Support Vector Machine
Gradient Descent & Multi-Class Classification

M. R. Hasan
Readings

- Alpaydin: 13.3
- Murphy: 14.5.2.1, 14.5.2.4, 14.5.3
- Geron: chapter 5, Appendix C
What We Will Cover

- SVM: Gradient Descent
- Hinge Loss Cost Function
- Online SVM
- An analysis of SVM GD Loss Function
- Sparse Solution
- Multi-class classification
- How do we choose the best SVM model for classification?
SVM Classifier: Gradient Descent
SVM: Algorithms

• So far, we have discussed the constrained optimization algorithm to learn the SVM discriminant function.
• We can also use the Gradient Descent algorithm to learn the SVM discriminant.
• Why (and when) do we care about the GD algorithm for SVM?
SVM Classifier: Gradient Descent

• Recall in Logistic Regression we used Gradient Descent (GD) algorithm to solve the classification problem.

• The benefit of the GD approach is that by using the Stochastic Gradient Descent (SGD) we could design online classifier.

• The SGD based online classifiers enable us to learn incrementally, typically as new instances arrive.

• Is it possible to design an online SVM classifier?
SVM Classifier: Gradient Descent

- We will see how it is possible to design online SVM classifiers.
- For applying the GD/SGD algorithm, we need to define an unconstrained cost function for the soft margin case.

\[
\min_{\vec{w}, b, \xi_i} \frac{1}{2} \vec{w}^T \vec{w} + C \sum_{i=1}^{N} \xi_i \\
\text{subject to } y_i (\vec{w}^T \vec{x}_i + b) \geq 1 - \xi_i \quad \forall i \\
\xi_i \geq 0
\]
SVM Classifier: Gradient Descent

• Let’s derive the unconstrained cost/loss (optimization) function for the soft margin case.
• To do this, we need to express the two constraints on $\xi_i$ as a single constraint, and then consume it in the Lagrangian.

$$\min_{\mathbf{w}, b, \xi_i} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{N} \xi_i$$

subject to
$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0$$
SVM Classifier: Gradient Descent

• Consider the following decision rule for the soft margin case.

• It implies that we define **error/loss** if the instance is on the wrong side or if the margin is less than 1.

• This deviation is denoted by $\xi_i$.

$$y_i(\mathbf{w}^T \hat{x}_i + b) \geq 1 - \xi_i \quad \forall i$$
SVM Classifier: Gradient Descent

- In the SVM QP, the loss is measured by $\xi_i$:

$$\min_{\overline{w},b,\xi_i} \frac{1}{2} \overline{w}^T \overline{w} + C \sum_{i=1}^{N} \xi_i$$

subject to $y_i(\overline{w}^T \overline{x}_i + b) \geq 1 - \xi_i \quad \forall i$

$\xi_i \geq 0$

The range of the loss measure $\xi_i$ is:

$$\xi_i \geq 1 - y_i(\overline{w}^T \overline{x}_i + b)$$

$$\xi_i \geq 0$$

Hence, the loss $\xi_i$ can be expressed using a \textbf{max (.) function}:

$$\xi_i \geq \max \{0, 1 - y_i(\overline{w}^T \overline{x}_i + b)\}$$
SVM Classifier: Gradient Descent

- Observe that $\xi_i$ ranges between 0 & $(1 - y_i(w^T x_i + b))$.

- It is called the **hinge loss**.

- It is so-called because of its **similarity with the hinge loss function**.

\[
\xi_i \geq \max \{0, 1 - y_i(w^T x_i + b)\}
\]

\[
y_i(w^T x_i + b) \geq 1 - \xi_i \quad \forall i
\]
SVM Classifier: Gradient Descent

- The hinge loss function is defined as $\max(0, 1 - t)$.
- It is equal to 0 when $t \geq 1$ [i.e., if $y_i(w^T x_i + b \geq 1$, loss is 0].
- Its derivative (slope) is equal to $-1$ if $t < 1$ and 0 if $t > 1$.
- Thus no change in loss if $y_i(w^T x_i + b \geq 1)$, i.e., for correct classification.
- It is not differentiable at $t = 1$.

The function is so named because it looks like a door hinge!

\[ \xi_i \geq \max \{0, 1 - y_i(w^T \tilde{x}_i + b )\} \]
The range of the loss measure $\xi_i$ is:

$$ \xi_i \geq 1 - y_i(\vec{w}^T \vec{x}_i + b) \quad \forall i $$

$$ \xi_i \geq 0 $$

Hence, the hinge loss:

$$ \xi_i \geq \max \{0, 1 - y_i(\vec{w}^T \vec{x}_i + b)\} $$

$$ \min_{\vec{w}, b} \frac{1}{2} \vec{w}^T \vec{w} + C \sum_{i=1}^{N} h(y_i(\vec{w}^T \vec{x}_i + b)) $$

Regularization

Loss (hinge)
SVM Classifier: Gradient Descent

• It simply means that during classification, the **loss can be zero** (best case scenario).
• Or in case of deviation, loss will deviate from zero **in proportion to the amount of deviation** [which is, \(1 - y_i(w^T x_i + b)\)].

\[
\begin{align*}
\xi_i &\geq 1 - y_i(w^T \hat{x}_i + b) \\
\xi_i &\geq 0
\end{align*}
\]

Hence, the hinge loss:

\[
\xi_i \geq \max\{0, 1 - y_i(w^T \hat{x}_i + b)\}
\]
SVM Classifier: Gradient Descent

• To get a sense of the **amount of loss**, project a data point $x_i$ on the $w$ vector and **compute the length of the projection**.

• Let’s denote it by **score of $x_i = s_i$**

• This score measure **how far $x_i$ is** from the decision boundary.

\[
s_i \equiv \text{score by SVM of $i$th data point} = \hat{w}^T \hat{x}_i + b
\]

Then, the **hinge loss**:

\[
\xi_i \geq \max\{0, 1 - y_i s_i\}
\]

\[
\xi_i \geq \max \{0, 1 - y_i (\hat{w}^T \hat{x}_i + b )\}
\]
Finally, using the concept of **hinge loss** $h(.)$, we can write the constrained SVM QP as an **unconstrained** optimization:

$$
\min_{\mathbf{w}, b, \xi} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{N} \xi_i \\
\text{subject to } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i \newline
\xi_i \geq 0
$$

It's an **unconstrained** QP where

The hinge loss:

$$
h(z) = \max\{0, 1 - z\}
$$

$$
\xi_i \geq \max\{0, 1 - y_i s_i\}
$$
SVM Classifier: Gradient Descent

- This loss function is used in the GD/SGD algorithms as cost function to find optimal w and b.
- Recognize the regularization and the loss term in SVM QP:

\[
\min_{\vec{w}, b} \frac{1}{2} \vec{w}^T \vec{w} + C \sum_{i=1}^{N} h( y_i (\vec{w}^T \vec{x}_i + b ))
\]

- Hinge loss function:
  \[ h(z) = \max \{0, 1 - z\} \]
SVM Classifier: Gradient Descent

• Observe that the cost function is **convex** (sum of convex + Hinge).

• Thus GD/SGD could find the the **global minimum** by finding the optimal $w$ and $b$. 

\[
\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{N} h( y_i(w^T \hat{x_i} + b) )
\]

- Regularization
- Loss (hinge)
SVM Classifier: Gradient Descent

• The **first term** in the cost function will push the model to have a small weight vector $w$, leading to a **larger margin**.

• The second sum computes the **total of all margin violations**.

\[
\min_{\omega,b} \frac{1}{2} \omega^T \omega + C \sum_{i=1}^{N} h( y_i (\omega^T \tilde{x}_i + b ))
\]

\[
h(z) = \max\{0, 1 - z\}
\]

\[
\xi_i \geq \max\{0, 1 - y_i s_i\}
\]

**The hinge loss:**
SVM Classifier: Gradient Descent

- An **instance’s margin violation** is equal to 0 if it is located off the street and on the correct side.
- Or else it is proportional to the distance to the correct side of the street.
- Minimizing this term ensures that the model makes the margin violations as small and as few as possible.

The hinge loss:

\[
\min_{\vec{w},b} \frac{1}{2} \vec{w}^T \vec{w} + C \sum_{i=1}^{N} h(y_i(\vec{w}^T \vec{x}_i + b))
\]

\[
h(z) = \max\{0, 1 - z\}
\]

\[
\xi_i \geq \max\{0, 1 - y_i s_i\}
\]
SVM: Stochastic Gradient Descent

• See the following two Jupyter notebooks for the **SGD-based SVM** to solve binary classification problems for *linear* and *nonlinear* data.

  
Online SVM Classifier
Online SVM Classifier

- By using this cost function we can perform **online learning** with SVM.
- Online learning means learning incrementally, typically as new instances arrive.
- We design Online SVM classifier by using a **linear SVM classifier** that employs Gradient Descent to minimize the following cost function.

\[
\min_{\mathbf{w}, b} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{N} h(y_i (\mathbf{w}^T \mathbf{x}_i + b))
\]

- The hinge loss:

\[
h(z) = \max \{0, 1 - z\}
\]

\[
\xi_i \geq \max \{0, 1 - y_i s_i\}
\]
Online SVM Classifier

• However, note that this method **converges very slowly** as compared to the Kernel SVMs (QP based methods for solving the dual problem).
• It is also possible to implement **online kernel SVMs**.
• For large-scale nonlinear problems, it is more efficient to use **neural networks** instead.

\[
\min_{\vec{w}, b} \frac{1}{2} \vec{w}^T \vec{w} + C \sum_{i=1}^{N} h(y_i(\vec{w}^T \vec{x}_i + b))
\]

\[
h(z) = \max \{0, 1 - z\}
\]

The hinge loss:

\[
\xi_i \geq \max\{0, 1 - y_i s_i\}
\]
SVM GD Loss Function: Through the Lens of Linear & Logistic Regression
Previously we saw **loss functions** (as well as regularization terms) in linear regression and logistic regression.

- Linear regression: **squared loss function**
- Logistic regression: **cross-entropy**
- Does the SVM loss function **differ from** the previous loss functions?

\[
\min_{\mathbf{w}, b} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{N} h(y_i(\mathbf{w}^T \mathbf{x}_i + b))
\]

- **Regularization**
- **Loss (hinge)**
SVM: Hinge Loss Function

- Let’s compare the **Logistic Regression** cross-entropy loss or negative log-loss function with the SVM Hinge loss function.
- For convenience, consider only a single data point \( \bar{x} \) and its probability to belong to class 1 (\( y = 1 \)).

**Logistic Regression** Negative Log-Loss (cross-entropy) Function would be:

\[
NLL(\vec{w}) = -[y \log \sigma(\vec{w}^T \cdot \bar{x}) + (1 - y) \log (1 - \sigma(\vec{w}^T \cdot \bar{x}))]
\]

For \( y = 1 \) & isolating the bias term: \( b = w_0 \cdot x_0 \)

**SVM**: Loss function is defined by the Hinge Loss

\[
\xi \geq \max\{0, 1 - y(\vec{w}^T \cdot \bar{x} + b)\}
\]

\[
NLL(\vec{w}) = -\log \left( \frac{1}{1 + e^{-(\vec{w}^T \cdot \bar{x} + b)}} \right)
\]

\[
NLL(\vec{w}) = \log \left( 1 + e^{-(\vec{w}^T \cdot \bar{x} + b)} \right)
\]
Plot of the Logistic Regression loss function and the SVM Hinge loss function for a single data point $\vec{x}$ to belong to class 1 ($y = 1$).

**SVM:** Loss function **sharply goes to 0** when $w^T x + b = 1$

**Logistic Regression:** Loss function **smoothly approaches** to 0 when $w^T x + b > 1$; becomes 0 only at infinity.

**Why does this matter?**

Log-loss: $NLL(\vec{w}) = \log \left( 1 + e^{-\left(\vec{w}^T \cdot \vec{x} + b\right)} \right)$

Hinge Loss: 
$$\xi \geq \max\{0, 1 - y(\vec{w}^T \cdot \vec{x} + b)\}$$
SVM: Hinge Loss Function

- **SVM**: loss function \textbf{gradient becomes 0} at \( w^T x + b > 1 \)
- **Logistic Regression**: loss function \textbf{gradient is positive} at \( w^T x + b > 1 \)

Thus, in Logistic Regression the GD algorithm \textbf{keeps on optimizing} \( w \) even after \( \hat{x} \) is correctly classified as belonging to \( y = 1 \)
SVM: Hinge Loss Function

• The fact that SVM loss function gradient becomes 0 at $w^T x + b > 1$ results into \textbf{sparse solution}.

• In a sparse solution, the weight vector $w$ depends on a \textbf{sparse number of training data points} (i.e., support vectors).

In Logistic Regression $w$ depends on every data point.

As a consequence slight change in the data point will influence $w$ (i.e., decision boundary)
SVM: Hinge Loss Function

• This phenomenon of sparse solution in SVM has another consequence.
• For the data point $\vec{x}$ that belongs to $y = 1$, SVM assigns a score of 1; and then, it’s done!
• For all positive separable data points score is the same!
• We cannot differentiate our confidence on the score.

However, in Logistic Regression, due to the positive gradient, the algorithm still tries to optimize the solution and data points receive varying scores depending on their classes.
SVM: Hinge Loss Function

- Comparison of **hinge loss** with 0/1 loss (perceptron), squared error (linear regression), and cross-entropy (negative log-likelihood of Logistic Regression) for \( y = 1 \).

\[ w^T x_i + b \]
SVM: Benefit of the Sparse Solution
SVM: Sparse Solution

- Because of the sparse solutions, SVM is also known as Sparse Kernel Machine.
- What is the benefit of the sparsity in SVM?
  - It enables us to perform **feature selection**!
  - In SVM the concept of feature selection is slightly different.
SVM: Sparse Solution

• Feature selection in SVM is equivalent to selecting a subset of the training examples.
• It helps to reduce overfitting and computational cost.
• Recall that in Generalized Linear Models (GLM) such as Linear & Logistic Regression, we used feature selection very differently to overcome the overfitting problem.

• We employed Bayesian technique which essentially introduced regularization in the model loss function.
SVM: Sparse Solution

• In the GLM, while minimizing regularized loss function, we were able to **filter out features** that didn’t effectively explain the variance in the target.

• By feature selection the model **encourages the weight vector \( w \) to be sparse**, i.e., to have lots of zeros.

• Interestingly SVM does feature selection (by creating sparsity) **without using a Bayesian technique**.

• It modifies the likelihood term, which is **rather unnatural** from a Bayesian point of view.

• Nevertheless, the effect is similar.
Multi-Class Classification using SVM
SVM for Multi-Class Classification

- There are **two techniques** to perform multi-class classification using SVM.
  - One-versus-All (OvA) (aka One-versus-the-Rest, OvR)
  - One-versus-One (OvO)
OvA: create 3 binary classifiers

OvO: create $\frac{3(3-1)}{2} = 3$ binary classifiers

3 classes: $K = 3$
Scikit-Learn SVC class implements the **OvO technique** for multi-class classification. See this Jupyter notebook for an example:


OvO: create \( \frac{3(3-1)}{2} = 3 \) binary classifiers
SVM Multi-Class Classification: OvO

• The main advantage of OvO is that we don’t need the entire dataset for training.
• Each classifier only needs to be trained on the part of the training set for the two classes that it must distinguish.
• Since the SVM dual algorithm scales poorly with the size of the training set, OvO is the preferred technique.
• It is faster to train many classifiers on small training sets than training few classifiers on large training sets.
SVM for Multi-Class Classification

- However, unlike Logistic Regression (OvA or OvO), comparing the output of multiple binary classifiers in SVM is *not straight forward*.

- Because the SVM outputs are *not on a calibrated scale*.
  - Unlike logistic regression, they *don’t lie between 0 ~ 1*.
  - Hence, outputs are *hard to compare to each other*. 
Both OvA & OvO: Creates regions of input space which are ambiguously labeled.
Her life story—as a researcher, Nobel Prize winner, and public policy thinker—connects the CRISPR tale to some larger historical threads, including the role of women in science. Her work also illustrates, as Leonardo da Vinci’s did, that the key to innovation is connecting a curiosity about basic science to the practical work of devising tools that can be applied to our lives—moving discoveries from lab bench to bedside.

By telling her story, I hope to give an up-close look at how science works. What actually happens in a lab? To what extent do discoveries depend on individual genius, and to what extent has teamwork become more critical? Has the competition for prizes and patents undermined collaboration?

Most of all, I want to convey the importance of basic science, meaning quests that are curiosity-driven rather than application-oriented. Curiosity-driven research into the wonders of nature plants the seeds, sometimes in unpredictable ways, for later innovations. Research about surface-state physics eventually led to the transistor and microchip. Likewise, studies of an astonishing method that bacteria use to fight off viruses eventually led to a gene-editing tool and techniques that humans can use in their own struggle against viruses.

It is a story filled with the biggest of questions, from the origins of life to the future of the human race. And it begins with a sixth-grade girl who loved searching for “sleeping grass” and other fascinating phenomena amid the lava rocks of Hawaii, coming home from school one day and finding on her bed a detective tale about the people who discovered what they proclaimed to be, with only a little exaggeration, “the secret of life.”
SVM: How do We Choose the Best Model?
SVM: Choosing the Best Model

• When solving a classification problem, the pertinent question is which SVM model should we use from below:
  - Linear Model (solves the primal problem by QP solver or uses GD based algorithms using Hinge loss)
  - Kernelized SVM (solves the dual problem by QP solver)

• Also for the Kernelized SVM, which kernel function should we use (polynomial, Gaussian RBF, string, etc.).

**LinearSVC**: $O(Nd)$
Based on the liblinear library, implements linear SVM

**SVC**: $O(N^2d \sim N^3d)$
Based on the libsvm library, supports the kernel trick
SVM: Choosing the Best Model

• Below we provide a **rough guideline** to choose optimal method.

• For detail discussion see the notebook **“Support Vector Machine: Linearly Separable Data”**:
  

---

<table>
<thead>
<tr>
<th>Non-Complex Data</th>
<th>Complex Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>N is very large</td>
<td>N: Small ~ Medium</td>
</tr>
<tr>
<td>d: small (N &gt; d)</td>
<td>d: small (N &gt; d)</td>
</tr>
<tr>
<td>d is large relative to N (d ≥ N)</td>
<td></td>
</tr>
</tbody>
</table>

**Linear Model:**
- Logistic Regression
- SVM without kernel or GD based

**Kernelized SVM with the Gaussian RBF Kernel**

---

**LinearSVC:** $O(Nd)$
Based on the liblinear library, implements linear SVM

**SVC:** $O(N^2d \sim N^3d)$
Based on the libsvm library, supports the kernel trick
Beyond SVM: Very Large Complex Data

• What if we have very large complex (highly nonlinear) data set?
• Does Kernel SVM work well on this type of problem?
• No!
• Then, what do we do?

Use nonlinear models such as Artificial Neural Network

<table>
<thead>
<tr>
<th>Non-Complex Data</th>
<th>Complex Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>N is very large</td>
<td>N: Small ~ Medium</td>
</tr>
<tr>
<td>d: small (N &gt; d)</td>
<td>d: small</td>
</tr>
<tr>
<td>d is large relative to N</td>
<td>(N &gt; d)</td>
</tr>
<tr>
<td>N (d ≥ N)</td>
<td></td>
</tr>
</tbody>
</table>

Linear Model:
Logistic Regression
SVM without kernel or GD based

Kernelized SVM with the Gaussian RBF Kernel