CSI5180. Machine Learning for Bioinformatics Applications

Kernel methods in bioinformatics

by

Marcel Turcotte

Version **November 6, 2019**

Preamble

Preamble

Kernel methods in bioinformatics

In this lecture, we continue our exploration of **kernel methods** in bioinformatics. After an informal presentation of **support vector machines**, we now more formally investigate kernel methods. Still, the aim is to give you the intuition behind these methods. Specifically, you should understand how the data is implicitly embedded into a higher-dimensional space.

General objective :

- **Discuss** applications of kernel methods in bioinformatics

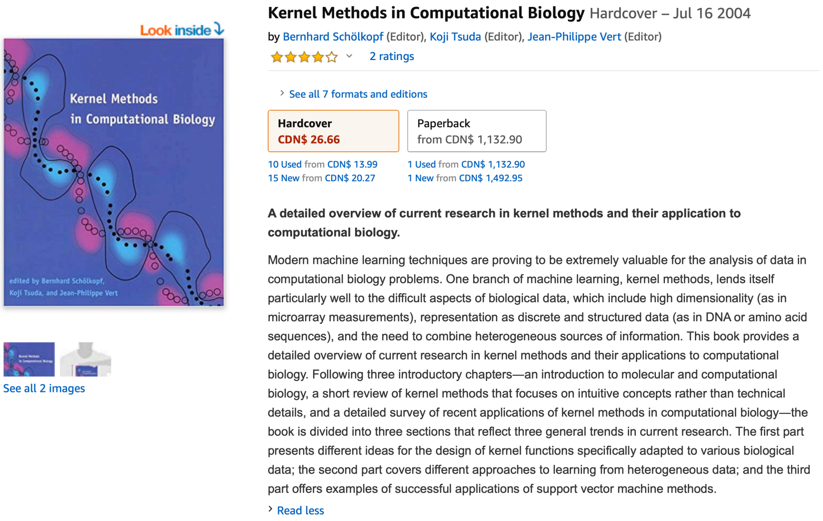
Learning objectives

- **Discuss** applications of kernel methods in bioinformatics

Reading:

- Berhhard Schölkopf, Koji Tsuda, and Jean-Philippe Vert (eds.), Kernel methods in computational biology, MIT Press, 2004. §2.

Kernel methods in computational biology


Look inside
Kernel Methods in Computational Biology
edited by Bernhard Schölkopf, Koji Tsuda, and Jean-Philippe Vert
See all 2 images

Kernel Methods in Computational Biology Hardcover – Jul 16 2004

by Bernhard Schölkopf (Editor), Koji Tsuda (Editor), Jean-Philippe Vert (Editor)

★★★★☆ 2 ratings

See all 7 formats and editions

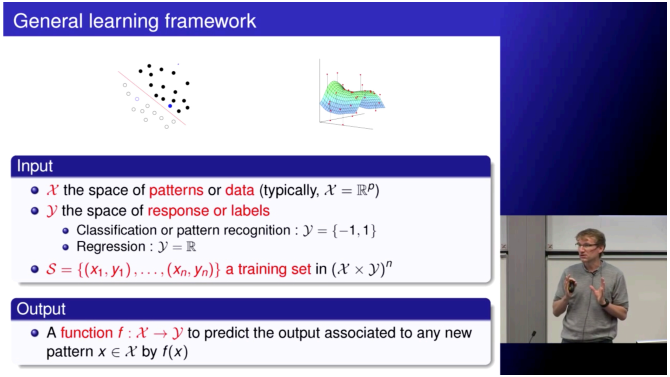
Hardcover	Paperback
CDN$ 26.66	from CDN$ 1,132.90
10 Used from CDN$ 13.99	1 Used from CDN$ 1,132.90
15 New from CDN$ 20.27	1 New from CDN$ 1,492.95

A detailed overview of current research in kernel methods and their application to computational biology.

Modern machine learning techniques are proving to be extremely valuable for the analysis of data in computational biology problems. One branch of machine learning, kernel methods, lends itself particularly well to the difficult aspects of biological data, which include high dimensionality (as in microarray measurements), representation as discrete and structured data (as in DNA or amino acid sequences), and the need to combine heterogeneous sources of information. This book provides a detailed overview of current research in kernel methods and their applications to computational biology. Following three introductory chapters—an introduction to molecular and computational biology, a short review of kernel methods that focuses on intuitive concepts rather than technical details, and a detailed survey of recent applications of kernel methods in computational biology—the book is divided into three sections that reflect three general trends in current research. The first part presents different ideas for the design of kernel functions specifically adapted to various biological data; the second part covers different approaches to learning from heterogeneous data; and the third part offers examples of successful applications of support vector machine methods.

Read less

Kernel methods in computational biology



General learning framework

Input

- $\mathcal{X}$ the space of patterns or data (typically, $\mathcal{X} = \mathbb{R}^p$)
- $\mathcal{Y}$ the space of response or labels
  - Classification or pattern recognition : $\mathcal{Y} = \{-1, 1\}$
  - Regression : $\mathcal{Y} = \mathbb{R}$
- $\mathcal{S} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ a training set in $(\mathcal{X} \times \mathcal{Y})^n$

Output

- A function $f : \mathcal{X} \to \mathcal{Y}$ to predict the output associated to any new pattern $x \in \mathcal{X}$ by $f(x)$

- `https://www.youtube.com/watch?v=svXc382Y3aw` (Part 1 - 1 hour 23 minutes)
- `https://www.youtube.com/watch?v=9QRVG1wB-ds` (Part 2 - 1 hour 31 minutes)
- `https://www.youtube.com/watch?v=KPpFc2OASIo` (Part 3 - 1 hour 38 minutes)

Plan

Introduction

Summary

- Just like **hidden Markov models** (**HMM**), **support vector machines** (**SVM**) have a strong theoretical foundation. **Kernel methods** are grounded into **statistical learning theory**.

Summary

- Just like **hidden Markov models** (**HMM**), **support vector machines** (**SVM**) have a strong theoretical foundation. **Kernel methods** are grounded into **statistical learning theory**.
- **Support vector machines** (**SVM**) are one of the most popular **kernel methods**.

Summary

- Just like **hidden Markov models** (**HMM**), **support vector machines** (**SVM**) have a strong theoretical foundation. **Kernel methods** are grounded into **statistical learning theory**.
- **Support vector machines** (**SVM**) are one of the most popular **kernel methods**.
- Excellent **performance** (accuracy).

Summary

- Just like **hidden Markov models** (**HMM**), **support vector machines** (**SVM**) have a strong theoretical foundation. **Kernel methods** are grounded into **statistical learning theory**.
- **Support vector machines** (**SVM**) are one of the most popular **kernel methods**.
- Excellent **performance** (accuracy).
- Handling **high dimensionality**.

Summary

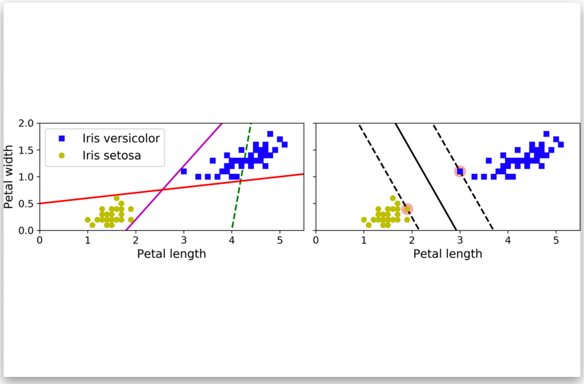
- Just like **hidden Markov models** (**HMM**), **support vector machines** (**SVM**) have a strong theoretical foundation. **Kernel methods** are grounded into **statistical learning theory**.
- **Support vector machines** (**SVM**) are one of the most popular **kernel methods**.
- Excellent **performance** (accuracy).
- Handling **high dimensionality**.
- Kernels for **strings**, **graphs**, and more.

Summary

- Just like **hidden Markov models** (**HMM**), **support vector machines** (**SVM**) have a strong theoretical foundation. **Kernel methods** are grounded into **statistical learning theory**.
- **Support vector machines** (**SVM**) are one of the most popular **kernel methods**.
- Excellent **performance** (accuracy).
- Handling **high dimensionality**.
- Kernels for **strings**, **graphs**, and more.
- **Combining** kernels for **data fusion**.

Summary

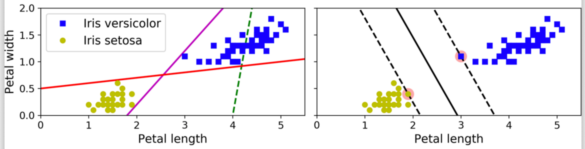
- The **support vectors** are the examples **closest** to the **separating hyperplane**.



Source: [19] Figure 5.1

Summary

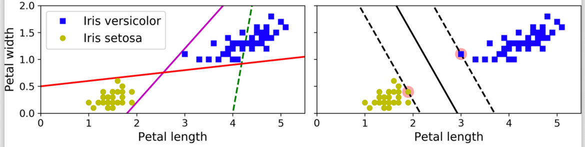
- The **support vectors** are the examples **closest** to the **separating hyperplane**.
- The **margin** is the distance between the **separating hyperplane** (decision boundary) and the **support vectors**.



Source: [19] Figure 5.1

Summary

- The **support vectors** are the examples **closest** to the **separating hyperplane**.
- The **margin** is the distance between the **separating hyperplane** (decision boundary) and the **support vectors**.
- **Problem:** of all possible **separating hyperplanes** find the one with the **largest margin**.



Source: [19] Figure 5.1

Summary

- Just like **logistic regression**, **support vector machines** are learning the parameters of a **linear decision boundary** separating the data in **two classes**.

Summary

- Just like **logistic regression**, **support vector machines** are learning the parameters of a **linear decision boundary** separating the data in **two classes**.
- However, **SVM** algorithms also rely on the concept of **maximum margin hyperplane** as a mechanism to lower generalization errors.

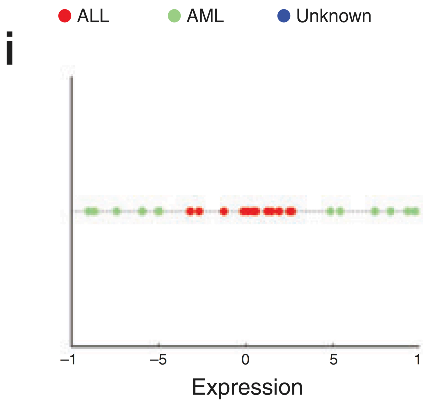
Summary

- Just like **logistic regression**, **support vector machines** are learning the parameters of a **linear decision boundary** separating the data in **two classes**.
- However, **SVM** algorithms also rely on the concept of **maximum margin hyperplane** as a mechanism to lower generalization errors.
- In order to handle a **small number of classification errors**, the algorithms introduce the concept of **soft margin**.

Summary

- Just like **logistic regression**, **support vector machines** are learning the parameters of a **linear decision boundary** separating the data in **two classes**.
- However, **SVM** algorithms also rely on the concept of **maximum margin hyperplane** as a mechanism to lower generalization errors.
- In order to handle a **small number of classification errors**, the algorithms introduce the concept of **soft margin**.
- Finally, because the data is **not always linearly separable**, these algorithms project the data to **higher dimensions** using a **kernel function**.

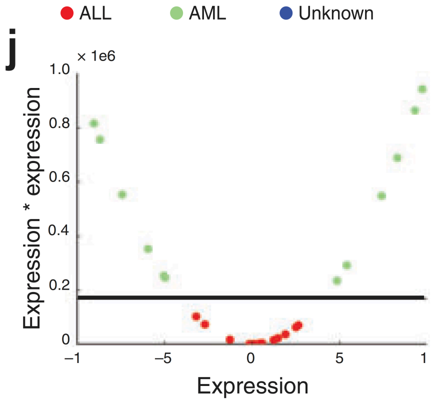
Summary



Source: [20] Figure 1i

- **No** single point can separate the two classes!

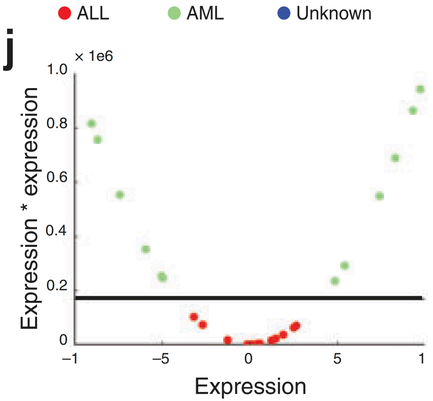
Summary



Source: [20] Figure 1j

- Adding a **new dimension** to our data.

Summary



Source: [20] Figure 1j

- Adding a **new dimension** to our data.
- Here, simply taking the square values of our feature.

Support vector machines

Support vector machines (SVM)

- Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik, A training algorithm for optimal margin classifiers, *COLT, ACM*, pp. 144152, 1992.

Notation

- Let $\mathcal{X}$ be a set of objects (the **universe**).

Notation

- Let $\mathcal{X}$ be a set of objects (the **universe**).
    - **Gene expression** profiles.

Notation

- Let $\mathcal{X}$ be a set of objects (the **universe**).
  - **Gene expression** profiles.
  - DNA, RNA or protein **sequences**.

Notation

- Let $\mathcal{X}$ be a set of objects (the **universe**).
  - **Gene expression** profiles.
  - DNA, RNA or protein **sequences**.
  - **Connectivity** of molecules, phylogenetic **trees**, RNA **structure**, molecular **pathways**, etc.

Notation

- Let $\mathcal{X}$ be a set of objects (the **universe**).
  - **Gene expression** profiles.
  - DNA, RNA or protein **sequences**.
  - **Connectivity** of molecules, phylogenetic **trees**, RNA **structure**, molecular **pathways**, etc.
- Let $\mathcal{S} = \{x_1, x_2, \ldots, x_N\}$ be a **data set**, each $x_i \in \mathcal{X}$.

Notation

- Let $\mathcal{X}$ be a set of objects (the **universe**).
    - **Gene expression** profiles.
    - DNA, RNA or protein **sequences**.
    - **Connectivity** of molecules, phylogenetic **trees**, RNA **structure**, molecular **pathways**, etc.
- Let $\mathcal{S} = \{x_1, x_2, \ldots, x_N\}$ be a **data set**, each $x_i \in \mathcal{X}$.
- Let $\mathcal{Y} = \{y_1, y_2, \ldots, y_N\}$ be the **labels** associated with each object.

Problem

- **Learn** some function $f : \mathcal{X} \rightarrow \mathcal{Y}$ from $\mathcal{S}$.

Problem

- **Learn** some function $f : \mathcal{X} \to \mathcal{Y}$ from $\mathcal{S}$.
- To **predict** the label of an $x \in \mathcal{X}$, evaluate $f(x)$.

Binary classification problem

- Herein, we consider the **binary classification problem**.

Binary classification problem

- Herein, we consider the **binary classification problem**.
- Let use the label **1** for **positive examples** and -**1** for **negative examples**.

Binary classification problem

- Herein, we consider the **binary classification problem**.
- Let use the label **1** for **positive examples** and -**1** for **negative examples**.
- Consequently, each y_i belongs to the set $\{-1, 1\}$.

Support Vector Machine (SVM)

- Assuming that $\mathcal{X} = \mathbb{R}^D$ (for now).

Support Vector Machine (SVM)

- Assuming that $\mathcal{X} = \mathbb{R}^D$ (for now).
- An **SVM** makes predictions using a function f of the following form:

$$f(x) = w^T x + b$$

where $w \in \mathbb{R}^D$ and $b \in \mathbb{R}$.

Support Vector Machine (SVM)

- Assuming that $\mathcal{X} = \mathbb{R}^D$ (for now).
- An **SVM** makes predictions using a function f of the following form:

$$f(x) = w^T x + b$$

  where $w \in \mathbb{R}^D$ and $b \in \mathbb{R}$.
- We want to assign the label **+1** to points $x \in \mathcal{X}$ such that $f(x) \geq 0$, and **-1** to points $x \in \mathcal{X}$ such that $f(x) < 0$.

Support Vector Machine (SVM)

- Assuming a **candidate function** $f(x)$ (it is our best current estimate).

Support Vector Machine (SVM)

- Assuming a **candidate function** $f(x)$ (it is our best current estimate).
- An example is **correctly classified** if

$$y_i f(x_i) \geq 0$$

Support Vector Machine (SVM)

- Assuming a **candidate function** $f(x)$ (it is our best current estimate).
- An example is **correctly classified** if

$$y_i f(x_i) \geq 0$$

  - If $y_i = 1$ and $f(x_i) = 1$, then $y_i f(x_i) \geq 0$

Support Vector Machine (SVM)

- Assuming a **candidate function** $f(x)$ (it is our best current estimate).
- An example is **correctly classified** if

$$y_i f(x_i) \geq 0$$

  - If $y_i = 1$ and $f(x_i) = 1$, then $y_i f(x_i) \geq 0$
  - If $y_i = -1$ and $f(x_i) = -1$, then $y_i f(x_i) \geq 0$

Support Vector Machine (SVM)

- Assuming a **candidate function** $f(x)$ (it is our best current estimate).
- An example is **correctly classified** if

$$y_i f(x_i) \geq 0$$

  - If $y_i = 1$ and $f(x_i) = 1$, then $y_i f(x_i) \geq 0$
  - If $y_i = -1$ and $f(x_i) = -1$, then $y_i f(x_i) \geq 0$
  - For the other two cases, the example is misclassified.

Support Vector Machine (SVM)

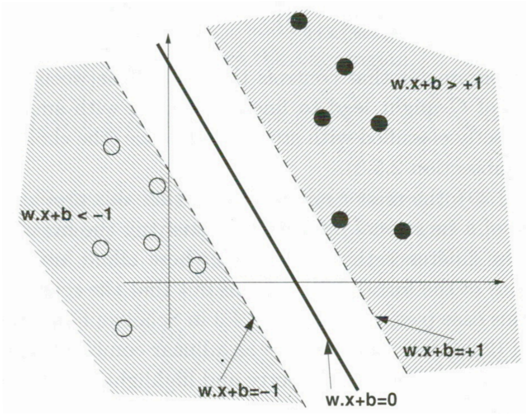
- A principled, termed **empirical risk minimization** suggests selecting the function $f(x)$ that makes the **fewer number of classification errors** on the training set $\mathcal{S}$.

Support Vector Machine (SVM)

- A principled, termed **empirical risk minimization** suggests selecting the function $f(x)$ that makes the **fewer number of classification errors** on the training set $\mathcal{S}$.
- As we have seen, when the data is **linearly separable**, there could be infinitely many such hyperplanes (decision boundaries).

Support Vector Machine (SVM)

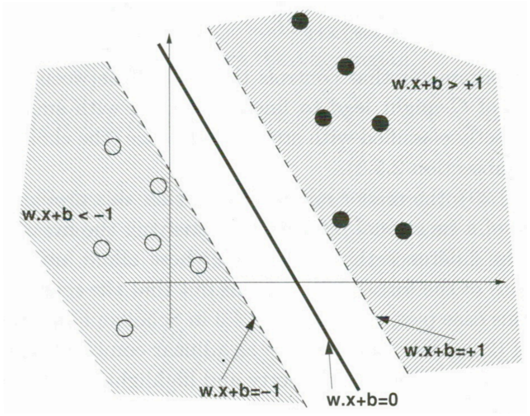
- Two **half-spaces**:



Source [1] Figure 2.9

Support Vector Machine (SVM)

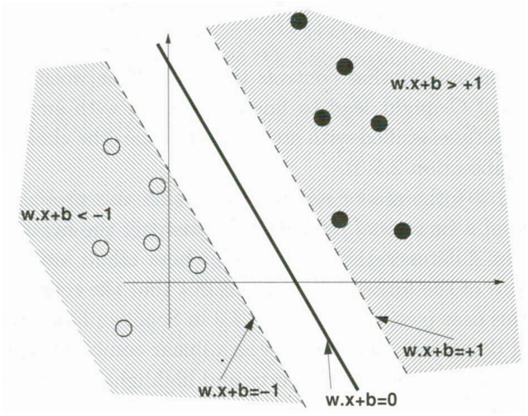
- Two **half-spaces**:
  - $h^+ = \{x : f(x) \geq 1\}$



Source [1] Figure 2.9

Support Vector Machine (SVM)

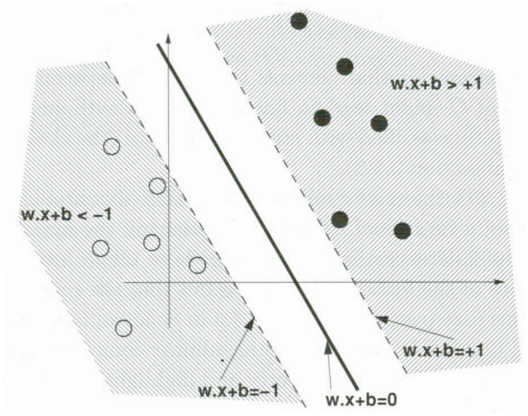
- Two **half-spaces**:
  - $h^+ = \{x : f(x) \geq 1\}$
  - $h^- = \{x : f(x) \leq -1\}$



Source [1] Figure 2.9

Support Vector Machine (SVM)

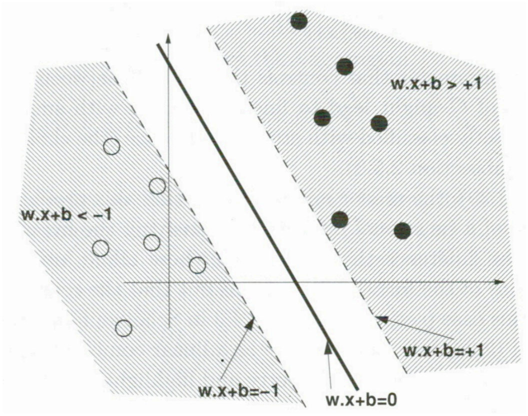
- Two **half-spaces**:
  - $h^+ = \{x : f(x) \geq 1\}$
  - $h^- = \{x : f(x) \leq -1\}$
  - The **distance** between the two half-spaces is called the **margin**.



Source [1] Figure 2.9

Support Vector Machine (SVM)

- Two **half-spaces**:
  - $h^+ = \{x : f(x) \geq 1\}$
  - $h^- = \{x : f(x) \leq -1\}$
  - The **distance** between the two half-spaces is called the **margin**.
  - It can be shown that the **margin** is exactly $\frac{2}{||w||}$

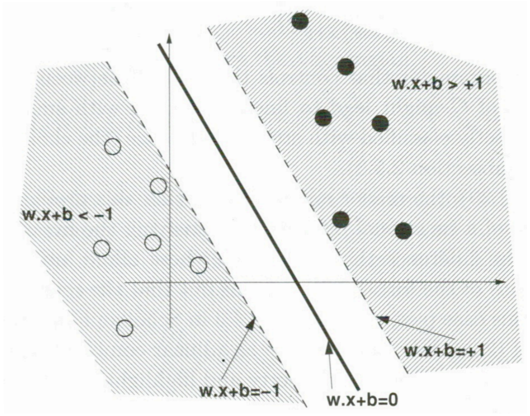


Source [1] Figure 2.9

Large margin

- **Optimization**: maximize $\frac{2}{||w||}$ subject to

$$y_i(w^T x + b) \geq 1 \text{ for } i = 1, \ldots, N$$



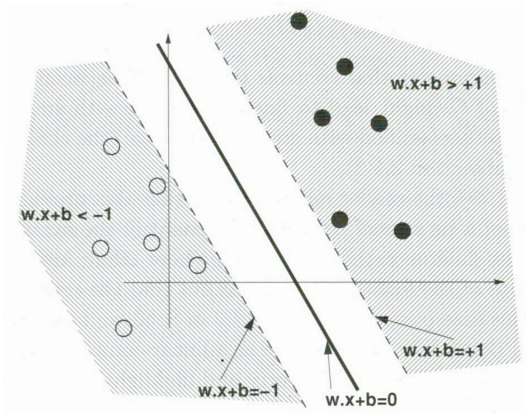
Source [1] Figure 2.9

Large margin

- **Optimization**: maximize $\frac{2}{||w||}$ subject to

$$y_i(w^T x + b) \geq 1 \text{ for } i = 1, \ldots, N$$

- There will be a solution only if the data set is **linearly separable**.

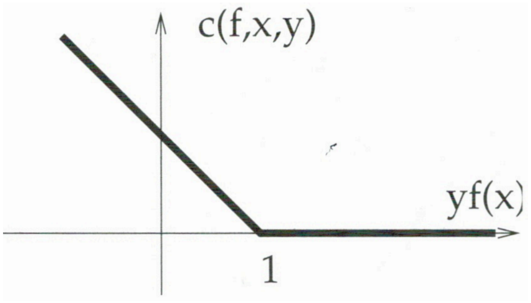


Source [1] Figure 2.9

Hinge loss function

- To accommodate for some examples to be **misclassified** a **continuous hinge loss function** is used.

$$c(f,x,y) = \max(0, 1 - yf(x))$$



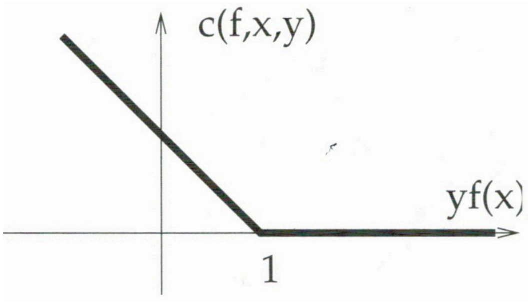
Source [1] Figure 2.9

Hinge loss function

- To accommodate for some examples to be **misclassified** a **continuous hinge loss function** is used.

$$c(f, x, y) = \max(0, 1 - yf(x))$$

- If $yf(x) \geq 1$, then $c(f, x, y) = 0$.



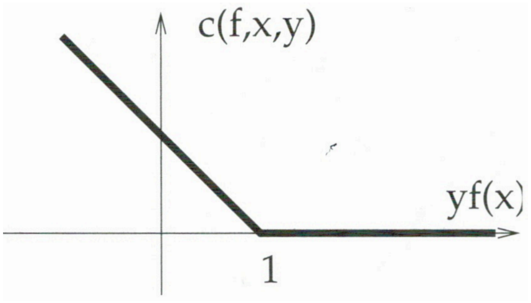
Source [1] Figure 2.9

Hinge loss function

- To accommodate for some examples to be **misclassified** a **continuous hinge loss function** is used.

$$c(f, x, y) = \max(0, 1 - yf(x))$$

- If $yf(x) \geq 1$, then $c(f, x, y) = 0$.
- If $0 \leq yf(x) \leq 1$, correctly classified, low confidence ([0,1]),



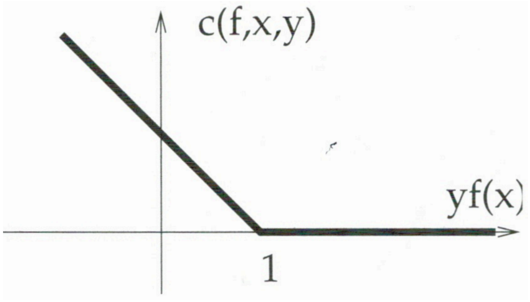
Source [1] Figure 2.9

Hinge loss function

- To accommodate for some examples to be **misclassified** a **continuous hinge loss function** is used.

$$c(f, x, y) = \max(0, 1 - yf(x))$$

- If $yf(x) \geq 1$, then $c(f, x, y) = 0$.
- If $0 \leq yf(x) \leq 1$, correctly classified, low confidence ([0,1]),
  - If $y = 1$, . . .



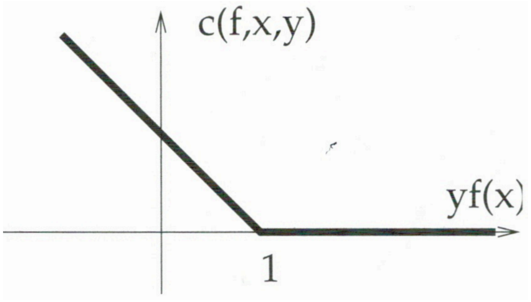
Source [1] Figure 2.9

Hinge loss function

- To accommodate for some examples to be **misclassified** a **continuous hinge loss function** is used.

$$c(f, x, y) = \max(0, 1 - yf(x))$$

- If $yf(x) \geq 1$, then $c(f, x, y) = 0$.
- If $0 \leq yf(x) \leq 1$, correctly classified, low confidence ([0,1]),
  - If $y = 1, \ldots$
  - If $y = -1, \ldots$



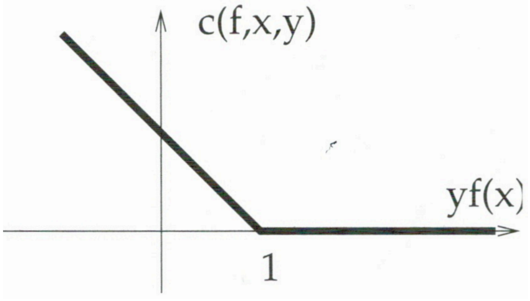
Source [1] Figure 2.9

Hinge loss function

- To accommodate for some examples to be **misclassified** a **continuous hinge loss function** is used.

$$c(f, x, y) = \max(0, 1 - yf(x))$$

- If $yf(x) \geq 1$, then $c(f, x, y) = 0$.
- If $0 \leq yf(x) \leq 1$, correctly classified, low confidence ([0,1]),
  - If $y = 1, \ldots$
  - If $y = -1, \ldots$
- if $yf(x) \leq 0$, the example is **misclassified** and the **cost** is $1 - yf(x)$.



Source [1] Figure 2.9

Support Vector Machine

- **Optimization**

$$\operatorname{argmin}_{f(x)=w^T x+b}\ \frac{1}{2}||w|| + C\sum_{i=1}^{N} c(f, x_i, y_i)$$

where C is a user-defined parameter controlling the tradeoff between having a **large margin** and **classification errors**.

Kernel representation

Representation - the universe

- Let $\mathcal{X}$ be a set of objects (the **universe**).

Representation - the universe

- Let $\mathcal{X}$ be a set of objects (the **universe**).
  - **Gene expression** profiles.

Representation - the universe

- Let $\mathcal{X}$ be a set of objects (the **universe**).
  - **Gene expression** profiles.
  - DNA, RNA or protein **sequences**.

Representation - the universe

- Let $\mathcal{X}$ be a set of objects (the **universe**).
    - **Gene expression** profiles.
    - DNA, RNA or protein **sequences**.
    - **Connectivity** of molecules, phylogenetic **trees**, RNA **structure**, molecular **pathways**, etc.

Representation - traditional learners

- Let $\mathcal{S} = \{x_1, x_2, \ldots, x_n\}$ be a **data set**, the objects to be analyzed, each $x_i \in \mathcal{X}$.

Representation - traditional learners

- Let $\mathcal{S} = \{x_1, x_2, \ldots, x_n\}$ be a **data set**, the objects to be analyzed, each $x_i \in \mathcal{X}$.
- In order to apply a machine learning algorithm, we need a **representation** for each object, $\phi(x) \in \mathcal{F}, \forall x \in \mathcal{X}$.

Representation - traditional learners

- Let $\mathcal{S} = \{x_1, x_2, \ldots, x_n\}$ be a **data set**, the objects to be analyzed, each $x_i \in \mathcal{X}$.
- In order to apply a machine learning algorithm, we need a **representation** for each object, $\phi(x) \in \mathcal{F}, \forall x \in \mathcal{X}$.
- $\phi(\mathcal{S}) = \{\phi(x_1), \phi(x_2), \ldots, \phi(x_n)\}$.

Representation - traditional learners

- Let $\mathcal{S} = \{x_1, x_2, \ldots, x_n\}$ be a **data set**, the objects to be analyzed, each $x_i \in \mathcal{X}$.
- In order to apply a machine learning algorithm, we need a **representation** for each object, $\phi(x) \in \mathcal{F}, \forall x \in \mathcal{X}$.
- $\phi(\mathcal{S}) = \{\phi(x_1), \phi(x_2), \ldots, \phi(x_n)\}$.
- With a traditional learning algorithm, such as **logistic regression**, the examples are represented as real-valued vectors.

Representation - traditional learners

- Let $\mathcal{S} = \{x_1, x_2, \ldots, x_n\}$ be a **data set**, the objects to be analyzed, each $x_i \in \mathcal{X}$.
- In order to apply a machine learning algorithm, we need a **representation** for each object, $\phi(x) \in \mathcal{F}, \forall x \in \mathcal{X}$.
- $\phi(\mathcal{S}) = \{\phi(x_1), \phi(x_2), \ldots, \phi(x_n)\}$.
- With a traditional learning algorithm, such as **logistic regression**, the examples are represented as real-valued vectors.
  - **Vectors.** For a given application, each object $x_i \in \mathcal{X}$ could represent the level of expression of D genes for the i^{th} sample , in this case $\mathcal{F} = \mathbb{R}^D$.

Representation - traditional learners

- Let $\mathcal{S} = \{x_1, x_2, \dots, x_n\}$ be a **data set**, the objects to be analyzed, each $x_i \in \mathcal{X}$.
- In order to apply a machine learning algorithm, we need a **representation** for each object, $\phi(x) \in \mathcal{F}, \forall x \in \mathcal{X}$.
- $\phi(\mathcal{S}) = \{\phi(x_1), \phi(x_2), \dots, \phi(x_n)\}$.
- With a traditional learning algorithm, such as **logistic regression**, the examples are represented as real-valued vectors.
    - **Vectors.** For a given application, each object $x_i \in \mathcal{X}$ could represent the level of expression of D genes for the i^{th} sample , in this case $\mathcal{F} = \mathbb{R}^D$.
- $\phi : \mathcal{X} \rightarrow \mathcal{F}$

Representation - traditional learners

- Let $\mathcal{S} = \{x_1, x_2, \ldots, x_n\}$ be a **data set**, the objects to be analyzed, each $x_i \in \mathcal{X}$.
- In order to apply a machine learning algorithm, we need a **representation** for each object, $\phi(x) \in \mathcal{F}, \forall x \in \mathcal{X}$.
- $\phi(\mathcal{S}) = \{\phi(x_1), \phi(x_2), \ldots, \phi(x_n)\}$.
- With a traditional learning algorithm, such as **logistic regression**, the examples are represented as real-valued vectors.
  - **Vectors.** For a given application, each object $x_i \in \mathcal{X}$ could represent the level of expression of D genes for the i^{th} sample , in this case $\mathcal{F} = \mathbb{R}^D$.
- $\phi : \mathcal{X} \rightarrow \mathcal{F}$
  - Each object $x \in \mathcal{X}$ is represented by $\phi(x) \in \mathcal{F}$.

Representation - kernel methods

- Let $\mathcal{S} = \{x_1, x_2, \ldots, x_n\}$ be a **data set**, the objects to be analyzed, each $x_i \in \mathcal{X}$.

Representation - kernel methods

- Let $\mathcal{S} = \{x_1, x_2, \ldots, x_n\}$ be a **data set**, the objects to be analyzed, each $x_i \in \mathcal{X}$.
- **Kernel methods** represent the data as set of **pairwise comparisons**.

Representation - kernel methods

- Let $\mathcal{S} = \{x_1, x_2, \ldots, x_n\}$ be a **data set**, the objects to be analyzed, each $x_i \in \mathcal{X}$.
- **Kernel methods** represent the data as set of **pairwise comparisons**.
    - $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

Representation - kernel methods

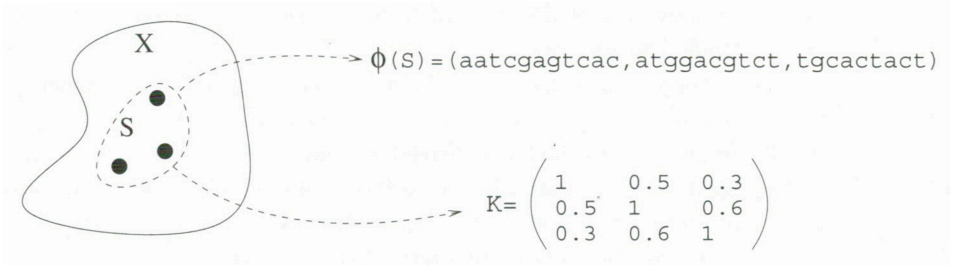
- Let $\mathcal{S} = \{x_1, x_2, \ldots, x_n\}$ be a **data set**, the objects to be analyzed, each $x_i \in \mathcal{X}$.
- **Kernel methods** represent the data as set of **pairwise comparisons**.
  - $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$.
  - Consequently, the **data set**, $\mathcal{S}$, is represented by a $n \times n$ matrix of pairwise comparisons $k_{i,j} = k(x_i, x_j)$.

Representation - kernel methods

- Let $\mathcal{S} = \{x_1, x_2, \ldots, x_n\}$ be a **data set**, the objects to be analyzed, each $x_i \in \mathcal{X}$.
- **Kernel methods** represent the data as set of **pairwise comparisons**.
    - $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.
    - Consequently, the **data set**, $\mathcal{S}$, is represented by a $n \times n$ matrix of pairwise comparisons $k_{i,j} = k(x_i, x_j)$.

Representation - kernel methods

- Let $\mathcal{S} = \{x_1, x_2, \ldots, x_n\}$ be a **data set**, the objects to be analyzed, each $x_i \in \mathcal{X}$.
- **Kernel methods** represent the data as set of **pairwise comparisons**.
  - $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.
  - Consequently, the **data set**, $\mathcal{S}$, is represented by a $n \times n$ matrix of pairwise comparisons $k_{i,j} = k(x_i, x_j)$.


X
S
φ(S) = (aatcgagtcac,atggacgtct,tgcactact)
$K = \begin{pmatrix} 1 & 0.5 & 0.3 \\ 0.5 & 1 & 0.6 \\ 0.3 & 0.6 & 1 \end{pmatrix}$

Source: [1] Figure 2.1

Representation - kernel methods

- The **data representation** as a square matrix is **independent** of the nature of the objects!

Representation - kernel methods

- The **data representation** as a square matrix is **independent** of the nature of the objects!
    - Algorithms are **modular**. The same algorithm can work **strings** or **graphs**, as long as a **function** k is defined for those objects (DNA, RNA, or protein sequences, graph connectivity, phylogenetic trees, RNA structures, molecular pathways, . . .).

Representation - kernel methods

- This size of the input matrix depends on the **number of examples**, $n \times n$, **not** the complexity of the objects.

Representation - kernel methods

- This size of the input matrix depends on the **number of examples**, $n \times n$, **not** the complexity of the objects.
  - An analysis involving 100 samples requires a 100×100 input matrix.

Representation - kernel methods

- This size of the input matrix depends on the **number of examples**, $n \times n$, **not** the complexity of the objects.
  - An analysis involving 100 samples requires a 100×100 input matrix.
  - Even if each sample comprises **thousands** of gene expression levels.

Representation - kernel methods

- This size of the input matrix depends on the **number of examples**, $n \times n$, **not** the complexity of the objects.
  - An analysis involving 100 samples requires a 100×100 input matrix.
  - Even if each sample comprises **thousands** of gene expression levels.
  - This is **computationally** attractive.

Representation - kernel methods

- Many algorithms, including **logistic regression** and **deep learning**, require a real-valued representation:

$$\phi : \mathcal{X} \to \mathbb{R}^D$$

Representation - kernel methods

- Many algorithms, including **logistic regression** and **deep learning**, require a real-valued representation:

$$\phi : \mathcal{X} \to \mathbb{R}^D$$

  - This is not always practical.

Representation - kernel methods

- Many algorithms, including **logistic regression** and **deep learning**, require a real-valued representation:

$$\phi : \mathcal{X} \to \mathbb{R}^D$$

  - This is not always practical.
    - **Macromolecular sequences** having different lengths.

Representation - kernel methods

- Many algorithms, including **logistic regression** and **deep learning**, require a real-valued representation:

$$\phi : \mathcal{X} \to \mathbb{R}^D$$

  - This is not always practical.
    - **Macromolecular sequences** having different lengths.
    - **Phylogenetic trees**.

Representation - kernel methods

- Many algorithms, including **logistic regression** and **deep learning**, require a real-valued representation:

$$\phi : \mathcal{X} \to \mathbb{R}^D$$

  - This is not always practical.
    - **Macromolecular sequences** having different lengths.
    - **Phylogenetic trees**.
  - Pairwise comparison methods are often readily available.

Representation - kernel methods

- Many algorithms, including **logistic regression** and **deep learning**, require a real-valued representation:

$$\phi : \mathcal{X} \to \mathbb{R}^D$$

  - This is not always practical.
    - **Macromolecular sequences** having different lengths.
    - **Phylogenetic trees**.
  - Pairwise comparison methods are often readily available.
    - **Pairwise sequence comparison** (alignment)

Representation - kernel methods

- Many algorithms, including **logistic regression** and **deep learning**, require a real-valued representation:

$$\phi : \mathcal{X} \to \mathbb{R}^D$$

  - This is not always practical.
    - **Macromolecular sequences** having different lengths.
    - **Phylogenetic trees**.
  - Pairwise comparison methods are often readily available.
    - **Pairwise sequence comparison** (alignment)
    - **Pairwise tree comparison** (Robinson-Foulds, RFL, etc.)

Representation - kernel methods

- Many algorithms, including **logistic regression** and **deep learning**, require a real-valued representation:

$$\phi : \mathcal{X} \rightarrow \mathbb{R}^D$$

  - This is not always practical.
    - **Macromolecular sequences** having different lengths.
    - **Phylogenetic trees**.
  - Pairwise comparison methods are often readily available.
    - **Pairwise sequence comparison** (alignment)
    - **Pairwise tree comparison** (Robinson-Foulds, RFL, etc.)
- This enables **data fusion/integration**.

Representation - kernel methods

- “Most kernel methods [. . .] can only process square matrices, which are **symmetric *positive semidefinite***. This means that if k it is a $n \times n$ matrix of pairwise comparisons, it should satisfy $k_{i,j} = k_{j,i}$ for any $1 \leq i, j \leq n$, and $c^T kc \geq 0$ for any $c \in \mathbb{R}^n$." [1] page 38.

Representation - kernel methods

- For the case where $\mathcal{X} = \mathbb{R}^D$, the **inner product** (aka dot product or scalar product) can be used as a kernel function, it is known as the **linear kernel**.

Representation - kernel methods

- For the case where $\mathcal{X} = \mathbb{R}^D$, the **inner product** (aka dot product or scalar product) can be used as a kernel function, it is known as the **linear kernel**.
- For cases where the objects are **not vectors**, a **kernel function** can be defined as follows:

$$k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

Representation - kernel methods

- For the case where $\mathcal{X} = \mathbb{R}^D$, the **inner product** (aka dot product or scalar product) can be used as a kernel function, it is known as the **linear kernel**.
- For cases where the objects are **not vectors**, a **kernel function** can be defined as follows:

$$k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

- "**Theorem** *For any kernel k on a space $\mathcal{X}$, there exists a Hilbert space $\mathcal{F}$ and mapping $\phi : \mathcal{X} \to \mathcal{F}$ such that*

$$k(x, x') = \langle \phi(x), \phi(x') \rangle, \text{for any } x, x' \in \mathcal{X}$$

where $\langle u, v \rangle$ represents the dot product in the Hilbert space between any two points $u, v \in \mathcal{F}$." [1] page 40.

Kernel trick

- Finding the optimal value of $f(x)$ for a **support vector machine** requires solving a **quadratic programming problem**.

Kernel trick

- Finding the optimal value of $f(x)$ for a **support vector machine** requires solving a **quadratic programming problem**.
- After many mathematical transformations, buried into the equation to be solved, there is a **dot product of the transformed vectors**.

Kernel trick

- Finding the optimal value of $f(x)$ for a **support vector machine** requires solving a **quadratic programming problem**.
- After many mathematical transformations, buried into the equation to be solved, there is a **dot product of the transformed vectors**.
- Thanks to the previous theorem (Mercer's theorem), the dot product can be replaced by the value of the kernel in the original space.

Prologue

Summary

- Kernel methods use as input a $N \times N$ matrix, representing all **pairwise comparisons** between examples.

Summary

- Kernel methods use as input a $N \times N$ matrix, representing all **pairwise comparisons** between examples.
- This allows kernel methods to handle a greater range of data types than most learning algorithms.

Next module

- Fundamentals of **deep learning**

References

Berhhard Schölkopf, Koji Tsuda, and Jean-Philippe Vert, editors.
Kernel Methods in Computational Biology.
MIT Press, 2004.

Dennis Wylie, Hans A Hofmann, and Boris V Zemelman.
SArKS: de novo discovery of gene expression regulatory motif sites and domains by suffix array kernel smoothing.
Bioinformatics, Mar 2019.

Dexiong Chen, Laurent Jacob, and Julien Mairal.
Biological sequence modeling with convolutional kernel networks.
Bioinformatics, 35(18):3294–3302, Sep 2019.

Asa Ben-Hur and William Stafford Noble.
Kernel methods for predicting protein-protein interactions.
Bioinformatics, 21 Suppl 1:i38–46, Jun 2005.

References

Bin Liu, Deyuan Zhang, Ruifeng Xu, Jinghao Xu, Xiaolong Wang, Qingcai Chen, Qiwen Dong, and Kuo-Chen Chou.
Combining evolutionary information extracted from frequency profiles with sequence-based kernels for protein remote homology detection.
Bioinformatics, 30(4):472–9, Feb 2014.

Dawei Liu, Debashis Ghosh, and Xihong Lin.
Estimation and testing for the effect of a genetic pathway on a disease outcome using logistic kernel machine regression via logistic mixed models.
BMC Bioinformatics, 9:292, Jun 2008.

Rui Kuang, Eugene Ie, Ke Wang, Kai Wang, Mahira Siddiqi, Yoav Freund, and Christina Leslie.
Profile-based string kernels for remote homology detection and motif extraction.
J Bioinform Comput Biol, 3(3):527–50, Jun 2005.

Sören Sonnenburg, Alexander Zien, and Gunnar Rätsch.
ARTS: accurate recognition of transcription starts in human.
Bioinformatics, 22(14):e472–80, Jul 2006.

References

Xing Chen, Lei Wang, Jia Qu, Na-Na Guan, and Jian-Qiang Li.
Predicting miRNA-disease association based on inductive matrix completion.
Bioinformatics, 34(24):4256–4265, 12 2018.

Jesper Salomon and Darren R Flower.
Predicting Class II MHC-Peptide binding: a kernel based approach using similarity scores.
BMC Bioinformatics, 7:501, Nov 2006.

George Karypis.
YASSPP: better kernels and coding schemes lead to improvements in protein secondary structure prediction.
Proteins, 64(3):575–86, Aug 2006.

Jean-Philippe Vert, Jian Qiu, and William S Noble.
A new pairwise kernel for biological network inference with support vector machines.
BMC Bioinformatics, 8 Suppl 10:S8, 2007.

References

Seonho Kim, Juntae Yoon, Jihoon Yang, and Seog Park.
Walk-weighted subsequence kernels for protein-protein interaction extraction.
BMC Bioinformatics, 11:107, Feb 2010.

Mehmet Gönen and Adam A. Margolin.
Localized data fusion for kernel k-Means clustering with application to cancer biology.
In *NIPS*, pages 1305–1313, 2014.

Peter Meinicke, Maike Tech, Burkhard Morgenstern, and Rainer Merkl.
Oligo kernels for datamining on biological sequences: a case study on prokaryotic translation initiation sites.
BMC Bioinformatics, 5:169, Oct 2004.

Yingiun Ma, Limin Yu, Tingting He, Xiaohua Hu, and Xingpeng Jiang.
Prediction of long non-coding RNA-protein interaction through kernel soft-neighborhood similarity.
In *BIBM*, pages 193–196. IEEE Computer Society, 2018.

References

Han Zhang, Xueting Huo, Xia Guo, Xin Su, Xiongwen Quan, and Chen Jin.
A disease-related gene mining method based on weakly supervised learning model.
In *BIBM*, pages 169–174. IEEE Computer Society, 2018.

Dan Liu, Xiaohua Hu, Tingting He, and Xingpeng Jiang.
Virus-host association prediction by using kernelized logistic matrix factorization on heterogeneous networks.
In *BIBM*, pages 108–113. IEEE Computer Society, 2018.

Aurélien Géron.
Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow.
O'Reilly Media, 2nd edition, 2019.

William S Noble.
What is a support vector machine?
Nat Biotechnol, 24(12):1565–7, Dec 2006.

Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik.
A training algorithm for optimal margin classifiers.
In *COLT*, pages 144–152. ACM, 1992.

Marcel **Turcotte**
Marcel.Turcotte@uOttawa.ca

School of Electrical Engineering and **Computer Science** (EECS)
University of Ottawa