Subscribe to KDnuggets

Submit a blog to KDnuggets

search KDnuggets Search

- Blog
- Opinions
- Tutorials
- Top stories
- Courses
- Datasets
- Education: Online
- Certificates
- Events / Meetings
- Jobs
- Software
- Webinars

Powerful statistics with interactive visualization. Try JMP free for 30 days

Topics: AI | Data Science | Data Visualization | Deep Learning | Machine Learning | NLP | Python | R | Statistics

KDnuggets Home » News » 2020 » Mar » Tutorials, Overviews » A Top Machine Learning Algorithm Explained: Support Vector Machines (SVM) (20:n12)

A Top Machine Learning Algorithm Explained: Support Vector Machines (SVM)

<= Previous post
Next post =>

Like 21 Share 21 Tweet Share Share 76

Tags: Algorithms, Explained, Linear Algebra, Machine Learning, Support Vector Machines, SVM

Support Vector Machines (SVMs) are powerful for solving regression and classification problems. You should have this approach in your machine learning arsenal, and this article provides all the mathematics you need to know -- it's not as hard you might think.

Anaconda | Secure your open-source software supply chain | See how

---

By Clare Liu, Fintech industry.

comments

One of the most prevailing and exciting supervised learning models with associated learning algorithms that analyse data and recognise patterns is Support Vector Machines (SVMs). It is used for solving both regression and classification problems. However, it is mostly used in solving classification problems. SVMs were first introduced by B.E. Boser et al. in 1992 and has become popular due to success in handwritten digit recognition in 1994. Before the emergence of Boosting Algorithms, for example, XGBoost and AdaBoost, SVMs had been commonly used.

If you want to have a consolidated foundation of Machine Learning algorithms, you should definitely have it in your arsenal. The algorithm of SVMs is powerful, but the concepts behind are not as complicated as you think.

Problem with Logistic Regression

In my previous article, I have explained clearly what Logistic Regression is (link). It helps solve classification problems separating the instances into two classes. However, there is an infinite number of decision boundaries, and Logistic Regression only picks an arbitrary one.

- For point C, since it's far away from the decision boundary, we are quite certain to classify it as 1 (green).
- For point A, even though we classify it as 1 for now, since it is pretty close to the decision boundary, if the boundary moves a little to the right, we would mark point A as "0" instead. Hence, we're much more confident about our prediction at C than at A

Logistic Regression doesn't care whether the instances are close to the decision boundary. Therefore, the decision boundary it picks may not be optimal. As we can see from the above graph, if a point is far from the decision boundary, we may be more confident in our predictions. Therefore, the optimal decision boundary should be able to maximize the distance between the decision boundary and all instances. i.e., maximize the margins. That's why the SVM algorithm is important!

What is Support Vector Machines (SVMs)?

Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

The objective of applying SVMs is to find the best line in two dimensions or the best hyperplane in more than two dimensions in order to help us separate our space into classes. The hyperplane (line) is found through the **maximum margin,** i.e., the maximum distance between data points of both classes.

> *Don't you think the definition and idea of SVM look a bit abstract? No worries, let me explain in details.*

Support Vector Machines

Imagine the labelled training set are two classes of data points (two dimensions): Alice and Cinderella. To separate the two classes, there are so many possible options of hyperplanes that separate correctly. As shown in the graph below, we can achieve exactly the same result using different hyperplanes (L1, L2, L3). However, if we add new data points, the consequence of using various hyperplanes will be very different in terms of classifying new data point into the right group of class.

Different hyperplanes (L1, L2, L3).

How can we decide a separating line for the classes? Which hyperplane shall we use?

Support Vector, Hyperplane, and Margin

The vector points closest to the hyperplane are known as the **support vector points** because only these two points are contributing to the result of the algorithm, and other points are not. If a data point is not a support vector, removing it has no effect on the model. On the other hand, deleting the support vectors will then change the position of the hyperplane.

The dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

The distance of the vectors from the hyperplane is called the ***margin,*** which is a separation of a line to the closest class points. We would like to choose a hyperplane that maximises the margin between classes. The graph below shows what good margin and bad margin are.

Good Margin

- all sector vectors have the same distance with the maximum margin hyperplane

Bad Margin

- very close to either class -1 support vectors or class +1 support vectors

Hard Margin

If the training data is linearly separable, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible.

Soft Margin

As most of the real-world data are not fully linearly separable, we will allow some margin violation to occur, which is called soft margin classification. It is better to have a large margin, even though some constraints are violated. Margin violation means choosing a hyperplane, which can allow some data points to stay in either the incorrect side of the hyperplane and between the margin and the correct side of the hyperplane.

In order to find the **maximal margin**, we need to maximize the margin between the data points and the hyperplane. In the following session, I will share the mathematical concepts behind this algorithm.

Linear Algebra Revisited

Before we move on, let's review some concepts in Linear Algebra.

We can expand our 2-dimensional (x-y) coordinate system into a 3-dimensional coordinate system, using x-, y-, and z-axes. The general point P (2, 3, 5) is shown in graph.

A vector is denoted by a bolded alphabet (e.g., $\mathbf{x}$).

The magnitude of a n-dimensional vector can be calculated as follows:

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

A unit vector $\hat{\mathbf{x}}$ is a vector with length 1 and it can be calculated by:

$$\hat{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$$

Maximising the Margin

You probably learned that an equation of a line is **y=ax+b**. However, you will often find that the equation of a hyperplane is defined by:

Note that

$$y = ax + b$$

is the same thing as

$$y - ax - b = 0$$

Given two vectors $\mathbf{w}\begin{pmatrix} -b \\ -a \\ 1 \end{pmatrix}$ and $\mathbf{x}\begin{pmatrix} 1 \\ x \\ y \end{pmatrix}$

$$\mathbf{w}^T\mathbf{x} = -b \times (1) + (-a) \times x + 1 \times y$$

$$\mathbf{w}^T\mathbf{x} = y - ax - b$$

The two equations are just two different ways of expressing the same thing.

For **Support Vector Classifier** (SVC), we use $\mathbf{w}^T\mathbf{x}+b$ where $\mathbf{w}$ is the weight vector, and b is the bias.

$$\mathbf{w}^T\mathbf{x} + b = 0$$

You can see that the name of the variables in the hyperplane equation are $\mathbf{w}$ and $\mathbf{x}$, which means they are vectors! A vector has magnitude (size) and direction, which works perfectly well in 3 or more dimensions. Therefore, the application of "**vector**" is used in the SVMs algorithm.

Suppose we have two support vectors where one is a positive example (e.g., $\mathbf{x}^+$) and the other is a negative example (e.g., $\mathbf{x}^-$)

We also have the weight vector $\mathbf{w}$ which is orthogonal to the decision boundary

Since the support vectors define the margin, to calculate the margin, we can first subtract $\mathbf{x}^+$ by $\mathbf{x}^-$, and perform dot product on the subtracted vector and the unit vector of $\mathbf{w}$

$$(\mathbf{x}^+ - \mathbf{x}^-) \cdot \hat{\mathbf{w}} = (\mathbf{x}^+ - \mathbf{x}^-) \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} = \mathbf{x}^+ \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} - \mathbf{x}^- \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|}$$

The equation of calculating the Margin.

Cost Function and Gradient Updates

Maximizing-Margin is equivalent to Minimizing Loss.

In the SVM algorithm, we are looking to maximize the **margin** between the data points and the hyperplane. The loss function that helps maximize the margin is **hinge loss**.

$$L(w) = \sum_{i=1} \underbrace{max(0, 1 - y_i[w^T x_i + b])}_{\text{Loss function}} \underbrace{+ \lambda ||w||_2^2}_{\text{regularization}}$$

λ=1/C (C is always used for regularization coefficient).

The function of the first term, **hinge loss,** is to penalize misclassifications. It measures the error due to misclassification (or data points being closer to the classification boundary than the margin). The second term is the regularization term, which is a technique to avoid overfitting by penalizing large coefficients in the solution vector. The λ(lambda) is the regularization coefficient, and its major role is to determine the trade-off between increasing the margin size and ensuring that the xi lies on the correct side of the margin.

“Hinge” describes the fact that the error is 0 if the data point is classified correctly (and is not too close to the decision boundary).

When the true class is -1 (as in your example), the hinge loss looks like this in the graph.

We need to minimise the above loss function to find the max-margin classifier.

We can derive the formula for the margin from the **hinge-loss**. If a data point is on the margin of the classifier, the hinge-loss is exactly zero. Hence, on the margin, we have:

$$max(0, 1 - y_i[w^T x_i + \ b\ ]) = 0$$
$$\Rightarrow y_i[w^T x_i + \ b\ ] = 1$$

Note that yi is either +1 or -1.

Therefore, we have:

$$\mathbf{x}^+ \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} - \mathbf{x}^- \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} = \frac{1-b}{\|\mathbf{w}\|} - \frac{-b-1}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

Our objective function is then:

$$\max \frac{2}{\|\mathbf{w}\|} \rightarrow \max \frac{1}{\|\mathbf{w}\|} \rightarrow \min \|\mathbf{w}\| \rightarrow \ \min \frac{1}{2} \|\mathbf{w}\|^2$$

To minimize such an objection function, we should then use Lagrange Multiplier.

Classifying non-linear data

What about data points are not linearly separable?

Non-linear separate.

SVM has a technique called the **kernel trick**. These are functions that take low dimensional input space and transform it into a higher-dimensional space, i.e., it converts not separable problem to separable problem. It is mostly useful in non-linear separation problems. This is shown as follows:

Mapping to a Higher Dimension

Case 1 - 1 Dimensional Space

- Points in 1 Dimension Plan.
- Represented by function f(x)=x
- They cannot be separated or classified.

Case 2 - 2 Dimensional Space

- Transformed into higher Dimension.
- Represented by function f(x)=x^2
- They care now be easily separated or classified.

Some Frequently Used Kernels

Gaussian RBF Kernel

$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

Sigmoid Kernel

$$K(X, Y) = \tanh(\gamma \cdot X^T Y + r)$$

Polynomial Kernel

$$K(X, Y) = (\gamma \cdot X^T Y + r)^d, \gamma > 0$$

Why SVMs

1. Solve the data points are not linearly separable
2. Effective in a higher dimension.
3. Suitable for small data set: effective when the number of features is more than training examples.
4. Overfitting problem: The hyperplane is affected by only the support vectors, so SVMs are not robust to the outliner.

Summary: Now you should know

- Problem with Logistic Regression
- What is Support Vector, Hyperplane, and Margin
- How to find the maximised margin using hinge-loss
- How to deal with non-linear separable data using different kernels

Original. Reposted with permission.

Related:

- A Friendly Introduction to Support Vector Machines
- Support Vector Machine (SVM) Tutorial: Learning SVMs From Examples
- What is a Support Vector Machine, and Why Would I Use it?

What do you think?

41 Responses

Upvote Funny Love Surprised Angry Sad

5 Comments **KDnuggets** **Disqus' Privacy Policy** **Login**

Favorite Tweet Share **Sort by Best**

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Abha Sharma • 2 months ago

I guess this piece of information is very detailed

Reply • Share ›

Gregory Piatetsky Mod → Abha Sharma • 2 months ago

Abha, why are you making so many meaningless comments on so many KDnuggets

---

<= Previous post
Next post =>

Top Stories Past 30 Days

Most Popular

1. **How I Tripled My Income With Data Science in 18 Months**
2. **How to Build Strong Data Science Portfolio as a Beginner**
3. **Data Scientist vs Data Engineer Salary**
4. **The 20 Python Packages You Need For Machine Learning and Data Science**
5. **Data science SQL interview questions from top tech firms**

Most Shared

1. **Data Science Portfolio Project Ideas That Can Get You Hired (Or Not)**
2. **Exclusive: OpenAI summarizes KDnuggets**
3. **38 Free Courses on Coursera for Data Science**
4. **How I Tripled My Income With Data Science in 18 Months**
5. **Teaching AI to Classify Time-series Patterns with Synthetic Data**

Latest News

- Top Stories, Oct 18-24: How I Tripled My Income With Da...
- 365 Data Science courses free until 18 November
- Guide To Finding The Right Predictive Maintenance Machi...
- 2021 Data Engineer Salary Report Shares Insights on a S...
- Save Sarah Connor with Data Science
- Learn To Reproduce Papers: Beginner's Guide

Top Stories Last Week

Most Popular

1. **How I Tripled My Income With Data Science in 18 Months**
2. **Data Scientist vs Data Engineer Salary**
3. **The 20 Python Packages You Need For Machine Learning and Data Science**
4. **Real Time Image Segmentation Using 5 Lines of Code**
5. **Data Science Portfolio Project Ideas That Can Get You Hired (Or Not)**

Most Shared

1. **Data Science Portfolio Project Ideas That Can Get You Hired (Or Not)**
2. **Exclusive: OpenAI summarizes KDnuggets**
3. **How I Tripled My Income With Data Science in 18 Months**
4. **Avoid These Five Behaviors That Make You Look Like A Data Novice**
5. **Introduction to AutoEncoder and Variational AutoEncoder (VAE)**

More Recent Stories

- Learn To Reproduce Papers: Beginner's Guide
- Exclusive: OpenAI summarizes KDnuggets **[Silver Blog]**
- How to Transform Your Data in Snowflake
- Deploying Serverless spaCy Transformer Model with AWS Lambda
- Introduction to AutoEncoder and Variational AutoEncoder (VAE)
- Find the Best-Matching Distribution for Your Data Effortlessly
- DATAnalyze 2021 Analytics Hackathon Sponsored by Microsoft and...
- Training BPE, WordPiece, and Unigram Tokenizers from Scratch u...
- How I Tripled My Income With Data Science in 18 Months **[Gold Blog]**
- Simple Question Answering Web App with HuggingFace Pipelines
- Level-Up This November with the ODSC West 2021 Keynotes and Tr...
- Data Preparation in R using dplyr, with Cheat Sheet!
- Data Science Portfolio Project Ideas That Can Get You Hired (Or Not) **[Gold Blog]**
- Data Scientist vs Data Engineer Salary **[Silver Blog]**
- KDnuggets 21:n40, Oct 20: The 20 Python Packages You Need F...
- How Data Professionals Can Impress Even When Busy
- 11 Most Practical Data Science Skills for 2022
- How to Create an Interactive Dashboard in Three Steps with KNI...
- Top Stories, Oct 11-17: Query Your Pandas DataFrames with SQL
- Knowledge Graph Forum: Technology Ecosystem and Business Appli...

© 2021 KDnuggets. | About KDnuggets | Contact | Privacy policy | Terms of Service

Subscribe to KDnuggets News
X