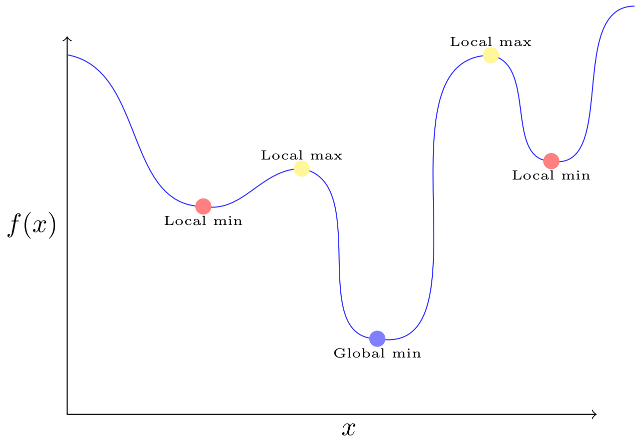
$\min_{x\in\mathbb{R}^n} f(x)$	$\min_{x\in\mathbb{R}^n} f(x)$ $f_i(x) \le 0 \quad \text{for } i = 1,\dots,m$
Unconstrained Minimization	Constrained Minimization

- **objective function** $f\colon \mathbb{R}^n \to \mathbb{R}$
- may **maximize** f by minimizing the function $g(x) := -f(x)$

Definition 1 (Global and local optima)**.** *The vector x^* is a*

- global minimizer *if $f(x^*) \le f(x)$ for all $x \in \mathbb{R}^n$.*
- local minimizer *if $f(x^*) \le f(x)$ for all x satisfying $\|x - x^*\| \le \epsilon$ for some $\epsilon > 0$.*
- strict local minimizer *if $f(x^*) < f(x)$ for all $x \neq x^*$ satisfying $\|x - x^*\| \le \epsilon$ for some $\epsilon > 0$.*



Theorem 2. *Let S be a nonempty set that is closed and bounded. Suppose that $f\colon S \to \mathbb{R}$ is continuous. Then the problem $\min\{f(x) : x \in S\}$ attains its minimum.*

Definition 3. *A* critical point *is a point $\bar{x}$ where $\nabla f(\bar{x}) = 0$.*

Theorem 4. *Suppose that $f\colon \mathbb{R}^n \to \mathbb{R}$ is differentiable. If $\min\{f(x) : x \in \mathbb{R}^n\}$ has an optimizer x^*, then x^* is a critical point of f (i.e., $\nabla f(x^*) = 0$).*

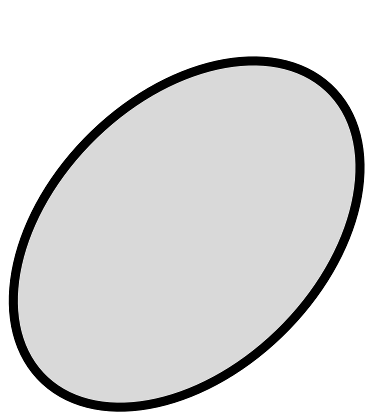
0.1 Convex Sets

Definition 5 (Convex Combination)**.** *Given two points x, y, a* convex combination *is any point z that lies on the line between x and y. Algebraically, a* convex combination *is any point z that can be represented as $z = \lambda x + (1-\lambda)y$ for some multiplier $\lambda \in [0,1]$.*

Definition 6 (Convex Set)**.** *A set C is convex if it contains all convex combinations of points in C. That is, for any $x, y \in C$, it holds that $\lambda x + (1-\lambda)y \in C$ for all $\lambda \in [0,1]$.*

Definition 7 (Convex Sets)**.** *A set S is* convex *if for any two points in S, the entire line segment between them is also contained in S. That is, for any $x, y \in S$*

$$\lambda x + (1 - \lambda y) \in S \quad \textit{for all } \lambda \in [0,1].$$


(a) Convex Set

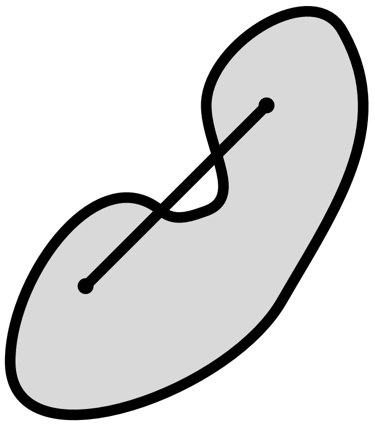

(b) Non-convex set

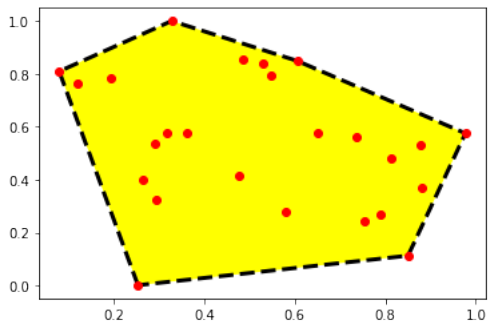
Figure 1: *Examples of convex and non-convex sets.*

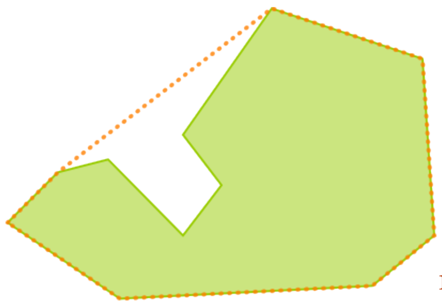
Examples Convex Sets

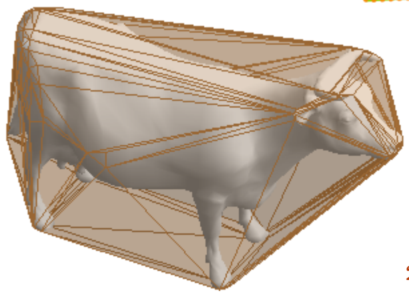
1. *Hyperplane* $H = \{x \in \mathbb{R}^n : a^\top x = b\}$
2. *Halfspace* $H = \{x \in \mathbb{R}^n : a^\top x \leq b\}$
3. *Polyhedron* $P = \{x \in \mathbb{R}^n : Ax \leq b\}$
4. *Second Order Cone* $S = \{(x,t) \in \mathbb{R}^n \times \mathbb{R} : \sum_{i=1}^n x_i^2 \leq t^2\}$

Definition 8 (Convex Hull)**.** *Let* $S \subseteq \mathbb{R}^n$*. The convex hull* $\mathrm{conv}(S)$ *is the smallest convex set containing* S*.*

t




1


2

Theorem 9 (Caratheodory's Theorem). *Let $x \in \operatorname{conv}(S)$ and $S \subseteq \mathbb{R}^n$. Then there exist $x^1, \dots, x^k \in S$ such that $x \in \operatorname{conv}(\{x^1, \dots, x^k\})$ and $k \le n+1$.*

0.2 Convex Functions

Convex function are "nice" functions that "open up". They represent an extremely important class of functions in optimization and typically can be optimized over efficienty.

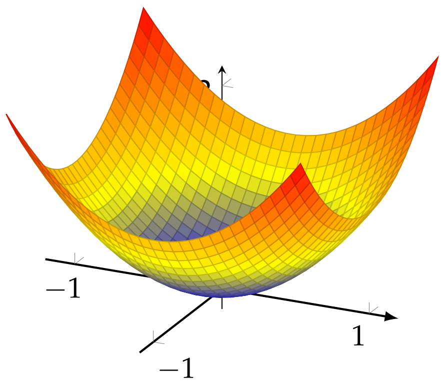
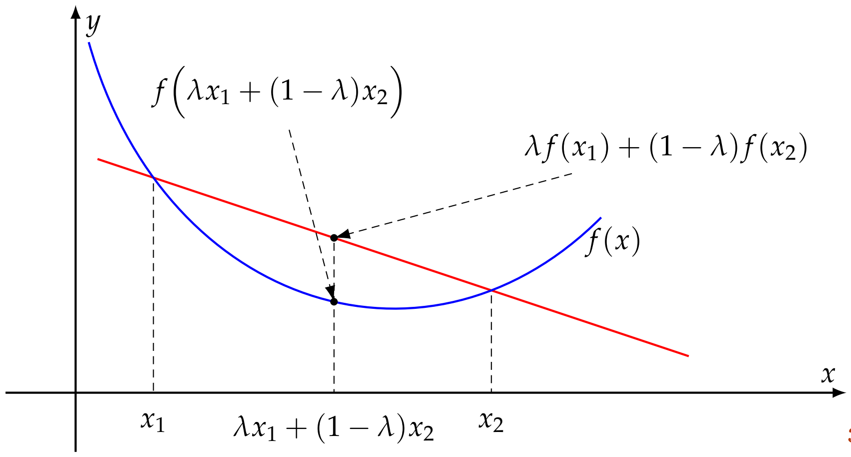


Figure 2: *Convex Function $f(x,y) = x^2 + y^2$.*

Definition 10 (Convex Functions). *A function $f \colon \mathbb{R}^n \to \mathbb{R}$ is* convex *if for all $x, y \in \mathbb{R}^n$ and $\lambda \in [0,1]$ we have*

$$\lambda f(x) + (1-\lambda) f(y) \ge f(\lambda x + (1-\lambda) y). \tag{0.2.1}$$



y
$f\Big(\lambda x_1 + (1-\lambda)x_2\Big)$
$\lambda f(x_1) + (1-\lambda) f(x_2)$
$f(x)$
x
x_1
$\lambda x_1 + (1-\lambda)x_2$
x_2

[3]

An equivalent definition of convex function are through the epigraph.

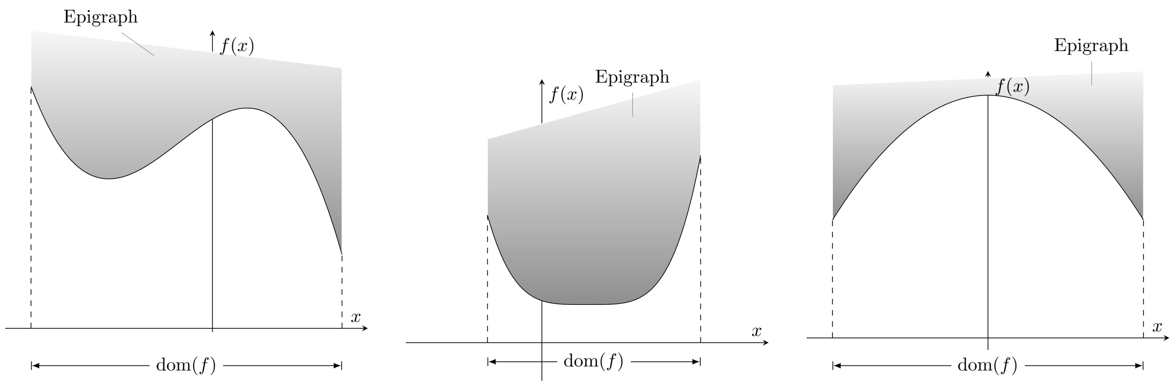
[2] https://www.boost.org/doc/libs/1_63_0/libs/geometry/doc/html/geometry/reference/algorithms/convex_hull.html

[2] https://www.wolfram.com/mathematica/new-in-10/data-and-mesh-regions/convex-hulls.html

[3] https://tex.stackexchange.com/questions/394923/how-one-can-draw-a-convex-function

Definition 11 (Epigraph). *The* epigraph *of f is the set $\{(x,y) : y \geq f(x)\}$. This is the set of all points "above" the function.*

Theorem 12. *$f(x)$ is a convex function if and only if the epigraph of f is a convex set.*



[4]

Example 1: Examples of Convex functions
Some examples are

- $f(x) = ax + b$
- $f(x) = x^2$
- $f(x) = x^4$
- $f(x) = |x|$
- $f(x) = e^x$
- $f(x) = -\sqrt{x}$ on the domain $[0, \infty)$.
- $f(x) = x^3$ on the domain $[0, \infty)$.
- $f(x,y) = \sqrt{x^2 + y^2}$
- $f(x,y) = x^2 + y^2 + x$
- $f(x,y) = e^{x+y}$
- $f(x,y) = e^x + e^y + x^2 + (3x + 4y)^6$

[4] https://tex.stackexchange.com/questions/261501/function-epigraph-possibly-using-fillbetween

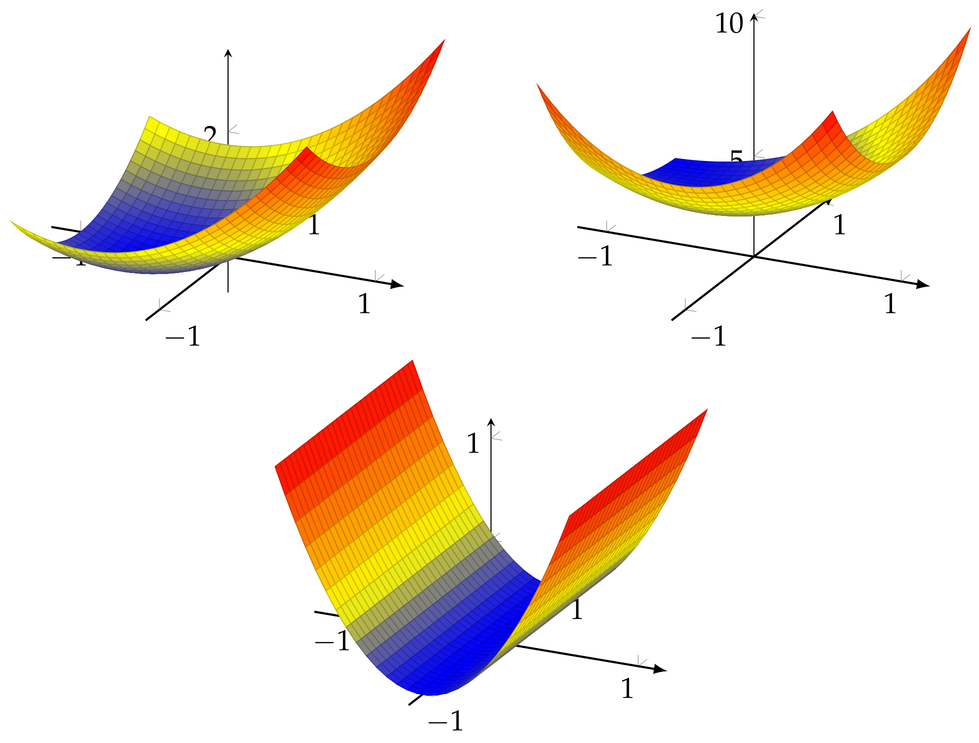
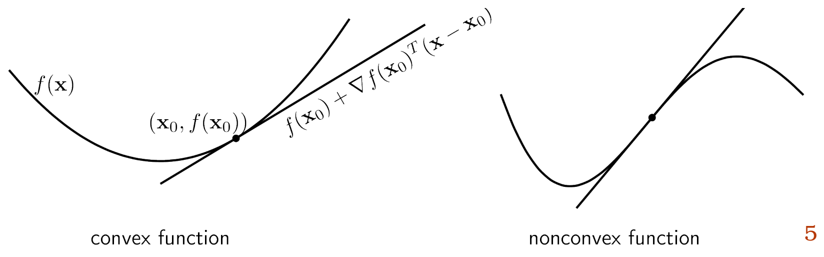


Figure 3: *Convex Functions* $f(x,y) = x^2 + y^2 + x$, $f(x,y) = e^{x+y} + e^{x-y} + e^{-x-y}$, *and* $f(x,y) = x^2$.

0.2.1 Proving Convexity - Characterizations

Theorem 13 (Convexity: First order characterization - linear underestimates)**.** *Suppose that* $f\colon \mathbb{R}^n \to \mathbb{R}$ *is differentiable. Then* f *is convex if and only if for all* $\bar{x} \in \mathbb{R}^n$*, then linear tangent is an underestimator to the function, that is,*

$$f(\bar{x}) + (x - \bar{x})^\top \nabla f(\bar{x}) \le f(x).$$


$f(\mathbf{x})$
$(\mathbf{x}_0, f(\mathbf{x}_0))$
$f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^T(\mathbf{x} - \mathbf{x}_0)$
convex function
nonconvex function
[5]

Theorem 14 (Convexity: Second order characterization - positive curvature)**.** *We give statements for uni-variate functions and multi-variate functions.*

- *Suppose* $f\colon \mathbb{R} \to \mathbb{R}$ *is twice differentiable. Then* f *is convex if and only if* $f''(x) \ge 0$ *for all* $x \in \mathbb{R}$.

[5] https://machinelearningcoban.com/2017/03/12/convexity/

- *Suppose $f\colon \mathbb{R}^n \to \mathbb{R}$ is twice differentiable. Then f is convex if and only if $\nabla^2 f(x) \preceq 0$ for all $x \in \mathbb{R}^n$.*

0.2.2 Proving Convexity - Composition Tricks

Positive Scaling of Convex Function is Convex:
If f is convex and $\alpha > 0$, then αf is convex.
Example: $f(x) = e^x$ is convex. Therefore, $25e^x$ is also convex.

Sum of Convex Functions is Convex:
If f and g are both convex, then $f + g$ is also convex.
Example: $f(x) = e^x, g(x) = x^4$ are convex. Therefore, $e^x + x^4$ is also convex.

Composition with affine function:
If $f(x)$ is convex, then $f(a^\top x + b)$ is also convex.
Example: $f(x) = x^4$ are convex. Therefore, $(3x + 5y + 10z)^4$ is also convex.

Pointwise maximum:
If f_i are convex for $i = 1, \dots, t$, then $f(x) = \max_{i=1,\dots,t} f_i(x)$ is convex.
Example: $f_1(x) = e^{-x}$, $f_2(x) = e^x$ are convex. Therefore, $f(x) = \max(e^x, e^{-x})$ is also convex.

Other compositions:
Suppose

$$f(x) = h(g(x)).$$

1. If g is convex, h is convex and **non-decreasing**, then f is convex.
2. If g is concave, h is convex and **non-increasing**, then f is convex.

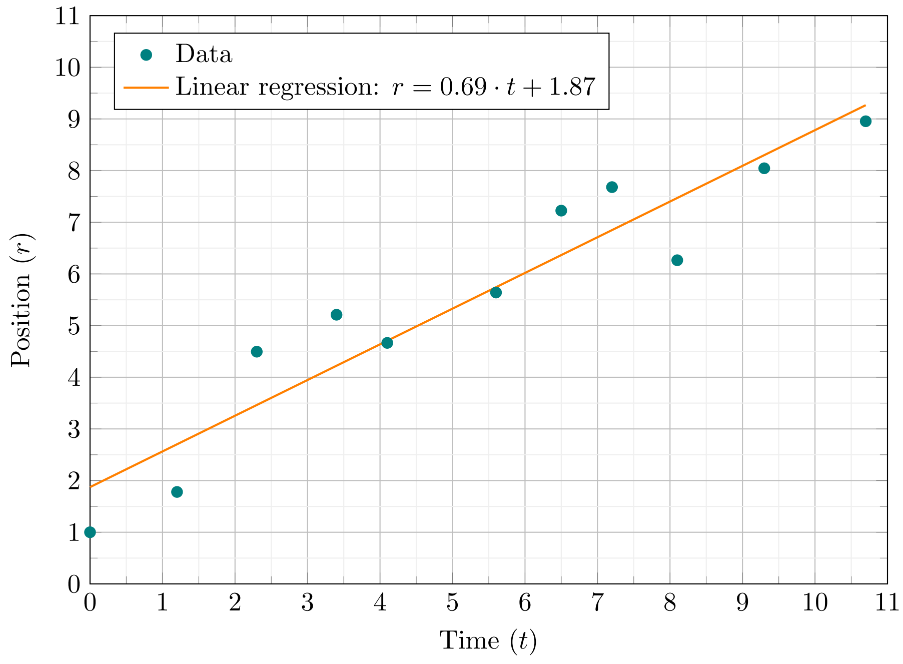
Example 1: $g(x) = x^4$ is convex, $h(x) = e^x$ is convex and non-decreasing. Therefore, $f(x) = e^{x^4}$ is also convex.

Example 2: $g(x) = \sqrt{x}$ is concave (on $[0, \infty)$), $h(x) = e^{-x}$ is convex and non-increasing. Therefore, $f(x) = e^{-\sqrt{x}}$ is convex on $x \in [0, \infty)$.

0.3 Convex Optimization Examples

0.3.1 Unconstrained Optimization: Linear Regression

Given data points $x^1, \ldots, x^N \in \mathbb{R}^d$ and out values $y^i \in \mathbb{R}$, we want to find a linear function $y = \beta \cdot x$ that best approximates $x^i \cdot \beta \approx y^i$. For example, the data could $x =$ (time) and the output could be $y =$ position.



[6]

As is standard, we choose the error (or "loss") from each data point as the squared error. Hence, we can model this as the optimization problem:

$$\min_{\beta \in \mathbb{R}^d} \sum_{i=1}^{N} (x^i \cdot \beta - y^i)^2 \tag{0.3.1}$$

This problem has a nice closed form solution. We will derive this solution, and then in a later section discuss why using this solution might be too slow to compute on large data sets. In particular, the solution comes as a system of linear equations. But when N is really large, we may not have time to solve this system, so an alternative is to use decent methods, discussed later in this chapter.

Theorem 15 (Linear Regression Solution)**.** *The solution to* (0.3.1) *is*

$$\beta = (X^\top X)^{-1} X^\top Y, \tag{0.3.2}$$

where

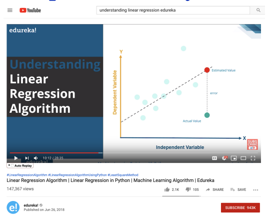
$$X = \begin{bmatrix} x^1 \\ \vdots \\ x^N \end{bmatrix} = \begin{bmatrix} x_1^1 & x_2^1 & \ldots & x_d^1 \\ \vdots & & & \\ x_1^N & x_2^N & \ldots & x_d^N \end{bmatrix}$$

[6]https://latexdraw.com/linear-regression-in-latex-using-tikz/

Proof. Solve for $\nabla f(\beta) = 0$.

To be completed.... □

https://www.youtube.com/watch?v=E5RjzSK0fvY


YouTube
understanding linear regression edureka
edureka!
Understanding Linear Regression Algorithm
Y
Dependent Variable
Estimated Value
error
Actual Value
X
Independent Variable
10:12 / 28:35
Auto Replay
#LinearRegressionAlgorithm #LinearRegressionAlgorithmUsingPython #LeastSquareMethod
Linear Regression Algorithm | Linear Regression in Python | Machine Learning Algorithm | Edureka
147,367 views
2.1K
105
SHARE
SAVE
edureka!
Published on Jun 26, 2018
SUBSCRIBE 943K

0.4 Machine Learning - SVM

Support Vector Machine (SVM) is a tool used in machine learning for classifying data points. For instance, if there are red and black data points, how can we find a good line that separates them? The input data that you are given is as follows:

Input:

- d-dimensional data points $x^1, \dots, x^N$
- 1-dimensional labels $z^1, \dots, z^N$ (typically we will use z_i is either 1 or -1)

The output to the problem should he a hyperplane $w^\top x + b = 0$ that separates the two data types (either exact separation or approximate separation).

Output:

- A d-dimensional vector w
- A 1-dimensional value b

Given this output, we can construct a classification function $f(x)$ as

$$f(x) = \begin{cases} 1 & \text{if } w^\top x + b \geq 0, \\ -1 & \text{if } w^\top x + b < 0. \end{cases} \tag{0.4.1}$$

There are three versions to consider:

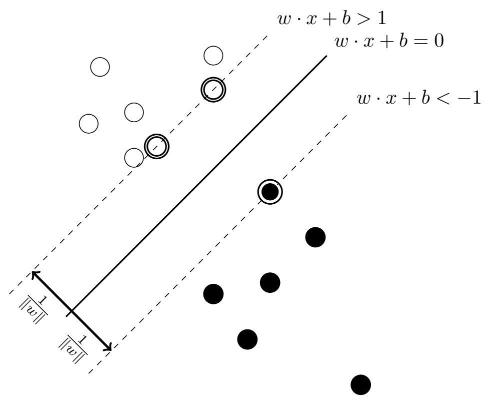
Feasible separation

If we only want to a line that separates the data points, we can use the following optimization model.

$$\begin{aligned} \min \quad & 0 & \\ \text{such that} \quad & z^i(w^\top x^i + b) \geq 1 & \text{for all } i = 1, \dots, N \\ & w \in \mathbb{R}^d & \\ & b \in \mathbb{R} & \end{aligned}$$

SVM

We can modify the objective function to find a best separation between the points. This can be done in the following way



$$\begin{aligned} \min \quad & \|w\|_2^2 \\ \text{such that} \quad & z^i(w^\top x^i + b) \geq 1 && \text{for all } i = 1, \ldots, N \\ & w \in \mathbb{R}^d \\ & b \in \mathbb{R} \end{aligned}$$

Here, $\|w\|_2^2 = \sum_{i=1}^d w_i^2 = w_1^2 + \cdots + w_d^2$.

Approximate SVM

We can modify the objective function and the constraints to allow for approximate separation. This would be the case when you want to ignore outliers that don't fit well with the data set, or when exact SVM is not possible. This is done by changing the constraints to be

$$z^i(w^\top x^i + b) \geq 1 - \delta_i$$

where $\delta_i \geq 0$ is the error in the constraint for datapoint i. In order to reduce these errors, we add a penalty term in the objective function that encourages these errors to be small. For this, we can pick some number C and write the objective as

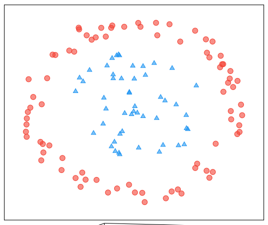
$$\min \|w\|_2^2 + C \sum_{i=1}^N \delta_i.$$

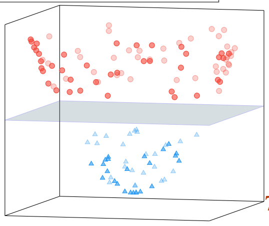
This creates the following optimization problem

$$\begin{aligned} \min \quad & \|w\|_2^2 + C \sum_{i=1}^N \delta_i \\ \text{such that} \quad & z^i(w^\top x^i + b) \geq 1 - \delta_i && \text{for all } i = 1, \ldots, N \\ & w \in \mathbb{R}^d \\ & b \in \mathbb{R} \\ & \delta_i \geq 0 \text{ for all } i = 1, \ldots, N \end{aligned}$$

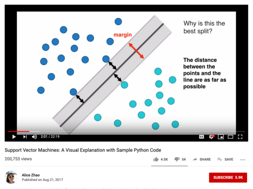
See information about the scikit-learn module for svm here: https://scikit-learn.org/stable/modules/svm.html.

0.4.1 SVM with non-linear separators



[7]

https://www.youtube.com/watch?time_continue=6&v=N1vOgolbjSc



The distance between the points and the line are as far as possible

Support Vector Machines: A Visual Explanation with Sample Python Code

200,753 views

Alice Zhao

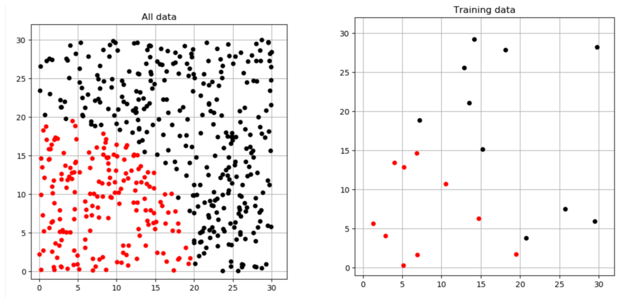
Published on Aug 21, 2017

SVMs are a popular classification technique used in data science and machine learning.

Suppose for instance you are given data $x^1, \dots, x^N \in \mathbb{R}^2$ (2-dimensional data) and given labels are dependent on the distance from the origin, that is, all data points x with $x_1^2 + x_2^2 > r$ are given a label $+1$ and all data points with $x_1^2 + x_2^2 \leq r$ are given a label -1. That is, we want to learn the function

$$f(x) = \begin{cases} 1 & \text{if } x_1^2 + x_2^2 > r, \\ -1 & \text{if } x_1^2 + x_2^2 \leq r. \end{cases} \tag{0.4.2}$$

Example 2:



Here we have a classification problem where the data cannot be separated by a hyperplane. On the left, we have all of the data given to use. On the right, we have a subset

[7] https://zenodo.org/record/2564435

of the data that we could try using for training and then test our learned function on the remaining data. As we saw in class, this amount of data was not sufficient to properly classify most of the data.

We cannot learn this classifier from the data directly using the hyperplane separation with SVM in the last section. But, if we modify the data set, then we can do this.

For each data point x, we transform it into a data point X by adding a third coordinate equal to $x_1^2 + x_2^2$. That is

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad \to \quad X = \begin{pmatrix} x_1 \\ x_2 \\ x_1^2 + x_2^2 \end{pmatrix}. \tag{0.4.3}$$

In this way, we convert the data $x^1, \dots, x^N$ into data $X^1, \dots, X^N$ that lives in a higher-dimensional space. But with this new dataset, we can apply the hyperplane separation technique in the last section to properly classify the data.

This can be done with other nonlinear classification functions.

0.4.2 Support Vector Machines

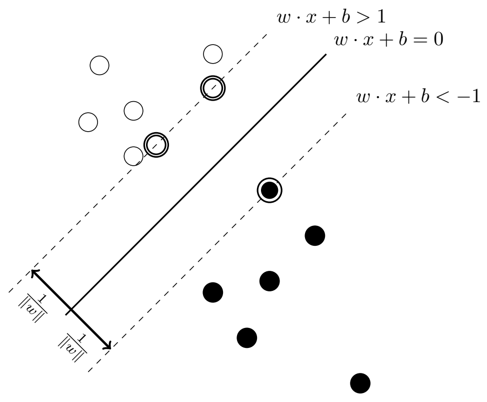
Support Vector Machine - Exact Classification:

Given labeled data (x^i, y_i) for $i = 1, \dots, N$, where $x^i \in \mathbb{R}^d$ and $y^i \in \{-1, 1\}$, find a vector $w \in \mathbb{R}^d$ and a number $b \in \mathbb{R}$ such that

$$x^i \cdot w + b > 0 \qquad \text{if } y^i = 1 \tag{0.4.4}$$

$$x^i \cdot w + b < 0 \qquad \text{if } y^i = -1 \tag{0.4.5}$$

There may exist many solutions to this problem. Thus, we are interested in the "best" solution. Such a solution will maximize the separation between the two sets of points. To consider an equal margin on either size, we set the right hand sides to 1 and -1 and then compute the margin form the hyperplane. Notice that it is sufficient to use 1 and -1 on the right hand sides since any scaling can happen in w and b.



We will show that the margin under this model can be computed as $\frac{2}{\|w\|}$ where $\|w\| = \sqrt{w_1^2 + \cdots + w_d^2}$. Hence, maximizing the margin is equivalent to minimizing $w_1^2 + \cdots + w_d^2$. We arrive at the model

$$\min \sum_{i=1}^{d} w_i^2 \tag{0.4.6}$$

$$x^i \cdot w + b \geq 1 \qquad \text{if } y^i = 1 \tag{0.4.7}$$

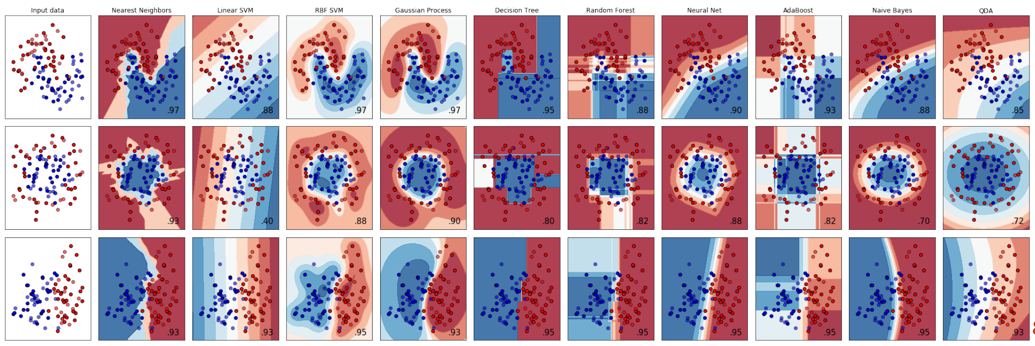
$$x^i \cdot w + b \leq -1 \qquad \text{if } y^i = -1 \tag{0.4.8}$$

Or even more compactly written as

$$\min \sum_{i=1}^{d} w_i^2 \tag{0.4.9}$$

$$y^i(x^i \cdot w + b) \geq 1 \qquad \text{for } i = 1, \ldots, N \tag{0.4.10}$$

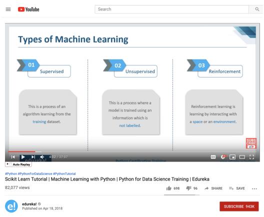
0.5 Classification



[8]

0.5.1 Machine Learning

https://www.youtube.com/watch?v=bwZ3Qiuj3i8&list=PL9ooVrP1hQOHUfd-g8GUpKI3hHOwM_



Types of Machine Learning

01 Supervised

This is a process of an algorithm learning from the training dataset.

02 Unsupervised

This is a process where a model is trained using an information which is not labelled.

03 Reinforcement

Reinforcement learning is learning by interacting with a space or an environment.

#Python #PythonForDataScience #PythonTutorial

Scikit Learn Tutorial | Machine Learning with Python | Python for Data Science Training | Edureka

82,077 views 698 96

edureka!

Published on Apr 18, 2018

9Dn&index=13

https://towardsdatascience.com/solving-a-simple-classification-problem-with-python-fruits-lo

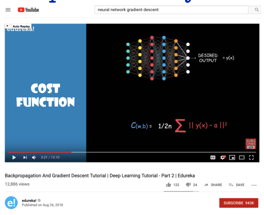
0.5.2 Neural Networks

https://www.youtube.com/watch?v=bVQUSndDllU

https://www.youtube.com/watch?v=8bNIkfRJZpo

[8]From Scikitlearn webpage https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

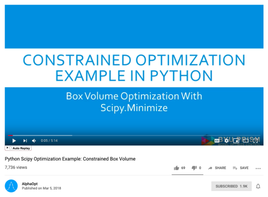
https://www.youtube.com/watch?v=Dws9Zveu9ug



Backpropagation And Gradient Descent Tutorial | Deep Learning Tutorial - Part 2 | Edureka
12,886 views
edureka!
Published on Aug 26, 2018

0.6 Box Volume Optimization in Scipy.Minimize

https://www.youtube.com/watch?v=iSnTtV6b0Gw


CONSTRAINED OPTIMIZATION EXAMPLE IN PYTHON
Box Volume Optimization With Scipy.Minimize
0:05 / 5:14
Auto Replay
69
0
SHARE
SAVE
SUBSCRIBED 1.9K

Python Scipy Optimization Example: Constrained Box Volume
7,736 views
AlphaOpt
Published on Mar 5, 2018

0.7 Modeling

We will discuss a few models and mention important changes to the models that will make them solvable.

Important tips

1. **Find a convex formulation.** It may be that the most obvious model for your problem is actually non-convex. Try to reformulate your model into one that is convex and hence easier for solvers to handle.

2. **Intelligent formulation.** Understanding the problem structure may help reduce the complexity of the problem. Try to deduce something about the solution to the problem that might make the problem easier to solve. This may work for special cases of the problem.

3. **Identify problem type and select solver.** Based on your formulation, identify which type of problem it is and which solver is best to use for that type of problem. For instance, Gurobi can handle some convex quadratic problems, but not all. `Ipopt`is a more general solver, but may be slower due to the types of algorithms that it uses.

4. **Add bounds on the variables.** Many solvers perform much better if they are provided bounds to the variables. This is because it reduces the search region where the variables live. Adding good bounds could be the difference in the solver finding an optimal solution and not finding any solution at all.

5. **Warm start.** If you know good possible solutions to the problem (or even just a feasible solution), you can help the solver by telling it this solution. This will reduce the amount of work the solver needs to do. In JUMP this can be done by using the

command *setvalue(x,[2 4 6])*, where here it sets the value of vector x to $[2\ 4\ 6]$. It may be necessary to specify values for all variables in the problem for it to start at.

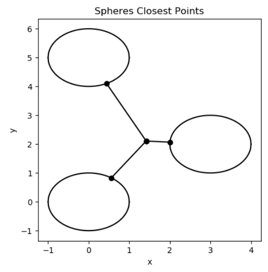
6. **Rescaling variables.** It sometimes is useful to have all variables on the same rough scale. For instance, if minimizing $x^2 + 100^2 y^2$, it may be useful to define a new variable $\bar{y} = 100y$ and instead minimize $x^2 + \bar{y}^2$.
7. **Provide derivatives.** Working out gradient and hessian information by hand can save the solver time. Particularly when these are sparse (many zeros). These can often be provided directly to the solver.

See `http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=4982C26EC5F25564BCC239FD3785E2D3?doi=10.1.1.210.3547&rep=rep1&type=pdf` for many other helpful tips on using `Ipopt`.

0.7.1 Minimum distance to circles

The problem we will consider here is: Given n circles, find a center point that minimizes the sum of the distances to all of the circles.

Minimize distance to circles:
Given circles described by center points (a_i, b_i) and radius r_i for $i = 1, \dots, n$, find a point $c = (c_x, c_y)$ that minimizes the sum of the distances to the circles.



Minimize distance to circles - Model attempt #1:
Non-convex
Let (x_i, y_i) be a point in circle i. Let w_i be the distance from (x_i, y_i) to c. Then we can model the problem as follows:

$$\begin{array}{lll} \min & \sum_{i=1}^3 w_i & \text{Sum of distances} \\ \text{s.t.} & \sqrt{(x_i - a_i)^2 + (y_i - b_i)^2} = r, \quad i = 1, \dots, n & (x_i, y_i) \text{ is in circle } i \\ & \sqrt{(x_i - c_x)^2 + (y_i - c_y)^2} = w_i \quad i = 1, \dots, n & w_i \text{ is distance from } (x_i, y_i) \text{ to } c \end{array} \tag{0.7.1}$$

This model has several issues:

1. If the center c lies inside one of the circles, then the constraint $\sqrt{(x_i - a_i)^2 + (y_i - b_i)^2} = r$ may not be valid. This is because the optimal choice for (x_i, y_i) in this case would be inside the circle, that is, satisfying $\sqrt{(x_i - a_i)^2 + (y_i - b_i)^2} \leq r$.

2. This model is **nonconvex**. In particular the equality constraints make the problem nonconvex.

Fortunately, we can relax the problem to make it convex an still model the correct solution. In particular, consider the constraint

$$\sqrt{(x_i - c_x)^2 + (y_i - c_y)^2} = w_i.$$

Since we are minimizing $\sum w_i$, it is equivalent to have the constraint

$$\sqrt{(x_i - c_x)^2 + (y_i - c_y)^2} \leq w_i.$$

This is equivalent because any optimal solution make w_i the smallest it can, and hence will meet that constraint at equality.

What is great about this change, it that it makes the constraint **convex!**. To see this we can write $f(z) = \|z\|_2^2$, $z = (x_i - c_x, y_i - x_y)$. Since $f(z)$ is convex and the transformation into variables x_i, c_x, y_i, c_y is linear, we have that $f(x_i - c_x, y_i - x_y)$ is convex. Then since $-w_i$ is linear, we have that

$$f(x_i - c_x, y_i - x_y) - w_i$$

is a convex function. Thus, the constraint

$$f(x_i - c_x, y_i - x_y) - w_i \leq 0$$

is a convex constraint.

This brings us to our second model.

Minimize distance to circles - Model attempt #2:
Convex
Let (x_i, y_i) be a point in circle i. Let w_i be the distance from (x_i, y_i) to c. Then we can model the problem as follows:

$$\begin{array}{lll} \min & \sum_{i=1}^3 w_i & \text{Sum of distances} \\ \text{s.t.} & \sqrt{(x_i - a_i)^2 + (y_i - b_i)^2} \leq r, \quad i = 1, \dots, n & (x_i, y_i) \text{ is in circle } i \\ & \sqrt{(x_i - c_x)^2 + (y_i - c_y)^2} \leq w_i \quad i = 1, \dots, n & w_i \text{ is distance from } (x_i, y_i) \text{ to } c \end{array} \tag{0.7.2}$$

Lastly, we would like to make this model better for a solver. For this we will

1. Add bounds on all the variables

2. Change format of non-linear inequalities

Minimize distance to circles - Model attempt #3:
Convex
Let (x_i, y_i) be a point in circle i. Let w_i be the distance from (x_i, y_i) to c. Then we can model the problem as follows:

$$\begin{array}{lll} \min & \sum_{i=1}^3 w_i & \text{Sum of distances} \\ \text{s.t.} & (x_i - a_i)^2 + (y_i - b_i)^2 \le r^2, \quad i = 1, \dots, n & (x_i, y_i) \text{ is in circle } i \\ & (x_i - c_x)^2 + (y_i - c_y)^2 \le w_i^2 \quad i = 1, \dots, n & w_i \text{ is distance from } (x_i, y_i) \text{ to } c \\ & 0 \le w_i \le u_i & \\ & a_i - r \le x_i \le a_i + r & \\ & b_i - r \le y_i \le b_i + r & \end{array} \tag{0.7.3}$$

Example: Minimize distance to circles [Code: **??**]
Here we minimize the distance of three circles of radius 1 centered at $(0,0)$, $(3,2)$, and $(0,5)$. Note: The bounds on the variables here are not chosen optimally.

$$\begin{array}{rl} \min & w_1 + w_2 + w_3 \\ \text{Subject to} & (x_1 - 0)^2 + (y_1 - 0)^2 \le 1 \\ & (x_2 - 3)^2 + (y_2 - 2)^2) \le 1 \\ & (x_3 - 0)^2 + (y_3 - 5)^2) \le 1 \\ & (x_1 - c_x)^2 + (y_1 - c_y)^2) \le w_1^2 \\ & (x_2 - c_x)^2 + (y_2 - c_y)^2 \le w_2^2 \\ & (x_3 - c_x)^2 + (y_3 - c_y)^2 \le w_3^2 \\ & -1 \le x_i \le 10 \quad \forall i \in \{1,2,3\} \\ & -1 \le y_i \le 10 \quad \forall i \in \{1,2,3\} \\ & 0 \le w_i \le 40 \quad \forall i \in \{1,2,3\} \\ & -1 \le c_x \le 10 \\ & -1 \le c_y \le 10 \end{array}$$

0.8 Machine Learning

There are two main fields of machine learning:

- Supervised Machine Learning,
- Unsupervised Machine Learning.

Supervised machine learning is composed of *Regression* and *Classification*. This area is thought of as being given labeled data that you are then trying to understand the trends of this labeled data.

Unsupervised machine learning is where you are given unlabeled data and then need to decide how to label this data. For instance, how can you optimally partition the people in a room into 5 groups that share the most commonalities?

0.9 Machine Learning - Supervised Learning - Regression

See the video lecture information.

0.10 Machine learning - Supervised Learning - Classification

The problem of data *classification* begins with *data* and *labels*. The goal is *classification* of future data based on sample data that you have by constructing a function to understand future data.

Goal: *Classification - create a function $f(x)$ that takes in a data point x and outputs the correct label.*

These functions can take many forms. In binary classification, the label set is $\{+1, -1\}$, and we want to correctly (as often as we can) determine the correct label for a future data point.

There are many ways to determine such a function $f(x)$. In the next section, we will learn about SVM that determines the function by computing a hyperplane that separates the data labeled $+1$ from the data labeled -1.

Later, we will learn about *neural networks* that describe much more complicated functions.

Another method is to create a *decision tree*. These are typically more interpretable functions (neural networks are often a bit mysterious) and thus sometimes preferred in settings where the classification should be easily understood, such as a medical diagnosis. We will not discuss this method here since it fits less well with the theme of nonlinear programming.

0.10.1 Python SGD implementation and video

https://github.com/llSourcell/Classifying_Data_Using_a_Support_Vector_Machine/blob/master/support_vector_machine_lesson.ipynb