Large Scale Support Vector Machines

Huanle Xu
Motivation

• Introduce the widely used classification tool: Support Vector Machine (SVM)
• Understand the model and parameter estimation method in terms of big data
Motivation

Suppose we have 50 photographs of elephants and 50 photos of tigers.

We digitize them into 100 x 100 pixel images, so we have $x \in \mathbb{R}^n$ where $n = 10,000$.

Now, given a new (different) photograph we want to answer the question: is it an elephant or a tiger? [we assume it is one or the other.]
What if there are millions of photos, how to make the SVM training scalable?
SVMs: History

• SVMs introduced in COLT-92 by Boser, Guyon & Vapnik. Became rather popular since.

• Theoretically well motivated algorithm: developed from Statistical Learning Theory (Vapnik & Chervonenkis) since the 60s.

• Empirically good performance: successful applications in many fields (bioinformatics, text, image recognition, . . . )
SVMs: History

• Centralized website: www.kernel-machines.org.

• Several textbooks, e.g. “An introduction to Support Vector Machines” by Cristianini and Shawe-Taylor is one.

• A large and diverse community work on them: from machine learning, optimization, statistics, neural networks, functional analysis, etc.
Linear SVMs

• Data
  – Training examples: \((x_1, y_1), \ldots, (x_n, y_n)\)
  – Each \(x_i \in \mathbb{R}^d, y_i \in \{+1, -1\}\)
  – Want to find a hyperplane \(y = w^T x + b\) to separate “+” from “-”

• What’s the best hyperplane defined by \(w\)?
Largest Margin

• Distance from the separating hyperplane corresponds to the “confidence” of prediction

• Example: We have more confidence to say A and B belong to “+” than C
Largest Margin

- **Support Vectors:** Examples closest to the hyperplane
- **Margin** $\rho$ : width of separation between support vectors of classes.
Largest Margin

- Distance from example to the separator is:

\[ r = y \frac{w^T x + b}{\|w\|} \]

- Proof:

\[ x' - x \parallel w, \text{ unit vector is } w/\|w\|, \]
so line is \( r w/\|w\|, x' = x - yrw/\|w\| \)
since \( x' \) is on the separator, \( w^T x' + b = 0 \)
so \( w^T (x - yrw/\|w\|) = 0, w = \sqrt{(w^T w)} \)
so \( w^T x - yr\|w\| + b = 0 \),
then we get \( r = y \frac{w^T x + b}{\|w\|} \)
Largest Margin

- Assume that all data is at least distance 1 from the hyperplane, then the following constraints follow for a training set \( \{(x_i, y_i)\}_{i=1}^{n} \)
  \[
y_i(w^T x_i + b) \geq 1
  \]
- For support vectors, the inequality becomes an equality

- Recall that
  \[
r = y \frac{w^T x + b}{\|w\|}
  \]
- Margin is:
  \[
  \rho = \frac{2}{\|w\|}
  \]
Linear SVMs

• Note that we assume that all data points are **linearly separated** by the hyperplane.

• The margin is **invariant** to scaling of parameters.
  – i.e. by changing $w, b$ to $5w, 5b$, the margin doesn’t change
Linear SVMs

• **Maximize** the margin
  – Good according to intuition, theory (VC dimension) & practice

• The problem of linear SVMs is formulated as:

\[
\max_w \rho = \frac{2}{\|w\|}
\]

\[
s.t. \quad y_i(w^T x_i + b) \geq 1 \quad \forall i = 1, \ldots, n
\]

• An equivalent form is:

\[
\min_w \frac{1}{2}\|w\|^2
\]

\[
s.t. \quad y_i(w^T x_i + b) \geq 1 \quad \forall i = 1, \ldots, n
\]
Non-Linear Separable SVMs

• In reality, training samples are usually **not linearly separable**.

• **Soft Margin Classification**
  
  – Idea: **allow** errors but introduce **slack variable** $\xi_i$ to **penalize** errors
  
  – Still try to minimize training set errors, and to place hyperplane “far” from each class (large margin)
Soft Margin Classification

• The problem becomes:

\[
\min_{\mathbf{w}} \frac{1}{2} \| \mathbf{w} \|^2 + C \sum \xi_i \\
\text{s.t. } y_i (w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i = 1, \ldots, n
\]

– Minimize \( \| \mathbf{w} \|^2 \) plus the number of training mistakes

– Set C using cross validation
Soft Margin Classification

- If point $x_i$ is on the wrong side of the margin then get penalty $\xi_i$
- Thus all mistakes are not equally bad!

For each datapoint:
If margin $\geq 1$, don’t care
If margin $< 1$, pay linear penalty
Slack Penalty C

\[
\min_w \frac{1}{2} \|w\|^2 + C \sum \xi_i \\
\text{s.t. } y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i = 1, \ldots, n
\]

• What is the role of penalty C:
  
  – \( C = 0 \): can set \( \xi_i \) to anything, then \( w=0 \) (basically ignore the data)
  
  – \( C = \infty \): Only want \( w,b \) to separate the data
Soft Margin Classification

- SVM in the “natural” form

\[
\arg\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \max\{0, 1 - y_i (w^T x_i + b)\}
\]

- SVM uses “Hinge Loss”:

\[
\min_{w} \frac{1}{2} \|w\|^2 + C \sum \xi_i \\
\text{s.t. } y_i (w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i = 1, \ldots, n
\]
Non-linear Separable SVMs

• Linear classifiers *aren’t* complex enough sometimes.
  – Map data into a *richer feature space* including non-linear features
  – Then construct a *hyperplane* in that space so all other equations are the same
Non-linear Separable SVMs

- Formally, process the data with:
  \[ x \mapsto \Phi(x) \]
- Then learn the map from \( \Phi(x) \) to \( y \)
  \[ f(x) = w \cdot \Phi(x) + b \]
Example: Polynomial Mapping

\[ \Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3 \]

\[(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2)\]
Example: MNIST

- Data: 60,000 training examples, 10,000 test examples, 28x28
- Linear SVM has around 8.5% test error. Polynomial SVM has around 1% test error.
MINIST Results

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Test Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>linear</td>
<td>8.4%</td>
</tr>
<tr>
<td>3-nearest-neighbor</td>
<td>2.4%</td>
</tr>
<tr>
<td>RBF-SVM</td>
<td>1.4%</td>
</tr>
<tr>
<td>Tangent distance</td>
<td>1.1%</td>
</tr>
<tr>
<td>LeNet</td>
<td>1.1%</td>
</tr>
<tr>
<td>Boosted LeNet</td>
<td>0.7%</td>
</tr>
<tr>
<td>Translation invariant SVM</td>
<td>0.56%</td>
</tr>
</tbody>
</table>

Choosing a good mapping $\Phi(\cdot)$ (encoding prior knowledge + getting right complexity of function class) for your problem improves results.
SVM: How to Estimate \( w, b \)

• We take the soft margin classification for example:

\[
\begin{align*}
\min_w & \quad \frac{1}{2} \|w\|^2 + C \sum \xi_i \\
\text{s.t.} & \quad y_i (w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i = 1, \ldots, n
\end{align*}
\]

• Standard way: Use a solver!
  
  – Solver: software for finding solutions to “common” optimization problems, e.g. LIBSVM (http://www.csie.ntu.edu.tw/~cjlin/libsvm/)

• Problems: Solvers are \textit{inefficient} for big data!
SVM: How to Estimate w, b

- Want to estimate w, b!

- Alternative approach:
  - Want to minimize $f(w, b)$

  $$f(w, b) = \frac{1}{2} \sum_{j=1}^{d} (w^{(j)})^2 + C \sum_{i=1}^{n} \max\{0, 1 - y_i (\sum_{j=1}^{d} w^{(j)} x_i^{(j)} + b)\}$$

  - How to minimize convex functions $f(z)$

  - Use gradient descent: $\min_z f(z)$

  - Iterate: $z_{t+1} \leftarrow z_t - \eta f'(z_t)$
SVM: How to Estimate $w$?

- Want to minimize $f(w, b)$:

$$f(w, b) = \frac{1}{2} \sum_{j=1}^{d} (w^{(j)})^2 + C \sum_{i=1}^{n} \max\{0, 1 - y_i (\sum_{j=1}^{d} w^{(j)} x_i^{(j)} + b)\}$$

Empirical loss $L$

- Compute the gradient $\nabla (j)$ w.r.t $w^{(j)}$

$$\nabla (j) = \frac{\partial f(w, b)}{\partial w^{(j)}} = w^{(j)} + C \sum_{i=1}^{n} \frac{\partial L(x_i, y_j)}{\partial w^{(j)}}$$

$$\frac{\partial L(x_i, y_j)}{\partial w^{(j)}} = \left\{ \begin{array}{ll} 0 & \text{if } y_i (w \cdot x_i + b) \geq 1 \\ -y_i x_i^{(j)} & \text{otherwise} \end{array} \right.$$

35
SVM: How to Estimate $w$?

• Gradient descent:
  Iterate until convergence:
  • For $j = 1, \ldots, d$
    - Evaluate: $\nabla(j) = \frac{\partial f(w,b)}{\partial w(j)} = w^j + C \sum_{i=1}^{n} \frac{\partial L(x_i,y_i)}{\partial w(j)}$
    - Update: $w^{(j)} = w^{(j)} - \eta \nabla(j)$
      $\eta \ldots$ learning rate parameter
      $C \ldots$ regularization parameter

• Problem:
  – Computing $\nabla(j)$ takes $O(n)$ time
    • $n \ldots$ size of the training dataset
SVM: How to Estimate \( w \)?

• **Stochastic Gradient Descent**
  
  – Instead of evaluating gradient over all examples, evaluate it for each **individual** training example

\[
\nabla(j, i) = w^{(j)} + C \sum_{i=1}^{n} \frac{\partial L(x_i, y_i)}{\partial w^{(j)}}
\]

• **Stochastic gradient descent:**

  Iterate until full convergence:

  • For \( i = 1, \ldots, n \)
    
    – For \( j = 1, \ldots, d \)
      
      * Evaluate: \( \nabla(j, i) \)
      
      * Update: \( w^{(j)} \leftarrow w^{(j)} - \eta \nabla(j, i) \)
Optimization “Accuracy”

For optimizing $f(w,b)$ within reasonable quality, SGD-SVM is super fast.
SGD vs. Batch Conjugate Gradient

- **SGD** on full dataset vs. **Batch Conjugate**
  - **Gradient** on a sample of $n$ training examples

**Bottom line:** Doing a simple (but fast) SGD update many times is better than doing a complicated (but slow) BCG update a few times
Practical Considerations

• Need to choose learning rate $\eta$ and $t_0$

\[ w_{t+1} \leftarrow w_t - \frac{\eta_t}{t + t_0} \left( w_t + C \frac{\partial L(x_i, y_i)}{\partial w} \right) \]

• Leon suggests:
  – Choose $t_0$ so that the expected initial updates are comparable with the expected size of the weights
  – Choose $\eta$:
    • Select a small subsample
    • Try various rates $\eta$ (e.g., 10, 1, 0.1, 0.01, …)
    • Pick the one that most reduces the cost
    • Use $\eta$ for next 100k iterations on the full dataset
Practical Considerations

- **Sparse Linear SVM:**
  - Feature vector $x_i$ is sparse (contains many zeros)
    - Do not do: $x_i = [0, 0, 0, 1, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, \ldots]$  
    - But represent $x_i$ as a sparse vector $x_i = [(4, 1), (9, 5), \ldots]$  
  - Can we do the SGD update more efficiently?
    $$w \leftarrow w - \eta \left( w + C \frac{\partial L(x_i, y_i)}{\partial w} \right)$$
  - Approximated in 2 steps:
    $$w \leftarrow w - \eta C \frac{\partial L(x_i, y_i)}{\partial w}$$
    $$w \leftarrow w (1 - \eta)$$

  **Cheap:** $x_i$ is sparse and so few coordinates $j$ of $w$ will be updated  
  **Expensive:** $w$ is not sparse, all coordinates need to be updated
Practical Considerations

• **Solution 1:** \( w = s \cdot v \)
  
  – Represent vector \( w \) as the product of scalar \( s \) and the vector \( v \)
  
  – Then the update procedure is:
    
    • 1) \( v = v - \eta C \frac{\partial L(x_i, y_i)}{\partial w} \)
    
    • 2) \( s = s(1 - \eta) \)

• **Solution 2:**
  
  – Perform only step 1) for each training example
  
  – Perform step 2) with lower frequency and higher \( \eta \)
Practical Considerations

• **Stopping criteria:**

  How many iterations of SGD?

  – Early stopping with **cross validation**
    • Create validation set
    • Monitor cost function on the validation set
    • Stop when loss stops decreasing
Practical Considerations

• **Stopping criteria:**

  How many iterations of SGD?

  – Early Stopping

    • Extract two disjoint subsamples $A$ and $B$ of training data
    • Train on $A$, stop by validating on $B$
    • Number of epochs is an estimate of $k$
    • Train for $k$ epochs on the full dataset
What about Multiple Classes?

• Idea 1:
  – One against all
  Learn 3 classifiers
    • + vs. \{o, -\}
    • - vs. \{o, +\}
    • o vs. \{+, -\}
  Obtain: \(w_{+}b_{+}, w_{-}b_{-}, w_{o}b_{o}\)
  – Return class c

\[\arg\max_c w_c x + b_c\]
What about Multiple Classes?

• Idea 2:
  – Learn 3 sets of weights simultaneously
  – Want the correct class to have highest margin:

\[ w_{y_i} x_i + b_{y_i} \geq 1 + w_c x_i + b_c \quad \forall c \neq y_i, \forall i \]
Multiclass SVM

• Optimization problem:

\[
\min_{w, b} \frac{1}{2} \sum_c \|w_c\|^2 + C \sum_{i=1}^n \xi_i \\
w_{y_i} x_i + b_{y_i} \geq w_c x_i + b_c + 1 - \xi_i \quad \forall c \neq y_i, \xi_i \geq 0, \forall i
\]

– To obtain parameters \(w_c, b_c\) for each class \(c\), we can use similar techniques as for 2 class SVM

• SVM is widely perceived a very powerful learning algorithm
Reference

• http://www.stanford.edu/class/cs246/slides/13-svm.pdf
• http://www.stanford.edu/class/cs276/handouts/lecture14-SVMs.ppt
• http://i.stanford.edu/~ullman/pub/ch12.pdf
• http://www.svms.org/tutorials/
• http://www.csie.ntu.edu.tw/~cjlin/libsvm/
• Consider building an SVM over the (very little) data set shown in above figure, compute the SVM decision boundary.