Support Vector Machines

- Decision surface is a hyperplane (line in 2D) in feature space (similar to the Perceptron)
- Arguably, the most important recent discovery in machine learning
- In a nutshell:
  - map the data to a predetermined very high-dimensional space via a kernel function
  - Find the hyperplane that maximizes the margin between the two classes
  - If data are not separable find the hyperplane that maximizes the margin and minimizes the (a weighted average of the) misclassifications

Bias-variance trade-off
Support Vector Machines

Three main ideas:
1. Define what an optimal hyperplane is (in a way that can be identified in a computationally efficient way): \textit{maximize margin}
2. Extend the above definition for non-linearly separable problems: \textit{have a penalty term for misclassifications}
3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: \textit{reformulate problem so that data is mapped implicitly to this space (kernel)}
Support Vector Machines

Three main ideas:

1. Define what an optimal hyperplane is (in a computationally efficient way): maximize margin
2. Extend the above definition for non-linearly separable problems: have a penalty term for misclassifications
3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space
Which Separating Hyperplane to Use?
Maximizing the Margin

IDEA 1: Select the separating hyperplane that maximizes the margin!
Support Vectors

Var₁

Support Vectors

Margin Width

Var₂
Setting Up the Optimization Problem

The width of the margin is:
\[
\frac{2|k|}{\|w\|}
\]

So, the problem is:
\[
\max \frac{2|k|}{\|w\|}
\]

s.t. \( (w \cdot x + b) \geq k \), \( \forall x \) of class 1

\( (w \cdot x + b) \leq -k \), \( \forall x \) of class 2
Setting Up the Optimization Problem

There is a scale and unit for data so that $k=1$. Then problem becomes:

$$\max \frac{2}{\|w\|}$$

subject to

$$(w \cdot x + b) \geq 1, \forall x \text{ of class 1}$$

and

$$(w \cdot x + b) \leq -1, \forall x \text{ of class 2}$$
If class 1 corresponds to 1 and class 2 corresponds to -1, we can rewrite

\[(w \cdot x_i + b) \geq 1, \ \forall x_i \text{ with } y_i = 1\]
\[(w \cdot x_i + b) \leq -1, \ \forall x_i \text{ with } y_i = -1\]

as

\[y_i(w \cdot x_i + b) \geq 1, \ \forall x_i\]

So the problem becomes:

\[
\begin{align*}
\text{max} \ & \frac{2}{\|w\|} \quad \text{or} \quad \min \ & \frac{1}{2} \|w\|^2 \\
\text{s.t.} \ & y_i(w \cdot x_i + b) \geq 1, \ \forall x_i \\
\text{s.t.} \ & y_i(w \cdot x_i + b) \geq 1, \ \forall x_i
\end{align*}
\]
Linear, Hard-Margin SVM Formulation

Find $w,b$ that solves

$$\min \frac{1}{2} \|w\|^2$$

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

- Problem is convex so, there is a unique global minimum value (when feasible)
- There is also a unique minimizer, i.e. weight and $b$ value that provides the minimum
- Non-solvable if the data is not linearly separable
- Quadratic Programming
  - Very efficient computationally with modern constraint optimization engines (handles thousands of constraints and training instances).
Support Vector Machines

Three main ideas:
1. Define what an optimal hyperplane is (in a way that can be identified in a computationally efficient way): maximize margin
2. Extend the above definition for non-linearly separable problems: have a penalty term for misclassifications
3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space
Support Vector Machines

Three main ideas:

1. Define what an optimal hyperplane is (in way that can be identified in a computationally efficient way): maximize margin

2. Extend the above definition for non-linearly separable problems: have a penalty term for misclassifications

3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space
Non-Linearly Separable Data

Introduce slack variables $\xi_i$

Allow some instances to fall within the margin, but penalize them.
Formulating the Optimization Problem

Constraint becomes:
\[ y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \forall \mathbf{x}_i \]
\[ \xi_i \geq 0 \]

Objective function penalizes for misclassified instances and those within the margin
\[ \min \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_i \xi_i \]

C trades-off margin width and misclassifications
Linear, Soft-Margin SVMs

\[
\min \frac{1}{2} \| w \|^2 + C \sum_i \xi_i \quad \text{subject to} \quad y_i (w \cdot x_i + b) \geq 1 - \xi_i, \quad \forall x_i, \quad \xi_i \geq 0
\]

- Algorithm tries to maintain \( \xi_i \) to zero while maximizing margin
- Notice: algorithm does not minimize the number of misclassifications (NP-complete problem) but the sum of distances from the margin hyperplanes
- Other formulations use \( \xi_i^2 \) instead
- As \( C \to \infty \), we get closer to the hard-margin solution
Robustness of Soft vs Hard Margin SVMs

\[ \xi_i = \pm b \cdot \mathbf{w} \cdot \mathbf{x} \]

Soft Margin SVN

Hard Margin SVN
Soft vs Hard Margin SVM

- Soft-Margin always have a solution
- Soft-Margin is more robust to outliers
  - Smoother surfaces (in the non-linear case)
- Hard-Margin does not require to guess the cost parameter (requires no parameters at all)
Support Vector Machines

Three main ideas:

1. Define what an optimal hyperplane is (in a computationally efficient way): **maximize margin**

2. Extend the above definition for non-linearly separable problems: **have a penalty term for misclassifications**

3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: **reformulate problem so that data is mapped implicitly to this space**
Support Vector Machines

Three main ideas:

1. Define what an optimal hyperplane is (in a way that can be identified in a computationally efficient way): maximize margin
2. Extend the above definition for non-linearly separable problems: have a penalty term for misclassifications
3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space
Disadvantages of Linear Decision Surfaces
Advantages of Non-Linear Surfaces
Linear Classifiers in High-Dimensional Spaces

Find function $\Phi(x)$ to map to a different space
Mapping Data to a High-Dimensional Space

• Find function $\Phi(x)$ to map to a different space, then SVM formulation becomes:

$$\min \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \quad s.t. \quad y_i (w \cdot \Phi(x) + b) \geq 1 - \xi_i, \forall x_i$$

• Data appear as $\Phi(x)$, weights $w$ are now weights in the new space

• Explicit mapping expensive if $\Phi(x)$ is very high dimensional

• Solving the problem without explicitly mapping the data is desirable
The Dual of the SVM Formulation

Original SVM formulation
- \( n \) inequality constraints
- \( n \) positivity constraints
- \( n \) number of \( \xi \) variables

The (Wolfe) dual of this problem
- one equality constraint
- \( n \) positivity constraints
- \( n \) number of \( \alpha \) variables (Lagrange multipliers)
- Objective function more complicated

NOTICE: Data only appear as \( \Phi(x_i) \cdot \Phi(x_j) \)

\[
\begin{align*}
\text{min} & \quad \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \\
\text{s.t.} & \quad y_i (w \cdot \Phi(x) + b) \geq 1 - \xi_i, \forall x_i \\
& \quad \xi_i \geq 0
\end{align*}
\]

\[
\begin{align*}
\text{min} & \quad \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (\Phi(x_i) \cdot \Phi(x_j)) - \sum_i \alpha_i \\
\text{s.t.} & \quad C \geq \alpha_i \geq 0, \forall x_i \\
& \quad \sum_i \alpha_i y_i = 0
\end{align*}
\]
The Kernel Trick

- $\Phi(x_i) \cdot \Phi(x_j)$: means, map data into new space, then take the inner product of the new vectors
- We can find a function such that: $K(x_i \cdot x_j) = \Phi(x_i) \cdot \Phi(x_j)$, i.e., the image of the inner product of the data is the inner product of the images of the data
- Then, we do not need to explicitly map the data into the high-dimensional space to solve the optimization problem (for training)
- How do we classify without explicitly mapping the new instances? Turns out

$$\text{sgn}(wx + b) = \text{sgn}(\sum_i \alpha_i y_i K(x_i, x) + b)$$

where $b$ solves $\alpha_j (y_j \sum_i \alpha_i y_i K(x_i, x_j) + b - 1) = 0$, for any $j$ with $\alpha_j \neq 0$
Examples of Kernels

Assume we measure two quantities, e.g. expression level of genes *TrkC* and *SonicHedgehog* (*SH*) and we use the mapping:

\[ \Phi : \langle x_{TrkC}, x_{SH} \rangle \rightarrow \{x_{TrkC}^2, x_{SH}^2, \sqrt{2}x_{TrkC}x_{SH}, x_{TrkC}, x_{SH}, 1\} \]

Consider the function:

\[ K(x \cdot z) = (x \cdot z + 1)^2 \]

We can verify that:

\[ \Phi(x) \cdot \Phi(z) = \]

\[ x_{TrkC}^2 z_{TrkC}^2 + x_{SH}^2 z_{SH}^2 + 2x_{TrkC}x_{SH} z_{TrkC}z_{SH} + x_{TrkC} z_{TrkC} + x_{SH} z_{SH} + 1 = \]

\[ = (x_{TrkC} z_{TrkC} + x_{SH} z_{SH} + 1)^2 = (x \cdot z + 1)^2 = K(x \cdot z) \]
Polynomial and Gaussian Kernels

\[ K(x \cdot z) = (x \cdot z + 1)^p \]

- is called the polynomial kernel of degree \( p \).
- For \( p=2 \), if we measure 7,000 genes using the kernel once means calculating a summation product with 7,000 terms then taking the square of this number.
- Mapping explicitly to the high-dimensional space means calculating approximately 50,000,000 new features for both training instances, then taking the inner product of that (another 50,000,000 terms to sum).
- In general, using the Kernel trick provides huge computational savings over explicit mapping!
- Another commonly used Kernel is the Gaussian (maps to a dimensional space with number of dimensions equal to the number of training cases):

\[ K(x \cdot z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right) \]
The Mercer Condition

- Is there a mapping $\Phi(x)$ for any symmetric function $K(x,z)$? No
- The SVM dual formulation requires calculation $K(x_i, x_j)$ for each pair of training instances. The array $G_{ij} = K(x_i, x_j)$ is called the Gram matrix
- There is a feature space $\Phi(x)$ when the Kernel is such that $G$ is always semi-positive definite (Mercer condition)
Support Vector Machines

Three main ideas:

1. Define what an optimal hyperplane is (in way that can be identified in a computationally efficient way): maximize margin
2. Extend the above definition for non-linearly separable problems: have a penalty term for misclassifications
3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space
Other Types of Kernel Methods

- SVMs that perform regression
- SVMs that perform clustering
- $\nu$-Support Vector Machines: maximize margin while bounding the number of margin errors
- Leave One Out Machines: minimize the bound of the leave-one-out error
- SVM formulations that take into consideration difference in cost of misclassification for the different classes
- Kernels suitable for sequences of strings, or other specialized kernels
Variable Selection with SVMs

- Recursive Feature Elimination
  - Train a linear SVM
  - Remove the variables with the lowest weights (those variables affect classification the least), e.g., remove the lowest 50% of variables
  - Retrain the SVM with remaining variables and repeat until classification is reduced
- Very successful
- Other formulations exist where minimizing the number of variables is folded into the optimization problem
- Similar algorithm exist for non-linear SVMs
- Some of the best and most efficient variable selection methods
Comparison with Neural Networks

Neural Networks
- Hidden Layers map to lower dimensional spaces
- Search space has multiple local minima
- Training is expensive
- Classification extremely efficient
- Requires number of hidden units and layers
- Very good accuracy in typical domains

SVMs
- Kernel maps to a very-high dimensional space
- Search space has a unique minimum
- Training is extremely efficient
- Classification extremely efficient
- Kernel and cost the two parameters to select
- Very good accuracy in typical domains
- Extremely robust
Why do SVMs Generalize?

- Even though they map to a very high-dimensional space
  - They have a very strong bias in that space
  - The solution has to be a linear combination of the training instances
- Large theory on Structural Risk Minimization providing bounds on the error of an SVM
  - Typically the error bounds too loose to be of practical use
MultiClass SVMs

- One-versus-all
  - Train $n$ binary classifiers, one for each class against all other classes.
  - Predicted class is the class of the most confident classifier
- One-versus-one
  - Train $n(n-1)/2$ classifiers, each discriminating between a pair of classes
  - Several strategies for selecting the final classification based on the output of the binary SVMs
- Truly MultiClass SVMs
  - Generalize the SVM formulation to multiple categories

Conclusions

- SVMs express learning as a mathematical program taking advantage of the rich theory in optimization.
- SVM uses the kernel trick to map indirectly to extremely high dimensional spaces.
- SVMs extremely successful, robust, efficient, and versatile while there are good theoretical indications as to why they generalize well.
Suggested Further Reading

- http://www.kernel-machines.org/tutorial.html
- Hastie, Tibshirani, Friedman, The Elements of Statistical Learning, Springel 2001