Support Vector Machine Tutorial

16.711
Fall 2013
Prof. Karen Daniels
Overview

• Ideas from Bishop Chapter 7: Sparse Kernel Machines (but different derivations)

• Motivation: Data Classification
  – Example: Glass (UCI Machine Learning Repository)

• Formulating the Optimization Problem

• Solving the Optimization Problem
  – Example: (later) R interface to libsvm
  – Example: (later) GPU?
Motivation: Data Classification

• Example: Glass (UCI Machine Learning Repository)
  – Goal: predict type of glass based on attributes
  • Crime scene forensics motivation

– 214 data samples (see values within R using Glass.R script)

Attribute Information:

1. Id number: 1 to 214
2. RI: refractive index
3. Na: Sodium (unit measurement: weight percent in corresponding oxide, as are attributes 4-10)
4. Mg: Magnesium
5. Al: Aluminum
6. Si: Silicon
7. K: Potassium
8. Ca: Calcium
9. Ba: Barium
10. Fe: Iron
11. Type of glass: (class attribute)
   -- 1 building_windows_float_processed
   -- 2 building_windows_non_float_processed
   -- 3 vehicle_windows_float_processed
   -- 4 vehicle_windows_non_float_processed (none in this database)
   -- 5 containers
   -- 6 tableware
   -- 7 headlamps

Source: http://archive.ics.uci.edu/ml/datasets/Glass+Identification
Motivation: Data Classification

• Support Vector Machine (SVM): Vapnik
  – Supervised learning using training data
  – Binary (2-class) classification
  – Classify glass data using libsvm R interface
    • multi-class classification
      – Pair-wise “one-on-one” classification
      – Followed by ensemble approach: voting mechanism

Source: Meyer’s “Support Vector Machines: The Interface to libsvm in package e1071”
Formulating the Optimization Problem

• SVM binary classification approach

Technique only works if data is linearly separable in feature space.

Formulating the Optimization Problem

• **Training Goal**: Find “maximum margin” hyperplane
  – Minimizes over-fitting risk
  – First touches “support vectors”

• **Testing Goal**: Use hyperplane’s parameters to classify each test point
  – Evaluate an expression using support vectors
  – Sparse if few support vectors

Set of Hyperplanes

\[ f(x) = \langle w \cdot x \rangle + b = 0 \]
defines a set of hyperplanes (see backup slide)

-1 represents the other data class

(+1 represents one data class)

Functional margin (not scale invariant)

Geometric margin (scale invariant, using normalized weight vector)

Source: Lee 2005 and Vapnik
Optimization Problem

\[
\max \frac{2}{\|w\|}, \text{ subject to } y_i \langle w \cdot x_i \rangle + b \geq 1, \quad \forall i = 1, \ldots, N
\]

where \(N\) = number of data points; \(x_i\) is a training point; \(y_i\) is a training point’s classification

This is equivalent to (minimize instead of maximize):

\[
\min \frac{\|w\|}{2}, \text{ subject to } y_i \langle w \cdot x_i \rangle + b \geq 1, \quad \forall i = 1, \ldots, N
\]

To avoid square root in the norm we use:

\[
\min \frac{\|w\|^2}{2}, \text{ subject to } y_i \langle w \cdot x_i \rangle + b \geq 1, \quad \forall i = 1, \ldots, N
\]

Ugly optimization problem!

Source: Lee 2005 and Vapnik and Cristianini/Shawe-Taylor
Optimization Problem

$$\min \frac{\|w\|^2}{2}, \text{ subject to } y_i (w \cdot x_i) + b \geq 1, \quad \forall i = 1, \ldots, N$$

Ugly (primal constrained) optimization problem!

Lagrangian relaxation to the rescue!

Bring constraints $y_i (w \cdot x_i) + b \geq 1$ into objective function and introduce Lagrange multipliers $\alpha$'s.

$$\max_\alpha \min_{w, b} L(w, b, \alpha) = \frac{\|w\|^2}{2} - \sum_{i=1}^N \alpha_i \left( y_i (w \cdot x_i) + b \right) \geq 1 \quad \forall i = 1, \ldots, N$$

“weights” $\alpha_i \geq 0, \quad \forall i = 1, \ldots, N$

(minimize $L$ with respect to $w, b$ and maximize with respect to $\alpha$ to formulate Lagrangian dual)

(source: Lee 2005 and Vapnik)
Optimization Problem

\[
\max_{\alpha} \min_{w, b} \quad L(w, b, \alpha) = \frac{||w||^2}{2} - \sum_{i=1}^{N} \alpha_i \langle w \cdot x_i \rangle + b \cdot 1 \quad \forall i = 1, \ldots, N
\]

\[
\alpha_i \geq 0, \quad \forall i = 1, \ldots, N
\]

Still somewhat ugly due to max min...

But, KKT conditions \[\alpha_i \langle w \cdot x_i \rangle + b \cdot 1 = 0\] for this type of optimization problem allow only points nearest to hyperplane (support vectors) to have positive \(\alpha\)'s. Problem may therefore be sparse.

And, fortunately, solution is a saddle point of \(L\): \((w', b', \alpha')\)

Characterize saddle point using partial derivatives to obtain simplifying conditions:

\[
\frac{\partial}{\partial b} L(w', b', \alpha') = -\sum_{i=1}^{N} \alpha'_i y_i = 0 \Rightarrow \sum_{i=1}^{N} \alpha'_i y_i = 0
\]

Add this to constraints and substitute into objective function.

\[
\frac{\partial}{\partial w} L(w', b', \alpha') = w' - \sum_{i=1}^{N} \alpha'_i y_i x_i = 0 \Rightarrow w' = \sum_{i=1}^{N} \alpha'_i y_i x_i
\]

So optimal hyperplane is linear combination of data. Substitute this into objective function on next slide...

Source: Lee 2005 and Vapnik
Optimization Problem

\[
\max_\alpha \min_{w,b} \; L(w,b,\alpha) = \frac{\|w\|^2}{2} - \sum_{i=1}^N \alpha_i \langle w \cdot x_i \rangle + b \langle 1 \rangle_i \quad \forall i = 1, \ldots, N
\]

Recall: KKT conditions, optimal hyperplane is linear combination of data, and

\[
\|w\|^2 = \langle w \cdot w \rangle
\]

\[
\max_\alpha \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i \cdot x_j \rangle - \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i \cdot x_j \rangle - b \sum_i \alpha_i y_i + \sum_i \alpha_i
\]

= 0 since \( \sum_i \alpha_i' y_i = 0 \)

\[
\max_\alpha \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i \cdot x_j \rangle - \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i \cdot x_j \rangle + \sum_i \alpha_i
\]

\[
\max_\alpha \left( \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i \cdot x_j \rangle \right)
\]

Subject to constraints: \( \alpha_i \geq 0, \sum_i \alpha_i y_i = 0 \) \( \forall i = 1, \ldots, N \)

Source: Lee 2005 and Vapnik
Solving the Optimization Problem

\[
\max_{\alpha} \left( \sum_{i} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i \cdot x_j \rangle \right)
\]

Subject to constraints: \( \alpha_i \geq 0, \sum_{i} \alpha_i y_i = 0 \quad \forall i = 1, \ldots, N \)

How to find \( \alpha \)'s?

This is now a quadratic programming (QP) problem: quadratic in \( \alpha \)'s.
Solve using existing QP techniques to find \( \alpha \)'s.

Can be computationally expensive for large problems!

Also we have not yet taken into account that we will work in **feature space**!
Using Support Vectors for Classification

Recall KKT conditions for this type of optimization problem allow only points nearest to hyperplane (support vectors) to have positive $\alpha$’s.

$$\alpha_i = \begin{cases} > 0 & \text{if } x_i \text{ is a support vector} \\ = 0 & \text{otherwise} \end{cases}$$

- So, solving QP identifies support vectors.
- Now use support vectors to build classifying function.
- Need $b$ for this because hyperplane equation is of the form: $f(x) = \langle w \cdot x \rangle + b = 0$

$$\text{sign}(f(x)) \text{ where } f(x) = \sum_{i \in \text{support vectors}} \alpha_i y_i \langle x_i \cdot x \rangle + b_0$$

and where $b_0 = -\frac{1}{2} \left( \sum_{\text{support vectors} \in \{+1\}} \langle w_0 \cdot x_{sv}^{+1} \rangle + \sum_{\text{support vectors} \in \{-1\}} \langle w_0 \cdot x_{sv}^{-1} \rangle \right)$

and $x_{sv}^1 \in \text{support vector for class } \{+1\}$, $x_{sv}^{-1} \in \text{support vector for class } \{-1\}$

Where does equation for $b_0$ come from?

Source: Lee 2005 and Vapnik
Using Support Vectors for Classification

Start with these (support vectors):

-1 represents the other data class

(+1 represents one data class)

(scale invariant, using normalized weight vector)

Source: Lee 2005 and Vapnik
Using Support Vectors for Classification

Solve for $b$ in: $\langle w \cdot x^+ \rangle + b = +1$ and $\langle w \cdot x^- \rangle + b = -1$

$\begin{align*} b &= +1 - \langle w \cdot x^+ \rangle \\
     b &= -1 - \langle w \cdot x^- \rangle \end{align*}$

(dividing each by 2)

$\begin{align*} \frac{b}{2} &= +1/2 - \langle w \cdot x^+ \rangle / 2 \\
     \frac{b}{2} &= -1/2 - \langle w \cdot x^- \rangle / 2 \end{align*}$

(add equations)

$b = - \frac{1}{2} (\langle w \cdot x^+ \rangle + \langle w \cdot x^- \rangle)$

(use all support vectors)

$b_0 = - \frac{1}{2} \left( \sum_{\text{support vectors} \in \{ +1 \}} \langle w_0 \cdot x_{sv}^+ \rangle + \sum_{\text{support vectors} \in \{ -1 \}} \langle w_0 \cdot x_{sv}^- \rangle \right)$

The last missing piece of the puzzle is to transition to feature space...

Source: Lee 2005 and Vapnik
Working in Feature Space

• Recall SVM binary classification approach

Working in Feature Space

• $\Phi$ is a nonlinear mapping.
• Feature space is very high dimensional.
• But we don’t need to explicitly know $\Phi$!
• Recall that our focus is on dot product.
  – Feature space must support dot product (Hilbert space).
• Kernel function can specify dot product of image points: $K : X \times X \rightarrow R \quad K(x_i, x_j) = \langle \Phi(x_i) \cdot \Phi(x_j) \rangle$
• Kernel function must be symmetric positive definite (positive eigenvalues) by Mercer’s Theorem.

Working in Feature Space

• Example kernel function:
  – Gaussian radial basis function (RBF) kernel:
    \[ K(x, z) = e^{-\frac{\|x-z\|^2}{2\sigma}} \]
    • where \( \sigma \) = width of RBF kernel

• Choice of kernel function and its parameter(s) are important for successful SVM classification
  – data dependent

• Optimization formulation for feature space becomes:
  \[
  \max_{\alpha} \left( \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \Phi (x_i), \Phi (x_j) \rangle \right)
  \]
  \[
  \max_{\alpha} \left( \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K (\xi_i, x_j) \right)
  \]

Gaussian Radial Basis Function Kernel

Figure 7.2  Example of synthetic data from two classes in two dimensions showing contours of constant $y(x)$ obtained from a support vector machine having a Gaussian kernel function. Also shown are the decision boundary, the margin boundaries, and the support vectors.

$f(x)$ for our notation

Source: Bishop textbook
Summarizing so far...

- Examine data
- Select kernel function and its parameters
- Build $N \times N$ kernel matrix for training data
- Solve QP optimization to obtain support vectors (ones with positive $\alpha$’s)

\[
\max_{\alpha} \left( \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle K(\mathbf{x}_i), K(\mathbf{x}_j) \rangle \right)
\]

Subject to constraints: $\alpha_i \geq 0$, $\sum_i \alpha_i y_i = 0$ \quad $\forall i = 1, \ldots, N$

- Use support vectors to classify new test point $x$:

\[
\text{sign}(f(x)) \quad \text{where} \quad f(x) = \left( \sum_{i \in \text{support vectors}} \alpha'_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle \right) + b_0
\]

and where $b_0 = -\frac{1}{2} \left( \sum_{\text{support vectors} \in \{+1\}} \langle \mathbf{w}_0, \mathbf{x}_{sv}^{+1} \rangle + \sum_{\text{support vectors} \in \{-1\}} \langle \mathbf{w}_0, \mathbf{x}_{sv}^{-1} \rangle \right)$

Source: Lee 2005 and Vapnik
Soft-Margin Classifier

• But, what if data is noisy?
  – Improper classification vs. overfitting risk...

• Modify SVM formulation to allow some points to be on \textit{wrong} side of hyperplane.
  – Introduce new penalty term:
    • slack variable $\xi_i$ measures “wrong” distance of point $x_i$
    • constant $C$ controls “fraction” of wrong classifications

\[
\min \frac{\|w\|^2}{2}, \text{ subject to } y_i \langle w \cdot x_i \rangle + b \geq 1, \quad \forall i = 1, \ldots, N \quad \text{(from slide 8)}
\]

\[
\min \frac{\|w\|^2}{2} + C \sum_i \xi_i, \text{ subject to } y_i \langle w \cdot x_i \rangle + b \geq 1 - \xi_i, \quad \xi_i \geq 0, \forall i = 1, \ldots, N \quad 0 \leq \alpha_i \leq C
\]

Note: Bishop allows asymmetric error

Source: Lee 2005 and Vapnik
References

- “Support Vector Machines: The Interface to libsvm in package e1071” by Meyer.
BACKUP SLIDES
Distance of Point from Plane

Project \( \mathbf{w} \) onto \( \mathbf{v} \) to obtain length \( q \) using:

\[
\cos \theta = \frac{\mathbf{v} \cdot \mathbf{w}}{||\mathbf{v}|| ||\mathbf{w}||} = \frac{q}{||\mathbf{w}||} \Rightarrow \frac{\mathbf{v} \cdot \mathbf{w}}{||\mathbf{v}||} = q
\]

\( q \) can be further expressed as:

\[
\left\langle \left( \frac{\mathbf{v}}{||\mathbf{v}||} \right) \cdot \mathbf{w} \right\rangle + b
\]

where \( b \) is offset of plane from origin.

Source: Wolfram MathWorld
Lagrangian Relaxation

$$\min \frac{\|w\|^2}{2}, \text{ subject to } y_i (w \cdot x_i) + b \geq 1, \quad \forall i = 1, \ldots, N$$

Ugly (*primal constrained*) optimization problem!

Convex optimization problem: minimize quadratic function under linear inequality constraints.

Convex optimization techniques have come far, but Lagrangian relaxation works well here.

Lagrangian relaxation to the rescue!

Bring constraints $y_i (w \cdot x_i) + b \geq 1$ into objective function and introduce Lagrange multipliers $\alpha$’s.

$$\max_{\alpha} \min_{w,b} L(w, b, \alpha) = \frac{\|w\|^2}{2} - \sum_{i=1}^{N} \alpha_i (y_i (w \cdot x_i) + b - 1), \quad \forall i = 1, \ldots, N$$

"weights" $\alpha_i \geq 0, \quad \forall i = 1, \ldots, N$

(minimize $L$ with respect to $w, b$ and maximize with respect to $\alpha$ to formulate *Lagrangian dual*)

What is all this about??

Source: Lee 2005, Vapnik, Cristianini
Lagrangian Relaxation

• Consider a simpler example: \[ \min c^T x, \text{ subject to } A x \geq b, x \geq 0 \]
  
  – \( A \) (known) is \( m \times n \) matrix
  – \( x \) (unknown variables) is \( n \times 1 \) column vector
  – \( c \) (known) is \( 1 \times n \) row vector
  – \( b \) (known) is \( m \times 1 \) column vector
  – Example for \( m=3, n=2 \):

\[
A = \begin{bmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22} \\
a_{31} & a_{32}
\end{bmatrix}, \quad x = \begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}, \quad b = \begin{bmatrix}
b_1 \\
b_2 \\
b_3
\end{bmatrix}, \quad c = \begin{bmatrix}
1 \\
c_2
\end{bmatrix}
\]

• We can solve this using linear programming
  – but if objective function is nonlinear or \( x \) is integer, then problem is difficult and Lagrangian relaxation can sometimes be more efficient...
Lagrangian Relaxation

Subject to:
\[ A x \geq b \Rightarrow b - A x \leq 0 \]

\[ x \geq 0 \]

Unknown variables: \( x \)

Known parameters: \( A, c, b \)

Source: Reeves
Lagrangian Relaxation

\[
\min c \bar{x} + \alpha(b - A \bar{x})
\]

Subject to:

\[
A x \geq b
\]

\[
x \geq 0
\]

\[
\alpha \geq 0
\]

Unknown variables: \(x, \alpha\)

Known parameters: \(A, c, b\)

Note: Value of \(1\) > value of \(2\) since \(\alpha \geq 0\) and \(\langle -A \bar{x} \rangle \leq 0\) (adding term \(\leq 0\) to objective function).

Source: Reeves
Lagrangian Relaxation

Lagrangian Lower Bound Program (LLBP)

\[
\min \ c^T x + \alpha (b - A^T x)
\]

Subject to: (note removed \( A^T x \geq b \) constraints)

\[
\begin{align*}
x &\geq 0 \\
\alpha &\geq 0
\end{align*}
\]

Unknown variables: \( x, \alpha \)

Known parameters: \( A, c, b \)

Note: Value of 2 > value of 3 since we remove constraints.
Lagrangian Dual Program

\[
\begin{align*}
\max_{\alpha \geq 0} & \quad \min \left( c^T x + \alpha (b - A x) \right) \\
\text{Subject to:} & \quad x \geq 0 \\
& \quad \alpha \geq 0
\end{align*}
\]

Unknown variables: \( x, \alpha \)

Known parameters: \( A, c, b \)

Note: Value of 2 > value of 3 since we remove constraints.

Source: Reeves
Lagrangian Relaxation

- Search $\alpha$ space using technique such as subgradient optimization
  - Initialize $\alpha$'s (e.g. 0) and user-defined parameter $\pi$
  - Initialize upper bound $Z_{UB}$ from some heuristic
  - Solve LLBP using current $\alpha$’s to obtain $Z_{LB}$
  - Define subgradients $G$’s for relaxed constraints
  - Calculate step size using gap $(Z_{UB} - Z_{LB})$ and $\pi$ and $G$’s
  - Take a step in subgradient direction
    - Update $\alpha$’s

Iterate until stopping criteria satisfied

Source: Reeves
Lagrangian Relaxation

Putting it all together...

$$\min \frac{||w||^2}{2}, \text{ subject to } y_i \langle w \cdot x_i \rangle + b \geq 1, \ \forall i = 1, \ldots, N$$

Ugly (primal constrained) optimization problem!

Lagrangian relaxation to the rescue!

Bring constraints $y_i \langle w \cdot x_i \rangle + b \geq 1$ into objective function and introduce Lagrange multipliers $\alpha$'s.

$$\max_{\alpha} \min_{w,b} L(w,b,\alpha) = \frac{||w||^2}{2} - \sum_{i=1}^{N} \alpha_i \left( y_i \langle w \cdot x_i \rangle + b \right) - 1, \ \forall i = 1, \ldots, N$$

“weights” $\alpha_i \geq 0, \ \forall i = 1, \ldots, N$

(minimize $L$ with respect to $w, b$ and maximize with respect to $\alpha$ to formulate Lagrangian dual)

Source: Lee 2005, Vapnik, Cristianini