linear classifiers and Support Vector Machines
classification

- origins in pattern recognition
- considered a central topic in machine learning
- overlap between vision and learning unavoidable and very important
example: character recognition

classes

query
example: medical diagnoses

query

classes

Cancerous

Normal
example: assessing risk

Subject A: credit history, salary, marital status, education, ...

Subject B: credit history, salary, marital status, education, ...

classes
query

Will Default
Sell Credit Card
example: image classification

train a machine to classify

Smiling

Neutral
classification vs. clustering

- **supervised learning**: uncover recipe for classifying data based on relationships that are specified in training data

- **unsupervised learning**: unclear what the relationships are / how many of them exist; determine both simultaneously
training

“show” algorithm how it should behave through examples
testing
evaluate its performance on unseen data
some definitions

- recurring theme: treat data as collection of points in high-dimensional space
- associate label with each data point
goal

- develop algorithm that (given an unseen data point) can provide the correct label
goal

- develop algorithm that (given an unseen data point) can provide the correct label
brainstorm!
nearest neighbor classifier

idea: look for $n$ nearest neighbors, classify according to ratio of labels
nearest neighbor classifier

- idea: look for \( n \) nearest neighbors, classify according to ratio of labels

  - too few neighbors: classify as ‘unknown’
nearest neighbor classifier

- efficient neighbor search topic in computational geometry
  (hard problem in high-dimensions)
Perceptron

simple linear binary classifier invented in 1957 by Frank Rosenblatt (earliest feed-forward neural network)

\[
f(\vec{x}) = < \vec{w}, \vec{x} > + b
\]

\(\vec{x}\) : data point

\(f(\vec{x}) > 0\) : binary label (TRUE/FALSE)

\((\vec{w}, b)\) : linear parameters
geometry of a Perceptron

\[ f(x) = \langle \vec{w}, \vec{x} \rangle + b \]

\{x \mid f(x) = 0\}  

“decision boundary”
training rule

- for each data point $\vec{x}_i$:

$$\vec{w}' \leftarrow \vec{w} + (T - A)\vec{x}_i$$

$$b' \leftarrow b + (T - A)$$
update rule

\[ \mathbf{w} \]

\[ (T - A) = (0 - 1) = -1 \]
update rule

$(T - A) = (0 - 1) = -1$

$\vec{w}' \leftarrow \vec{w} - \vec{x}_i$

$b' \leftarrow b - 1$
update rule

simplest form of “backpropagation”

\[
(T - A) = (0 - 1) = -1
\]

\[
\vec{w}' \leftarrow \vec{w} - \vec{x}_i
\]

\[
b' \leftarrow b - 1
\]
Perceptron results
problems with Perceptron

- restricted to binary classification
- assumes data is linear separable
  - convergence? (learning rate)
- limited computational power: XOR
canonical variates

- idea: project data onto line that maximizes the ratio of separation between classes relative to the variance within classes

note: not principal components
canonical variates

- idea: project data onto line that maximizes the ratio of separation between classes relative to the variance within classes
canonical variates

- introduced by Sir Ronald Fisher in 1936 in the *Annals of Eugenics*
- used to discriminate populations based on set of characteristics
- often called ‘Fisher’s Linear Discriminant’
canonical variates
canonical variates

\[ \Sigma_A \]

\[ \vec{\mu}_A \]

\[ \vec{\mu}_B \]

\[ \vec{\mu} \]
canonical variates

\[ \sum_A \vec{v} \]

\[ \frac{\sigma_{between}^2(\vec{v})}{\sigma_{within}^2} = \frac{(\vec{v} \cdot \vec{\mu}_A - \vec{v} \cdot \vec{\mu}_B)^2}{\vec{v}^T \Sigma_A \vec{v} + \vec{v}^T \Sigma_B \vec{v}} \]
canonical variates

\[
\frac{\sigma^2_{\text{between}}(\tilde{v})}{\sigma^2_{\text{within}}} = \frac{(\tilde{v} \cdot (\tilde{\mu}_A - \tilde{\mu}_B))^2}{\tilde{v}^T (\Sigma_A + \Sigma_B) \tilde{v}}
\]
canonical variates

$$\sum_A A \quad \vec{v}$$

$$\sigma^2_{\text{between}}(\vec{v}) = \frac{\vec{v} \cdot (\vec{\mu}_A - \vec{\mu}_B)^2}{\vec{v}^T (\Sigma_A + \Sigma_B) \vec{v}}$$

$$\vec{v} \propto (\Sigma_A + \Sigma_B)^{-1} (\vec{\mu}_A - \vec{\mu}_B)$$
canonical variates

\[
\sum_A \vec{\nu} = \frac{\sigma_{\text{between}}^2 (\vec{\nu})}{\sigma_{\text{within}}^2} = \frac{\vec{\nu}^T S \vec{\nu}}{\vec{\nu}^T \Sigma \vec{\nu}}
\]

\[
S = \frac{1}{K-1} \sum_{i=1}^{K} (\vec{\mu}_i - \mu)(\vec{\mu}_i - \mu)^T
\]
canonical variates

\[ \bar{\mu} = \frac{1}{K} \sum_{i=1}^{K} \mu_i \]

\[ S = \frac{1}{K - 1} \sum_{i=1}^{K} (\bar{\mu}_i - \bar{\mu})(\bar{\mu}_i - \bar{\mu})^T \]
canonical variates

\[ \sum \]

\[ \vec{v} \propto (\Sigma_A + \Sigma_B)^{-1}(\vec{\mu}_A - \vec{\mu}_B) \]

\[ \bar{\mu} = \frac{1}{K} \sum_{i=1}^{K} \mu_i \]

\[ S = \frac{1}{K - 1} \sum_{i=1}^{K} (\vec{\mu}_i - \bar{\mu})(\vec{\mu}_i - \bar{\mu})^T \]

\[ \vec{v}^T S \vec{v} \]

\[ \vec{v}^T \Sigma \vec{v} \]
canonical variates

\[ \bar{\mu} = \frac{1}{K} \sum_{i=1}^{K} \mu_i \]

\[ S = \frac{1}{K-1} \sum_{i=1}^{K} (\bar{\mu}_i - \bar{\mu})(\bar{\mu}_i - \bar{\mu})^T \]

\[ S\vec{v} + \lambda \Sigma \vec{v} = 0 \]

\[ \vec{v}^T \Sigma \vec{v} \]

\[ \vec{v}^T S \vec{v} \]
Support Vector Machines (SVMs)

- linear classifier originally proposed by Vladimir Vapnik in 1963
- idea: find best decision boundary
Support Vector Machines

“maximum margin”
Support Vector Machines

“support vectors”
some notation

\((\vec{x}_1, y_1), \cdots, (\vec{x}_n, y_n)\)

\[ f(\vec{x}) = \langle \vec{w}, \vec{x} \rangle + b \]

\( \vec{x} \): data point

\( y \in -1, 1 \): label

\( \text{sign}(f(\vec{x})) \): classifier output (-/+)

\((\vec{w}, b)\): linear parameters
geometry of SVM

\[ y_i (\langle \vec{w}, \vec{x}_i \rangle + b) > 0 \]
geometry of SVM

\[ y_i (\langle \overline{w}, \overline{x}_i \rangle + b) \geq 1 \]
geometry of optimal solution

\[ y_i(\langle \vec{w}, \vec{x}_i \rangle + b) \geq 1 \]
geometry of optimal solution

punchline: finding widest margin is equal to minimizing $|w|$ subject to constraints

$y_i(<\vec{w}, \vec{x}_i> + b) \geq 1$
induced QP problem

minimize $\frac{1}{2} ||w||^2$, subject to $c_i (\vec{w} \cdot \vec{x}_i - b) \geq 1$
dual form

\[ \frac{1}{2} \bar{w} \cdot \bar{w} - \sum_{1}^{N} \alpha_i (y_i (\bar{w} \cdot \bar{x}_i + b) - 1) \]

\[ \sum_{1}^{N} \alpha_i y_i = 0 \]

\[ \bar{w} = \sum_{1}^{N} \alpha_i y_i \bar{x}_i \]

maximize \[ \sum_{i}^{N} \alpha_i - \frac{1}{2} \bar{\alpha} Q \bar{\alpha}^T \]

\[ Q_{ij} = y_i y_j \bar{x}_i \cdot \bar{x}_j \]

subject to \[ \alpha_i \geq 0 \text{ and } \sum_{i=1}^{N} \alpha_i y_i = 0 \]
Support Vector Machines

\[ \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \]
multicategory SVMs

- solve nested set of binary problems
  - “one vs. many”: split-off one category, merge others; recurse
  - “One vs. one”: create $k(k-1)/2$ models and look for consistency
nonlinear decision boundaries
nonlinear decision boundaries

- project onto higher-dimensional space where planar boundary does exist:

http://www.dtreg.com/svm.htm
nonlinear decision boundaries

\[(x, y, x^2 + y^2)\]
nonseparable training sets

Non-separable training sets

Use linear separation, but admit training errors.

Penalty of error: distance to hyperplane multiplied by error cost $C$. 

http://www.dtreg.com/svm.htm
underfitting and overfitting

- goal: model complexity = data complexity
- test by holding out random subsets of training data

http://www.dtreg.com/svm.htm