ENSEMBLE METHODS

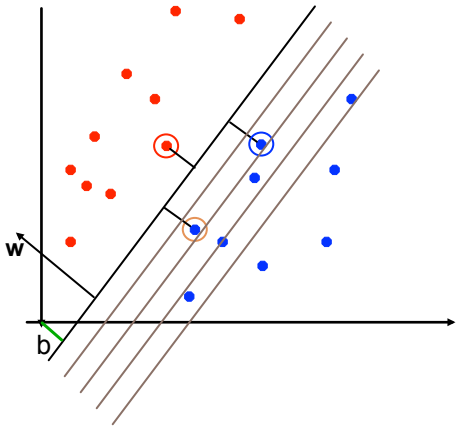
Today

- Reading
  - AIMA 18.10-18.11
- Goals
  - (Recap support vector machines)
  - Ensembles of classifiers

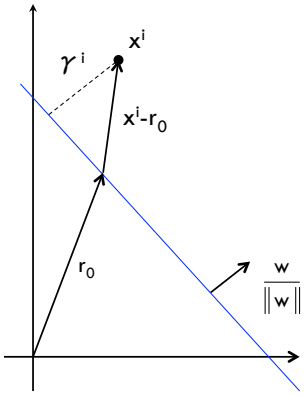
What defines a hyperplane?

A hyperplane is defined by:

- A vector w
  - Perpendicular to the hyperplane
  - Often called the "weight" vector
- A scalar b
  - Selects the hyperplane that is distance b from the origin from among all possible hyperplanes



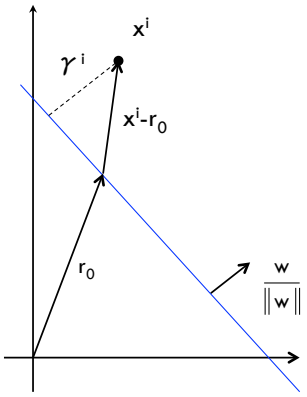
Deriving a support vector machine



- x^i is the i^{th} training example
- r_0 is any point on the decision boundary
- w is the weight vector which is perpendicular to the decision boundary
- The geometric margin of x_i is given by γ^i
- The geometric margin is equal to the length of the projection of (x^i-r_0) onto the vector w
- The length of the projection of (x^i-r_0) onto the vector w is given by:

$$\gamma^{(i)} = \frac{(x^{(i)} - r_0) \cdot w}{||w||}$$
$$= \frac{w \cdot x^{(i)} - r_0 \cdot w}{||w||}$$
$$= \frac{w \cdot x^{(i)} + b}{||w||}$$
$$\text{where } b = -r_0 \cdot w$$

Deriving a support vector machine



- Define γ to be minimum of the geometric margins $\gamma^{(i)}$
- Then the optimization task becomes:

$$\max_{\gamma,w,b} \ \gamma \quad \text{such that } y^{(i)}\frac{(w^\intercal x^{(i)}+b)}{||w||} \geq \gamma \quad \forall i$$

- Define $\hat{\gamma} = \gamma||w||$. Then we can rewrite the optimization problem as:

$$\max_{\hat{\gamma},w,b} \ \frac{\hat{\gamma}}{||w||} \quad \text{such that } y^{(i)}(w^\intercal x^{(i)}+b) \geq \hat{\gamma} \quad \forall i$$

- Since the optimization is invariant to the scaling of w, we scale w so that equals 1

$$\max_{w,b} \ \frac{1}{||w||} \quad \text{such that } y^{(i)}(w^\intercal x^{(i)}+b) \geq 1 \quad \forall i$$

- The maximum of $1/z$ is equivalent to the minimum of z^2 which is equivalent to the minimum of $0.5\ z^2$

$$\min_{w,b} \ \frac{1}{2}||w||^2 \text{ such that } y^{(i)}(w^\intercal x^{(i)}+b) \geq 1 \quad \forall i$$

Recap: Solving the Optimization Problem

$$\min_{w,b} \ \frac{1}{2}||w||^2 \text{ such that } y^{(i)}(w^\intercal x^{(i)}+b) \geq 1 \quad \forall i$$

- Need to optimize a *quadratic* function subject to *linear* constraints
- Quadratic optimization problems are a well-known class of mathematical programming problem and many algorithms exist for solving them
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* (a scalar value) is associated with every constraint in the primary problem

Solving the Optimization Problem

$$\min_{w,b} \frac{1}{2}||w||^2 \text{ such that } y^{(i)}\left(w^\intercal x^{(i)} + b\right) \geq 1 \quad \forall i$$

$$\max_\alpha \min_{w,b} \frac{1}{2}||w||^2 - \sum_{i=1}^{N} \alpha_i \left[y^{(i)}\left(w^\intercal x^{(i)} + b\right) - 1\right]$$

Dual

Lagrange multipliers

$$\max_\alpha \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y^{(i)} y^{(j)} x^{(i)} x^{(j)}$$

$$\text{subject to } \alpha_i \geq 0 \text{ and } \sum_i \alpha_i y^{(i)} = 0$$

Solving the Optimization Problem

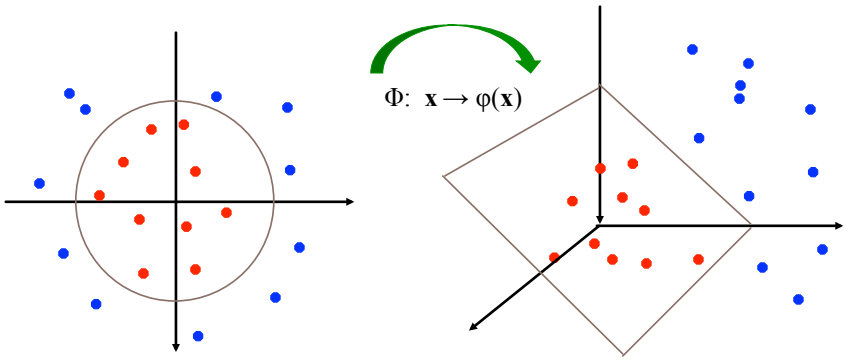
- The solution has the form:

$$w = \sum_{i=1}^{N} \alpha_i y^{(i)} x^{(i)} \text{ and } b = y^{(i)} - w^\intercal x^{(i)} \text{ for any } x^{(i)} \text{ s.t. } \alpha_i \neq 0$$

- Each non-zero alpha indicates corresponding x_i is a support vector
- The classifying function has the form: $g(x_i) = \text{sign}\left(\sum_i \alpha_i y^{(i)} x^{(i)} x + b\right)$
- Relies on an inner product between the test point x and the support vectors x_i

Non-linear SVMs

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



The "Kernel" trick

- The linear classifier relies on an inner product between vectors $x_i^T x_j$

$$g(x_i) = \text{sign}\left(\sum_i \alpha_i y^{(i)} x^{(i)} x + b\right)$$

- If every example is mapped into a high-dimensional space via some transformation $\Phi: \mathbf{x} \rightarrow \varphi(\mathbf{x})$ then the inner product becomes:

$$g(x_i) = \text{sign}\left(\sum_i \alpha_i y^{(i)} \varphi(x^{(i)})^T \varphi(x) + b\right)$$

- A kernel function is some function that corresponds to a dot product in some transformed feature space:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$$

The "Kernel" trick

- The kernel K may be cheaper to compute then the transformation φ
  - Implictly do the transformation

$$\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix}$$

$$\begin{aligned} K(x,z) &= \left(\sum_{i=1}^{n} x_i z_i\right)\left(\sum_{j=1}^{n} x_i z_i\right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j z_i z_j \\ &= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j) \end{aligned}$$

Kernels

Why use kernels?

- Make non-separable problem separable.
- Map data into better representational space

Common kernels

- Linear
- Polynomial $\mathbf{K(x,z) = (1+x^Tz)^d}$
- Radial basis function (infinite dimensional space)

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2}$$

SVMs Summary

- The classifier is a decision boundary (separating hyperplane)
- Most "important" training points are support vectors which define the hyperplane
- Quadratic optimization algorithms can identify which training points are support vectors (vectors with non-zero Lagrange multipliers)
- In the dual formation and in classifying an example, the training points appear only inside inner products
- Kernels allow us to efficiently map data to higher dimensional space

Ensembles of Classifiers

Which classifier should I use?

- Is there a classifier that is optimal for all classification problems?
- Factors to take into account:
  - How much training data is available?
  - How simple/complex is the problem? (linear vs. nonlinear decision boundary)
  - How noisy/skewed is the training data?
  - How stable is the problem over time?
  - Is it a singly-labeled or multi-labeled problem? Are the labels correlated?

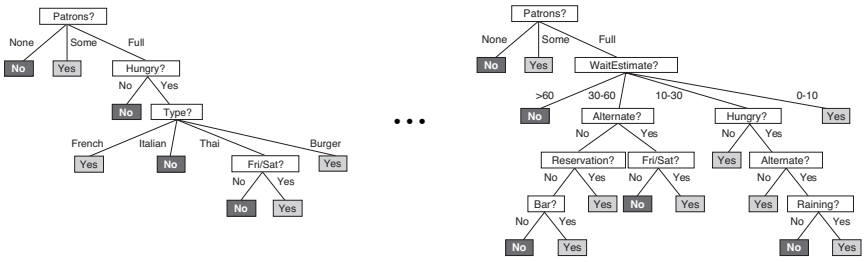
How Much Data?

- Learning theory (PAC learning)
  - Gives theoretical bounds on how much training data you need for a given accuracy (AIMA 18.5)
- Very Little
  - There are empirical results that naïve Bayes should do well in such circumstances (Ng and Jordan 2002 NIPS)
  - The interesting theoretical answer is to explore semi-supervised training methods: Bootstrapping, EM over unlabeled documents, …
  - The practical answer is to get more labeled data as soon as you can
- A reasonable amount of data
  - Start with SVMs
- A lot of data?
  - expensive methods like SVMs (train time) or kNN (test time) are quite impractical
  - Naïve Bayes! - with lots of data, simple methods work well

Ensembles of Classifiers

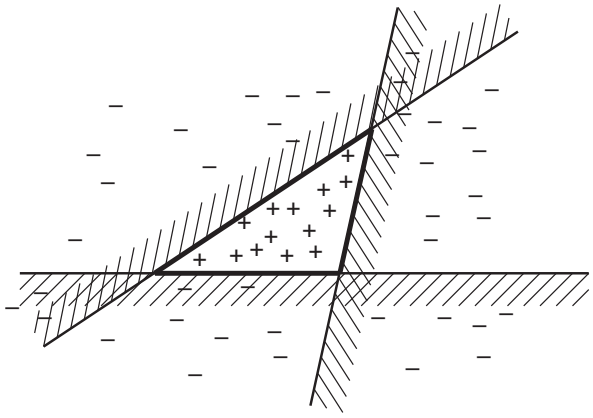
- Ensemble - A group of items viewed as a whole rather than individually
- An ensemble of classifiers – A group of classifiers whose predictions are combined to produce one final prediction
- Benefits
  - Harder to make a wrong prediction
  - More expressive hypothesis

Ensemble of decision trees



- Combine the prediction of each decision tree using majority vote
- Variation of this called a Random Forest

Ensemble of linear classifiers



- More expressive than any one linear classifier by itself

Ensemble Schemes

- Multi-expert combination methods
  - Global - All classifiers generate an output and all outputs are used in some way
    - e.g. weighting, voting, averaging
  - Local – A gating model chooses one (or very few) of the classifiers responsible for generating the output for a specific input
    - e.g. mixture of experts
- Multi-stage combination
  - Classifiers are trained with, or tested on, only the instances where the previous classifiers are not accurate enough
    - e.g. cascading

Boosting

- Boosting is one of the most common forms of constructing an ensemble of classifiers
  - Learn a series of weak classifiers, i.e. classifiers whose performance is slightly better than random chance
  - Weight each weak classifier to create a final strong classifier
  - Often the weight for each classifier is proportional to its accuracy
- A well-known boosting algorithm is AdaBoost short for "Adaptive Boosting" (Freund and Schapire 1995)

AdaBoost

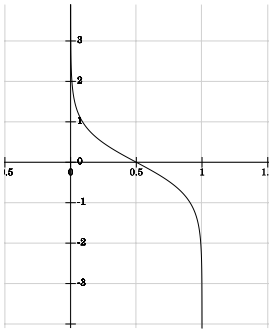
function ADABOOST($examples$, L, K) **returns** a weighted-majority hypothesis
  inputs: $examples$, set of N labeled examples $(x_1, y_1), \ldots, (x_N, y_N)$
    L, a learning algorithm
    K, the number of hypotheses in the ensemble
  local variables: **w**, a vector of N example weights, initially $1/N$
    h, a vector of K hypotheses
    z, a vector of K hypothesis weights

  for $k = 1$ **to** K **do**
    $\mathbf{h}[k] \leftarrow L(examples, \mathbf{w})$
    $error \leftarrow 0$
    for $j = 1$ **to** N **do**
      if $\mathbf{h}[k](x_j) \neq y_j$ **then** $error \leftarrow error + \mathbf{w}[j]$
    for $j = 1$ **to** N **do**
      if $\mathbf{h}[k](x_j) = y_j$ **then** $\mathbf{w}[j] \leftarrow \mathbf{w}[j] \cdot error/(1 - error)$
    $\mathbf{w} \leftarrow$ NORMALIZE(**w**)
    $\mathbf{z}[k] \leftarrow \log(1 - error)/error$
  return WEIGHTED-MAJORITY(**h**, **z**)

AdaBoost

- Generates a sequence of weak classifiers each focusing on the errors of the previous classifier
- AdaBoost returns a strong classifier, i.e. a classifier that can perfectly classify the training data for large enough K
- To classify a new example x:

$$h(x) = \text{sign}\left(\sum_{k=1}^{K} \mathbf{z}[k]\, h_k(x)\right) \quad \text{where} \quad \mathbf{z}[k] = \log\left(\frac{1 - error}{error}\right)$$

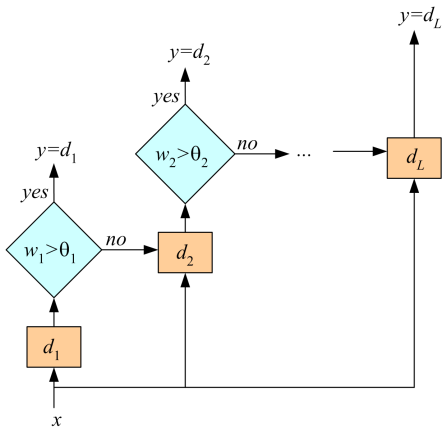


Bagging

- Short for "Bootstrap aggregating"
- Given training set D
  - Generate M new training sets D_i where $|D_i| < |D|$ by sampling from D with replacement
  - This is a statistical technique known as bootstrapping
  - Train a classifier on each of the M new training sets
  - Combine output of M classifiers using averaging or voting
- Random Forests (Breimen, 2001)
  - Bagged decision trees

Cascading classifiers

- Order classifiers by complexity, e.g. representational complexity
- Use i^{th} classifier d_i only if previous classifiers are not confident
- Good with high precision/ low recall classifiers



Ensemble methods

- Boosting
  - Weighted training sets
  - Ex: Adaboost
- Bagging
  - Resampled training sets
  - Ex: Random forests
- Cascading
  - Ordered collection of classifiers