CS 5614: (Big) Data Management Systems

B. Aditya Prakash

Lecture #20: Machine Learning 2
Supervised Learning

- **Example: Spam filtering**

<table>
<thead>
<tr>
<th></th>
<th>viagra</th>
<th>learning</th>
<th>the</th>
<th>dating</th>
<th>nigeria</th>
<th>spam?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$y_1 = 1$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$y_2 = -1$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$y_3 = 1$</td>
</tr>
</tbody>
</table>

- **Instance space** $x \in X$ ($|X| = n$ data points)
  - Binary or real-valued feature vector $x$ of word occurrences
  - $d$ features (words + other things, $d \sim 100,000$)

- **Class** $y \in Y$
  - $y$: Spam (+1), Ham (-1)

- **Goal:** Estimate a function $f(x)$ so that $y = f(x)$
More generally: Supervised Learning

- Would like to do prediction: estimate a function \( f(x) \) so that \( y = f(x) \)

- Where \( y \) can be:
  - Real number: Regression
  - Categorical: Classification
  - Complex object:
    - Ranking of items, Parse tree, etc.

- Data is labeled:
  - Have many pairs \( \{(x, y)\} \)
    - \( x \) ... vector of binary, categorical, real valued features
    - \( y \) ... class \( \{+1, -1\} \), or a real number
Supervised Learning

- **Task:** Given data \((X,Y)\) build a model \(f()\) to predict \(Y'\) based on \(X'\)

- **Strategy:** Estimate on .
  
  Hope that the same also works to predict unknown
  
  - The “hope” is called generalization
    
    - **Overfitting:** If \(f(x)\) predicts well \(Y\) but is unable to predict \(Y'\)
    
    - **We want to build a model that generalizes well to unseen data**
      
      - But how can we well on data we have never seen before?!?
Supervised Learning

**Idea:** Pretend we do not know the data/labels we actually do know

- Build the model $f(x)$ on the training data
- See how well $f(x)$ does on the test data
  - If it does well, then apply it also to $X'$

**Refinement: Cross validation**

- Splitting into training/validation set is brutal
- Let’s split our data $(X,Y)$ into 10-folds (buckets)
- Take out 1-fold for validation, train on remaining 9
- Repeat this 10 times, report average performance
Linear models for classification

- **Binary classification:**
  
  \[ f(x) = \begin{cases} 
  +1 & \text{if } w_1 x_1 + w_2 x_2 + \ldots + w_d x_d \geq \theta \\
  -1 & \text{otherwise} 
  \end{cases} \]

- **Input:** Vectors \(x^{(j)}\) and labels \(y^{(j)}\)
  - Vectors \(x^{(j)}\) are real valued where \(\|x\|_2 = 1\)

- **Goal:** Find vector \(w = (w_1, w_2, \ldots, w_d)\)
  - Each \(w_i\) is a real number

Note: Decision boundary is linear

Prakash 2014
Perceptron [Rosenblatt ‘58]

- **(very) Loose motivation:** Neuron
- Inputs are feature values
- Each feature has a weight $w_i$
- **Activation is the sum:**
  - $f(x) = \sum_i w_i x_i = w \cdot x$

- If the $f(x)$ is:
  - **Positive:** Predict +1
  - **Negative:** Predict -1
Perceptron: Estimating $w$

- **Perceptron:** $y' = \text{sign}(w \cdot x)$

- **How to find parameters $w$?**
  - Start with $w_0 = 0$
  - Pick training examples $x(t)$ **one by one** (from disk)
  - Predict class of $x(t)$ using current weights
    - $y' = \text{sign}(w(t) \cdot x(t))$
  - If $y'$ is correct (i.e., $y_t = y'$)
    - No change: $w^{(t+1)} = w^{(t)}$
  - If $y'$ is wrong: adjust $w(t)$
    - $w^{(t+1)} = w^{(t)} + \eta \cdot y^{(t)} \cdot x^{(t)}$
      - $\eta$ is the learning rate parameter
      - $x^{(t)}$ is the $t$-th training example
      - $y^{(t)}$ is true $t$-th class label ($\{+1, -1\}$)

Note that the Perceptron is a conservative algorithm: it ignores samples that it classifies correctly.
**Perceptron: The Good and the Bad**

- **Good: Perceptron convergence theorem:**
  - If there exist a set of weights that are consistent (i.e., the data is linearly separable) the Perceptron learning algorithm will converge

- **Bad: Never converges:**
  - If the data is not separable weights dance around indefinitely

- **Bad: Mediocre generalization:**
  - Finds a “barely” separating solution
Updating the Learning Rate

- Perceptron will oscillate and won’t converge
- So, when to stop learning?
- (1) Slowly decrease the learning rate $\eta$
  - A classic way is to: $\eta = c_1/(t + c_2)$
    - But, we also need to determine constants $c_1$ and $c_2$
- (2) Stop when the training error stops chaining
- (3) Have a small test dataset and stop when the test set error stops decreasing
- (4) Stop when we reached some maximum number of passes over the data
A Training Algorithm for Optimal Margin Classifiers

Bernhard E. Boser*
EECS Department
University of California
Berkeley, CA 94720
boser@eecs.berkeley.edu

Isabelle M. Guyon
AT&T Bell Laboratories
50 Fremont Street, 6th Floor
San Francisco, CA 94105
isabelle@neural.att.com

Vladimir N. Vapnik
AT&T Bell Laboratories
Crawford Corner Road
Holmdel, NJ 07733
vlad@neural.att.com

SUPPORT VECTOR MACHINES
Support Vector Machines

- Want to separate “+” from “−” using a line

Data:
- Training examples:
  - \((x_1, y_1) \ldots (x_n, y_n)\)
- Each example \(i\):
  - \(x_i = (x_i^{(1)}, \ldots, x_i^{(d)})\)
    - \(x_i^{(j)}\) is real valued
  - \(y_i \in \{-1, +1\}\)
- Inner product:
  \[ w \cdot x = \sum_{j=1}^{d} w^{(j)} \cdot x^{(j)} \]

Which is best linear separator (defined by \(w\))?
Largest Margin

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

- Example:
  - We are more sure about the class of A and B than of C
Largest Margin

- **Margin**: Distance of closest example from the decision line/hyperplane

The reason we define margin this way is due to theoretical convenience and existence of generalization error bounds that depend on the value of margin.
Why maximizing $\gamma$ a good idea?

- Remember: Dot product

$$A \cdot B = \|A\| \cdot \|B\| \cdot \cos \theta$$

$\|A\| \cos \theta$

$$\|A\| = \sqrt{\sum_{j=1}^{d} (A(j))^2}$$

Prakash 2014
Why maximizing $\gamma$ a good idea?

- **Dot product**
  \[ A \cdot B = \|A\| \|B\| \cos \theta \]
- What is $w \cdot x_1$, $w \cdot x_2$?

In this case
- $\gamma_1 \approx \|w\|^2$
- $\gamma_2 \approx 2\|w\|^2$

- So, $\gamma$ roughly corresponds to the margin
  - Bigger $\gamma$ bigger the separation
What is the margin?

Let:
- Line \( L \): \( w \cdot x + b = 0 \)
  \[ w^{(1)}x^{(1)} + w^{(2)}x^{(2)} + b = 0 \]
- \( w = (w^{(1)}, w^{(2)}) \)
- Point \( A = (x_A^{(1)}, x_A^{(2)}) \)
- Point \( M \) on a line = \( (x_M^{(1)}, x_M^{(2)}) \)

The margin is defined as:
\[
d(A, L) = |AH|
\]
\[
= |(A-M) \cdot w|
\]
\[
= |(x_A^{(1)} - x_M^{(1)}) w^{(1)} + (x_A^{(2)} - x_M^{(2)}) w^{(2)}|
\]
\[
= x_A^{(1)} w^{(1)} + x_A^{(2)} w^{(2)} + b
\]
\[
= w \cdot A + b
\]

Remember \( x_M^{(1)}w^{(1)} + x_M^{(2)}w^{(2)} = -b \) since \( M \) belongs to line \( L \):

Note we assume \( \|w\|_2 = 1 \)
Largest Margin

- Prediction = $\text{sign}(w \cdot x + b)$
- "Confidence" = $(w \cdot x + b) y_i$
- For $i$-th datapoint:
  \[ y_i = (w \cdot x_i + b)y_i \]
- Want to solve:
  \[ \max_w \min_i \gamma_i \]
- Can rewrite as
  \[ \max_w \gamma \]
  \[ \text{s.t.} \forall i, y_i (w \cdot x_i + b) \geq \gamma \]
Support Vector Machine

- Maximize the margin:
  - Good according to intuition, theory (VC dimension) & practice

\[
\begin{align*}
\max_{w, \gamma} \gamma \\
\text{s.t. } \forall i, y_i (w \cdot x_i + b) \geq \gamma
\end{align*}
\]

- \( \gamma \) is margin ... distance from the separating hyperplane

Prakash 2014
SUPPORT VECTOR MACHINES: DERIVING THE MARGIN
Support Vector Machines

- Separating hyperplane is defined by the support vectors
  - Points on +/- planes from the solution
  - If you knew these points, you could ignore the rest
  - Generally, $d+1$ support vectors (for $d$ dim. data)
Canonic Hyperplane: Problem

Problem:
- Let \((w \cdot x + b) y = \gamma\)
- Then \((2w \cdot x + 2b) y = 2\gamma\)
- Scaling \(w\) increases margin!

Solution:
- Work with normalized \(w\):
  \[\gamma = \left(\frac{w}{\|w\|} \cdot x + b\right) y\]
- Let’s also require **support vectors** \(x_j\)
  to be on the plane defined by: \(w \cdot x_j + b = \pm 1\)

\[\|w\| = \sqrt{\sum_{j=1}^{d} (w(j))^2}\]
Canonical Hyperplane: Solution

- Want to maximize margin $\gamma$!
- What is the relation between $x_1$ and $x_2$?
  - $x_1 = x_2 + 2\gamma \frac{w}{||w||}$
  - We also know:
    - $w \cdot x_1 + b = +1$
    - $w \cdot x_2 + b = -1$
- So:
  - $w \cdot x_1 + b = +1$
  - $w \left( x_2 + 2\gamma \frac{w}{||w||} \right) + b = +1$
  - $w \cdot x_2 + b + 2\gamma \frac{w \cdot w}{||w||} = +1$
  - $\Rightarrow \gamma = \frac{||w||}{w \cdot w} = \frac{1}{||w||}$

Note: $w \cdot w = ||w||^2$
Maximizing the Margin

- We started with
  \[ \max_{w, \gamma} \gamma \]
  
  s.t. \( \forall i, \ y_i (w \cdot x_i + b) \geq \gamma \)
  But \( w \) can be arbitrarily large!

- We normalized and...
  \[ \arg \max \gamma = \arg \max \frac{1}{\|w\|} = \arg \min \|w\| = \arg \min \frac{1}{2} \|w\|^2 \]

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

  s.t. \( \forall i, \ y_i (w \cdot x_i + b) \geq 1 \)

This is called SVM with "hard" constraints
Non-linearly Separable Data

- If data is **not separable** introduce **penalty**:
  \[ \min_w \frac{1}{2} \|w\|^2 + C \cdot (# \text{ number of mistakes}) \]
  \[ s.t. \forall i, y_i(w \cdot x_i + b) \geq 1 \]
  - Minimize \( \|w\|^2 \) plus the number of training mistakes
  - Set \( C \) using cross validation

- **How to penalize mistakes?**
  - All mistakes are not equally bad!
Support Vector Machines

- **Introduce slack variables** \( \xi_i \)

\[
\min_{w, b, \xi_i \geq 0} \frac{1}{2} \|w\|^2 + C \cdot \sum_{i=1}^{n} \xi_i
\]

\[s.t. \forall i, y_i (w \cdot x_i + b) \geq 1 - \xi_i\]

- If point \( x_i \) is on the wrong side of the margin then get penalty \( \xi_i \)

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

$$\min_w \frac{1}{2} \|w\|^2 + C \cdot (#\text{number\ of\ mistakes})$$

s.t. $\forall i, y_i (w \cdot x_i + b) \geq 1$

- **What is the role of slack penalty $C$:**
  - $C=\infty$: Only want to $w, b$ that separate the data
  - $C=0$: Can set $\xi_i$ to anything, then $w=0$ (basically ignores the data)
Support Vector Machines

- SVM in the “natural” form

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

- SVM uses “Hinge Loss”:

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i$$

s.t. \( \forall i, y_i \cdot (w \cdot x_i + b) \geq 1 - \xi_i \)

Hinge loss: \( \max\{0, 1-z\} \)
SUPPORT VECTOR MACHINES: HOW TO COMPUTE THE MARGIN?
SVM: How to estimate $w$?

$$\min_{w,b} \frac{1}{2} w \cdot w + C \cdot \sum_{i=1}^{n} \xi_i$$

$$s.t. \forall i, y_i \cdot (x_i \cdot w + b) \geq 1 - \xi_i$$

- **Want to estimate** \( w \) and \( \xi \)
  - **Standard way**: Use a solver!
    - **Solver**: software for finding solutions to “common” optimization problems

- **Use a quadratic solver**:
  - Minimize quadratic function
  - Subject to linear constraints

- **Problem**: Solvers are inefficient for big data!
SVM: How to estimate $w$?

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

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

- Side note:
  - How to minimize convex functions?
  - Use gradient descent: $\min_{z} g(z)$
  - Iterate: $z_{t+1} \leftarrow z_t - \eta \nabla g(z_t)$

$$\min_{w,b} \quad \frac{1}{2} w \cdot w + C \sum_{i=1}^{n} \xi_i$$

$$s.t. \forall i, y_i \cdot (x_i \cdot w + b) \geq 1 - \xi_i$$
SVM: How to estimate $w$?

- **Want to minimize** $f(w, b)$:
  
  \[
  f(w, b) = \frac{1}{2} \sum_{j=1}^{d} (w^{(j)})^2 + C \sum_{i=1}^{n} \max \left\{ 0, 1 - y_i \left( \sum_{j=1}^{d} w^{(j)} x_i^{(j)} + b \right) \right\}
  \]

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

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

  \[
  \frac{\partial L(x_i, y_i)}{\partial w^{(j)}} = 0 \quad \text{if} \quad y_i (w \cdot x_i + b) \geq 1
  \]

  \[
  = -y_i x_i^{(j)} \quad \text{else}
  \]
SVM: How to estimate \( w \)?

- Gradient descent:

Iterate until convergence:

  - For \( j = 1 \ldots d \)
  
    - Evaluate: \( \nabla f^{(j)} = \frac{\partial f(w, b)}{\partial w^{(j)}} = w^{(j)} + C \sum_{i=1}^{n} \frac{\partial L(x_i, y_i)}{\partial w^{(j)}} \)
  
    - Update: \( w^{(j)} \leftarrow w^{(j)} - \eta \nabla f^{(j)} \)

- Problem:
  
  - Computing \( \nabla f^{(j)} \) takes \( O(n) \) time!
    
      - \( n \) ... size of the training dataset

\( \eta \)...learning rate parameter
\( C \)... regularization parameter
SVM: How to estimate $w$?

- **Stochastic Gradient Descent**
  - Instead of evaluating gradient over all examples evaluate it for each *individual* training example
  
  $$
  \nabla f^{(j)}(x_i) = w^{(j)} + C \cdot \frac{\partial L(x_i, y_i)}{\partial w^{(j)}}
  $$

- **Stochastic gradient descent:**

  **Iterate until convergence:**
  - For $i = 1 \ldots n$
    - For $j = 1 \ldots d$
      - Compute: $\nabla f^{(j)}(x_i)$
      - Update: $w^{(j)} \leftarrow w^{(j)} - \eta \nabla f^{(j)}(x_i)$
SUPPORT VECTOR MACHINES: EXAMPLE
Example: Text categorization

- **Example by Leon Bottou:**
  - Reuters RCV1 document corpus
    - Predict a category of a document
      - One *vs.* the rest classification
  - \( n = 781,000 \) training examples (documents)
  - 23,000 test examples
  - \( d = 50,000 \) features
    - One feature per word
    - Remove stop-words
    - Remove low frequency words
Example: Text categorization

- **Questions:**
  - (1) Is SGD successful at minimizing \( f(w,b) \)?
  - (2) How quickly does SGD find the min of \( f(w,b) \)?
  - (3) What is the error on a test set?

<table>
<thead>
<tr>
<th></th>
<th>Training time</th>
<th>Value of ( f(w,b) )</th>
<th>Test error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard SVM</td>
<td>23,642 secs</td>
<td>0.2275</td>
<td>6.02%</td>
</tr>
<tr>
<td>&quot;Fast SVM&quot;</td>
<td>66 secs</td>
<td>0.2278</td>
<td>6.03%</td>
</tr>
<tr>
<td>SGD SVM</td>
<td>1.4 secs</td>
<td>0.2275</td>
<td>6.02%</td>
</tr>
</tbody>
</table>

(1) SGD-SVM is successful at minimizing the value of \( f(w,b) \)
(2) SGD-SVM is super fast
(3) SGD-SVM test set error is comparable
Optimization “Accuracy”

Optimization quality: $| f(w,b) - f(w^{opt},b^{opt}) |$

For optimizing $f(w,b)$ within reasonable quality
SGD-SVM is super fast

Prakash 2014
SGD vs. Batch Conjugate Gradient

- **SGD** on full dataset vs. **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) CG update a few times

Theory says: Gradient descent converges in linear time. Conjugate gradient converges in $\kappa$ condition number
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: $x_i = [0,0,0,1,0,0,0,5,0,0,0,0,0,0,0,...]$
    - But represent $x_i$ as a sparse vector $x_i = [(4,1), (9,5), ...]$
  - Can we do the SGD update more efficiently?
    
    $$w \leftarrow w - \eta C \frac{\partial L(x_i, y_i)}{\partial w}$$
    
    cheap: $x_i$ is sparse and so few coordinates $j$ of $w$ will be updated
    
    $$w \leftarrow w(1 - \eta)$$
    
    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 vector \( v \)
  - Then the update procedure is:
    - \( (1) \quad v = v - \eta C \frac{\partial L(x_i, y_i)}{\partial w} \)
    - \( (2) \quad s = s(1 - \eta) \)

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

Two step update procedure:

1. \( w \leftarrow w - \eta C \frac{\partial L(x_i, y_i)}{\partial w} \)
2. \( w \leftarrow w(1 - \eta) \)
Practical Considerations

- **Stopping criteria:**

  How many iterations of SGD?
  
  - Early stopping with cross validation
    - Create a validation set
    - Monitor cost function on the validation set
    - Stop when loss stops decreasing
  
  - 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:
  \[ w_+ b_+, w_- b_-, w_o b_o \]
  - **How to classify?**
  - Return class \( c \)
  \[ \arg \max_c w_c x + b_c \]
Learn 1 classifier: Multiclass SVM

- **Idea 2:** Learn 3 sets of weights simultaneously!
  - For each class $c$ estimate $w_c, b_c$
  - Want the correct class to have highest margin:
    $$w_{y_i} x_i + b_{y_i} \geq 1 + w_c x_i + b_c \quad \forall c \neq y_i \text{ , } \forall i$$
Multiclass SVM

- Optimization problem:

\[
\min_{w,b} \frac{1}{2} \sum_c \|w_c\|^2 + C \sum_{i=1}^n \xi_i \\
\text{subject to } \begin{array}{l}
wy_i \cdot x_i + b_y_i \geq wc \cdot x_i + bc + 1 - \xi_i \\
\forall c \neq y_i, \forall i \\
\xi_i \geq 0, \forall i
\end{array}
\]

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

- SVM is widely perceived a very powerful learning algorithm