Feature Selection for SVMs
by J. Weston, S. Mukherjee, O. Chapelle, M. Pontil,
T. Poggio, V. Vapnik

Sambarta Bhattacharjee
For EE E6882 Class Presentation
Sept. 29 2004

Review of
Support Vector Machines
A support vector machine classifies data as +1 or -1

- A decision boundary with maximum margin looks like it should generalize well.
• Minimize True Risk
\[ R(\theta) = E_x \{ L(x, y, \theta) \} = \int E_x \{ L(x, y, \theta) \} dx dy \in [0,1] \]
• Minimize Guaranteed Risk instead
\[ R(\theta) \leq J(\theta) = R_{\text{emp}}(\theta) + \frac{\log(\frac{2hn}{N}) + 1 - \log(\frac{1}{\delta})}{N} \]
• VC dimension \( h = \# \) of training points that can be shattered
  - eg. \( h = 3 \) for 2-D linear classifier
• To minimize \( J \), minimize \( h \)
• To minimize \( h \), maximize margin \( M \)
• Structural Risk Minimization: minimize \( R_{\text{emp}} \) while maximizing margin

Support Vector Machines

• Maximize margin subject to classifying all points correctly
  
  \[ \min \frac{1}{2} \|w\|^2 \quad \text{subject to} \quad y_i (w^T x_i + b) - 1 \geq 0 \]

  \[ m = \frac{2}{\|w\|} \quad H_+ \quad w^T x + b = +1 \\
  H_- \quad w^T x + b = -1 \]

  The support vector machine
  
• To classify:
  \[ f(x) = \text{sign}(x^T (\sum_{i} \alpha_i y_i x_i) + b) = \text{sign}(\sum_{i} \alpha_i y_i x_i^T x_i + b) \]
Support Vector Machines

- Support Vectors: 

\[ w^T x_i + b = \pm 1 \]

- Dual Problem

\[
\max \sum \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \\
\text{subject to } \sum \alpha_i y_i = 0 \quad \& \quad \alpha_i \geq 0
\]
Support Vector Machines

- Dual Problem
\[
\max \sum \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \\
\text{subject to } \sum \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0
\]
- Nonseparable?
\[
0 \leq \alpha_i \leq C
\]
- Nonlinear?
\[
k(x,y) = \phi(x)^T \phi(y)
\]

Cover's theorem on the separability of patterns: A pattern classification problem cast in a high-dimensional space is more likely to be linearly separable.
SVM Matlab Implementation

\[
\max \sum \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j x_i^T x_j \quad \text{subject to} \quad \sum \alpha_i y_i \leq 0 \quad \& \quad \alpha_i \geq 0
\]

\[
k(x,y) = \phi(x)^T \phi(y)
\]

%To train..
for i=1:N
for j=1:N
H(i,j)=Y(i)*Y(j)*svm_kernel(ker,X(i,:),X(j,:));
end
end
alpha=qp(H,f,A,b,vlb,vub);

X=QP(H,f,A,b) solves the quadratic programming problem:
\[
\min 0.5x^T H x + f' x \quad \text{subject to:} \quad Ax \leq b
\]

X=QP(H,f,A,b,VLB,VUB) defines a set of lower and upper bounds on the
design variables, X, so the soln is always in the range VLB <= X <= VUB.

SVM Matlab Implementation

\[
f(x) = \text{sign}(x^T w + b) = \text{sign}\left(\sum \alpha_i y_i x_i^T x + b\right)
\]

%To classify..
for i=1:M
for j=1:N
H(i,j)=Ytrn(j)*svm_kernel(ker,Xtst(i,:),Xtrn(j,:));
end
end
Ytst=sign(H*alpha+b0);

•Karush-Kuhn-Tucker Conditions (KKT): solve value of b
on margin (for nonzero alphas) have: \( w^T x_i + b = y_i \)
using known w, compute b for each support vector
\( b_i = y_i - w^T x_i \quad \forall i: \alpha_i > 0 \) 
then... \( b = \text{average}(b_i) \).
Support Vector Machines

• Summary
  – Use Matlab’s qp( ) to perform optimization on training points and get parameters of hyperplane
  – Use hyperplane to classify test points

Feature Selection for SVMs
Here's some data

60 data points

Row 20 is a 11-D data point

The data is classified as +1 (black) or -1 (white)

Col 3 is the 3rd dimension

Dimension 6 is pretty useless in classification
We want to find the relative discriminative ability of each dimension, and throw away the least discriminative dimensions

Dimensionality Reduction

- Improve generalization error
- Need less training data (avoid curse of dimensionality)
- Speed, computational cost
- (qualitative) Find out which features matter
- For SVMs, irrelevant features hurt performance
Formal problem

\[ \tau(\sigma, \alpha) = \int \text{loss functional} \]

- Weight each feature by 0 or 1
  \[ \sigma = (1, 0, 1, 1, 0), \ x = (x_1, x_2, x_3, x_4, x_5), \ x \cdot \sigma = (x_1, 0, x_3, x_4, 0) \]
- Which set of weights minimizes (average expected) loss?
  - Specifically, if we want to keep m features out of n, which set of weights minimizes loss subject to the constraint that weight vector sums to m?
- We don't know \( P(x, y) \)
Formal solution
(approximations)

\((x \cdot \sigma)\)

- Weight each feature by 0 or 1
• Weight each feature by 0 or 1
• Weight each feature by a real valued vector
• First approach suggests combinatorial search over all weights (intractable for large dimensionality)
• Second approach brings you closer to a gradient descent solution

\[ \approx (x \cdot \sigma) \]

• There’s a weight vector that minimizes (average expected) loss \( \tau(\sigma, \alpha) \)
• There’s a weight vector that minimizes expected leave-one-out error probability for weighted inputs \( EP_{err} \)
• There’s a weight vector that minimizes (average expected) loss \( \tau(\sigma, \alpha) \)
• There’s a weight vector that minimizes expected leave-one-out error probability for weighted inputs \( E\text{P}_{\text{err}} \)
• Let's pretend these are the same ("wrapper method")

A wrapper method is a search through the space of feature subsets using the estimated accuracy from an induction algorithm trained on preprocessed data as a measure of goodness of a particular subset [1].

• Theorem

\[
E\text{P}_{\text{err}} \leq \frac{1}{l} E[R^2W^2(\alpha^0)]
\]

• Data in sphere of size R, separable with margin M (1/M^2=W^2)
• Theorem

\[ EP_{err} \leq \frac{1}{l} E[R^2W^2(\alpha^0)] \]

• Data in sphere of size R, separable with margin M (1/M^2=W^2)
• To minimize error probability, let’s minimize R^2W^2 instead

• Someone gives us a contour map, telling us which direction to walk in weight vector space to get highest increase in R^2W^2
• We take a small step in the opposite direction
• Check map again
• Repeat above steps (until we stop moving)

This is gradient descent
Feature Selection for SVMs

\[ \frac{\partial K_\sigma(x_i, x_j)}{\partial \sigma_k} \]

- Choose kernel, find gradient, proceed with above algorithm to find weights
- Throw away lowest weighted dimension(s) after gradient descent finds minimum, repeat until you have specified number of dimensions left
  - E.g. You have 123 dimensions (41 average X Y Z coordinates of person’s joints) for walking/running classification. You want to reduce to 6 (maybe these will be the X Y Z coordinates of both ankles)
  - Throw away worst 2 dimensions after each run of algorithm until you have desired number left
Feature Selection for SVMs

- Throw away worst q dimensions after each run of algorithm until you have desired number left
- As we increase q, fewer calls to qp algorithm and faster performance

For this data

![Image of data visualization]
We get this weighting

Dimension 6 is the first to go

For this data

(images unrolled into one long vector)
We get this weighting

And...

• Automatic dimensionality reduction? (user doesn’t have to specify number of dimensions)
References


