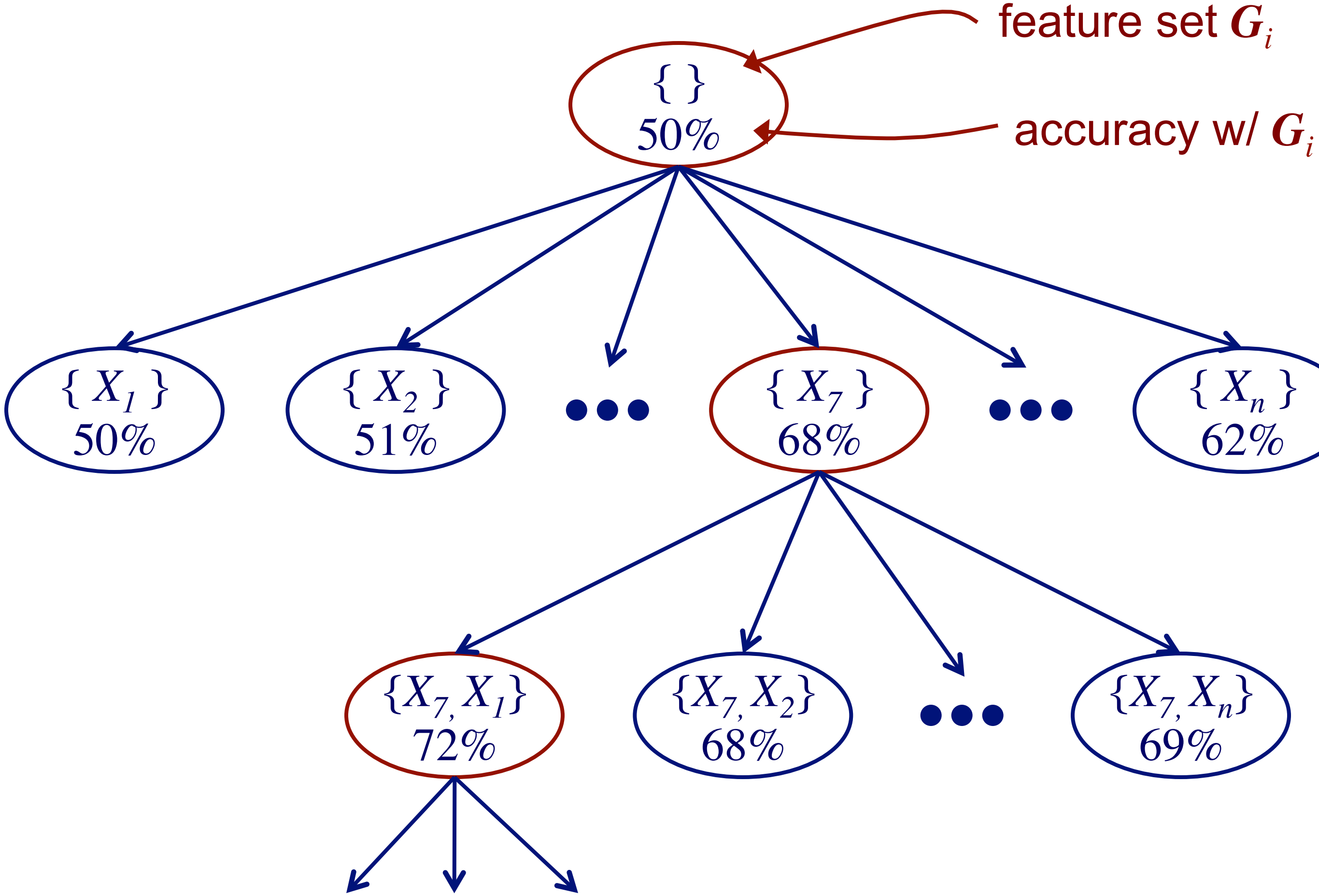
Forward selection

Given: feature set $\{X_i, \ldots, X_n\}$, training set D, learning method L

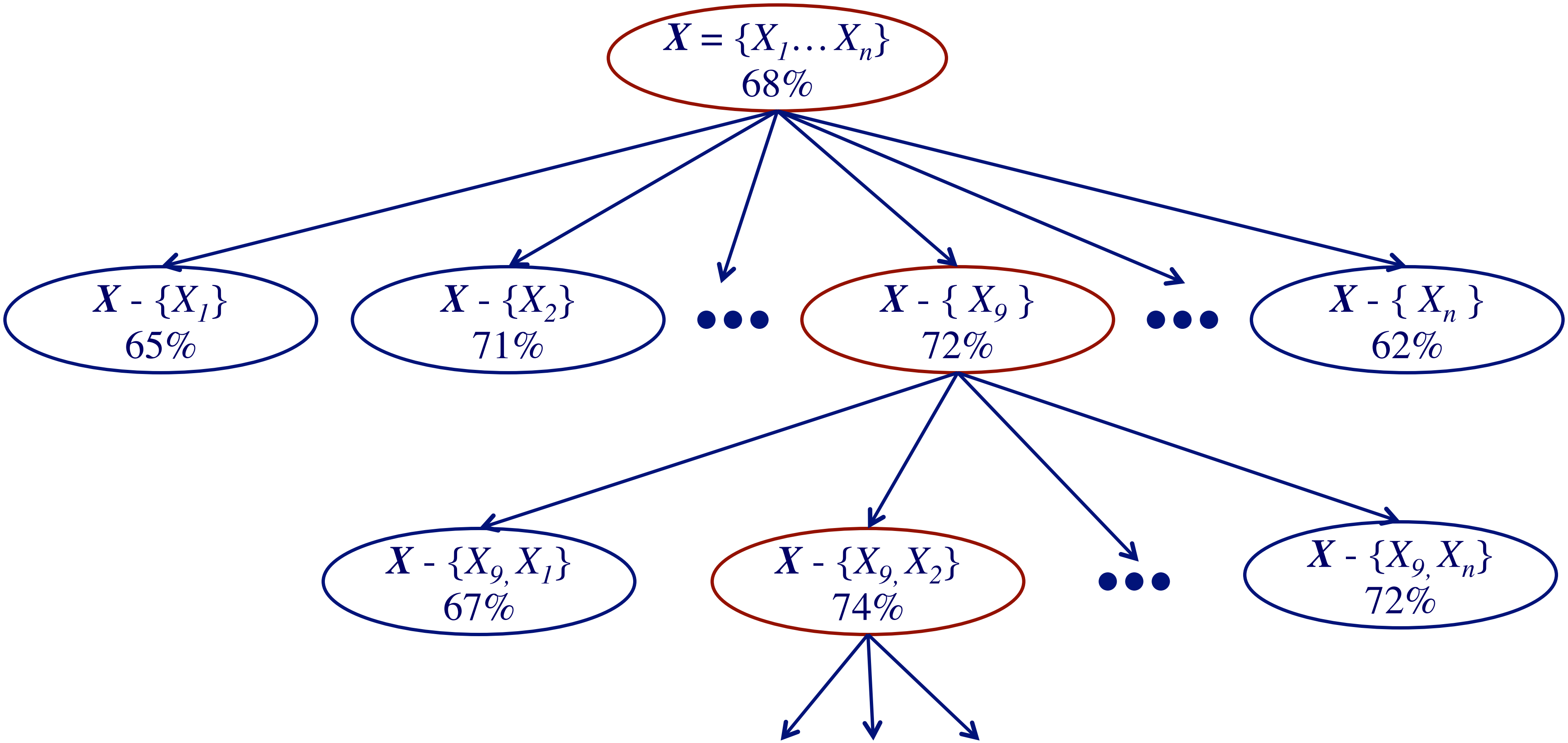
$\boldsymbol{F} \leftarrow \{\ \}$
while score of $\boldsymbol{F}$ is improving
    for $i \leftarrow 1$ to n do
        if $X_i \notin \boldsymbol{F}$
            $\boldsymbol{G}_i \leftarrow \boldsymbol{F} \cup \{ X_i \}$
            $Score_i$ = Evaluate($\boldsymbol{G}_i, L, D$)
    $\boldsymbol{F} \leftarrow \boldsymbol{G}_b$ with best $Score_b$
return feature set $\boldsymbol{F}$

scores feature set G by learning model(s) with L and assessing its (their) accuracy

Forward selection



Backward elimination



Forward selection vs. backward elimination

- both use a hill-climbing search

forward selection

- efficient for choosing a small subset of the features
- misses features whose usefulness requires other features (feature synergy)

backward elimination

- efficient for discarding a small subset of the features
- preserves features whose usefulness requires other features

Recursive Feature Elimination (RFE)

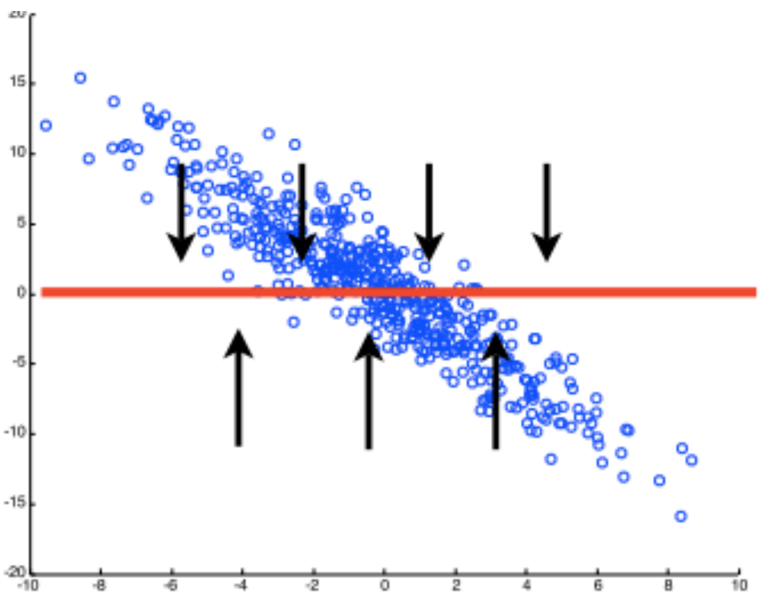
- Train model on all features
- Repeat until tuning set accuracy decreases:
  - Rank features by coefficient magnitude in last model
  - Remove lowest-ranked feature(s), e.g. bottom 10%
  - Retrain model
- Return previous model (prior to accuracy decrease)

RFE is a backwards selection wrapper method used with linear SVMs, but works for other linear methods too; it requires nontrivial modifications for nonlinear methods, since those do not yield coefficients on features.

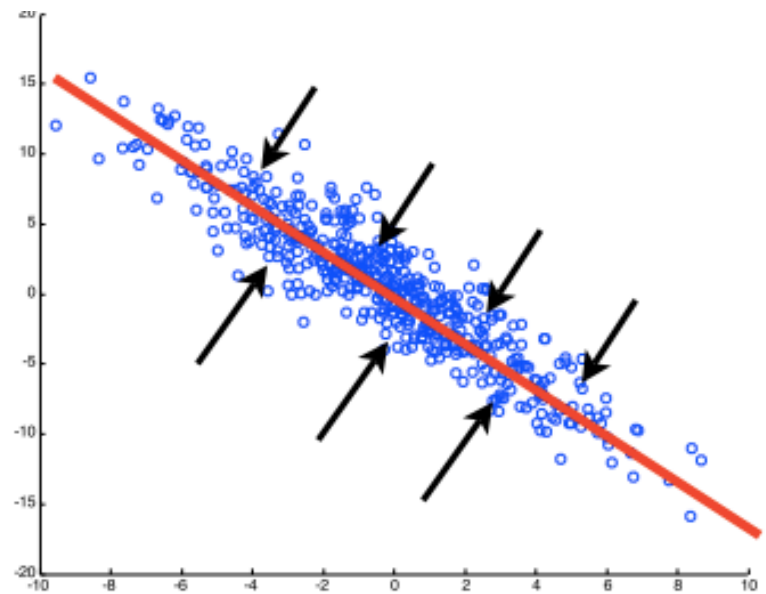
Dimensionality reduction

- *feature selection*: equivalent to projecting feature space to a lower dimensional subspace perpendicular to removed feature
- *dimensionality reduction*: allow other kinds of projection (e.g. PCA re-represents data using linear combinations of original features)

feature selection

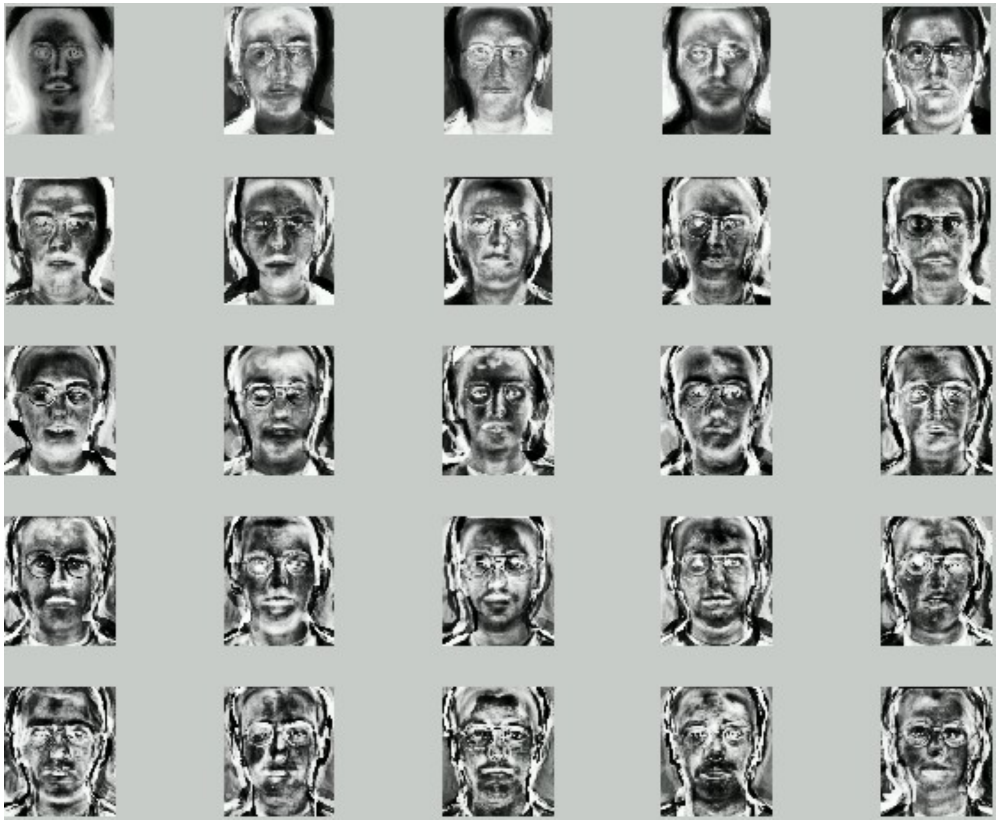


dimensionality reduction



Dimensionality reduction example

We can represent a face using all of the pixels in a given image (# features = # pixels)



More effective method: represent each face as a linear combination of *eigenfaces* (# features = 25)

Principal Components Analysis (PCA)

- Find a linear function of input features along which the data vary most: first component (see next slide)
- Repeatedly subtract component just added, and find next component
- Components are exactly the fixpoints for the following process: choose a feature vector and repeatedly multiply by covariance matrix, until reaching a vector (eigenvector) where matrix multiplication equals multiplication of vector by a constant (eigenvalue)

PCA: First Component

$$\mathbf{w}_{(1)} = \underset{\|\mathbf{w}\|=1}{\arg\max} \left\{ \sum_i \left(\mathbf{x}_{(i)} \cdot \mathbf{w} \right)^2 \right\}$$

Can then subtract contribution of this component to data matrix, and repeat to find next component, and repeat. But there are more efficient ways to find more/all components.

Reminder: standardization of data

- As a preprocessing step, we normalize the data so it is centered at 0 and has the same range of values by subtracting the mean and dividing by the standard deviation
- For $i = 1 \dots \dots N$

$$x_i \leftarrow \frac{x_i - \mu_i}{\sigma_i}$$

Reminder: Covariance Matrix

$$\Sigma = \begin{bmatrix} \mathrm{E}[(X_1-\mu_1)(X_1-\mu_1)] & \mathrm{E}[(X_1-\mu_1)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_1-\mu_1)(X_n-\mu_n)] \\ \mathrm{E}[(X_2-\mu_2)(X_1-\mu_1)] & \mathrm{E}[(X_2-\mu_2)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_2-\mu_2)(X_n-\mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n-\mu_n)(X_1-\mu_1)] & \mathrm{E}[(X_n-\mu_n)(X_2-\mu_2)] & \cdots & \mathrm{E}[(X_n-\mu_n)(X_n-\mu_n)] \end{bmatrix}$$

PCA

- It turns out that the solution to the maximization problem is just the eigen-decomposition of C, the covariance matrix where the vector w is an eigenvector of C.
- Let $\lambda_1, \dots \dots, \lambda_D$ *be the eigenvalues of* C *where* $\lambda_1 \geq \lambda_2 \geq \dots \dots \geq \lambda_D$ and $w_1, \dots \dots, w_D$ *be the corresponding eigenvectors.*
- Project $x_1, \dots \dots \dots, x_N$ onto the first d eigenvectors of C
    - For $i = 1, \dots \dots \dots, N$
        - $x_i \leftarrow \left(w_1^T x_i, \dots \dots \dots, w_d^T x_i\right)^T$
- The most common way to find the eigenvalues and eigenvectors of C is by Singular Value Decomposition

Comments on feature selection

- filtering-based methods are generally more efficient
- wrapper-based methods use the inductive bias of the learning method to select features
- forward selection and backward elimination are most common search methods in the wrapper approach, but others can be used [Kohavi & John, *Artificial Intelligence* 1997]; might consider lasso a wrapper approach, but lasso is more integrated
- feature-selection methods may sometimes be beneficial to get
  - more comprehensible models
  - more accurate models
- dimensionality reduction methods may sometimes be beneficial to get more accurate models