COMP 7180: Quantitative Methods for Data Analytics and Artificial Intelligence

Lecture 5: An Introduction to Differentiation and Optimization in AI and ML

Lecturer:

Dr. LIU Yang

J. Duchi, Introduction to Convex Optimization for Machine Learning, University of California, Berkeley, 2009.
M. P. Deisenroth, A. A. Faisal, and C. S. Ong, Mathematics for Machine Learning, 2019.

What is Optimization ?

- Finding the minimizer/maximizer of a function subject to constraints:

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & f_0(x) \\
\text{s.t.} \quad & f_i(x) \leq 0, \; i = \{1, \ldots, k\} \\
& h_j(x) = 0, \; j = \{1, \ldots, l\}
\end{aligned}
$$

- Example: Stock market
  - To minimize the variance of return subject to getting at least $50K

Why Do We Care About Optimization ?

Optimization is at the heart of many AI and machine learning algorithms!

- Maximum likelihood estimation:

$$\underset{\theta}{\text{maximize}} \quad \sum_{i=1}^{n} \log p_\theta(x_i)$$

- Linear regression:

$$\underset{w}{\text{minimize}} \quad \|Xw - y\|^2$$

- Support vector machine:

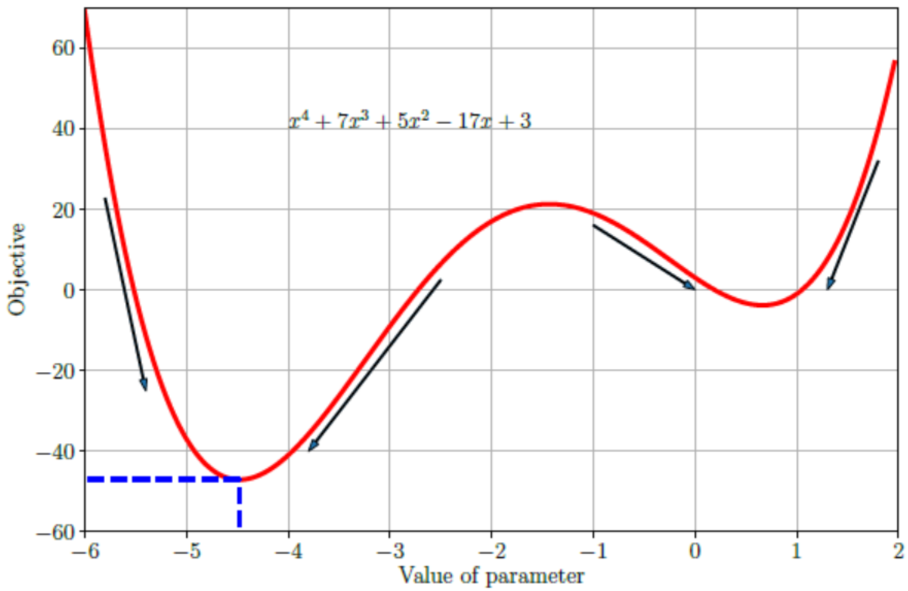
$$\underset{w}{\text{minimize}} \quad \|w\|^2 + C\sum_{i=1}^{n} \xi_i \quad \text{s.t.} \quad \xi_i \geq 1 - y_i x_i^T w, \xi_i \geq 0$$

Example

$$\text{Minimize } f(x) = x^4 + 7x^3 + 5x^2 - 17x + 3$$

$$\text{Gradient: } \frac{df(x)}{dx} = 4x^3 + 21x^2 + 10x - 17 = 0$$

$$\text{Candidates: } x_1 = -4.5, x_2 = -1.4, x_3 = 0.7$$



Differentiation

Limit

In the study of differentiation and calculus, we are interested in what happens to the value of a function as the independent variable *gets very close* to a particular value.

Example:

$$f(x) = \frac{x^2 - 2x - 3}{x - 3}$$

What is the value of the function as x **approaches** 3?

Solution:

Limit: More Exercises

Find the limit $\lim_{x \to \infty} \left(\frac{5 - 3x}{6x + 1} \right)$

$$\lim_{x \to 0} \frac{\sqrt{x+1} - 1}{x}$$

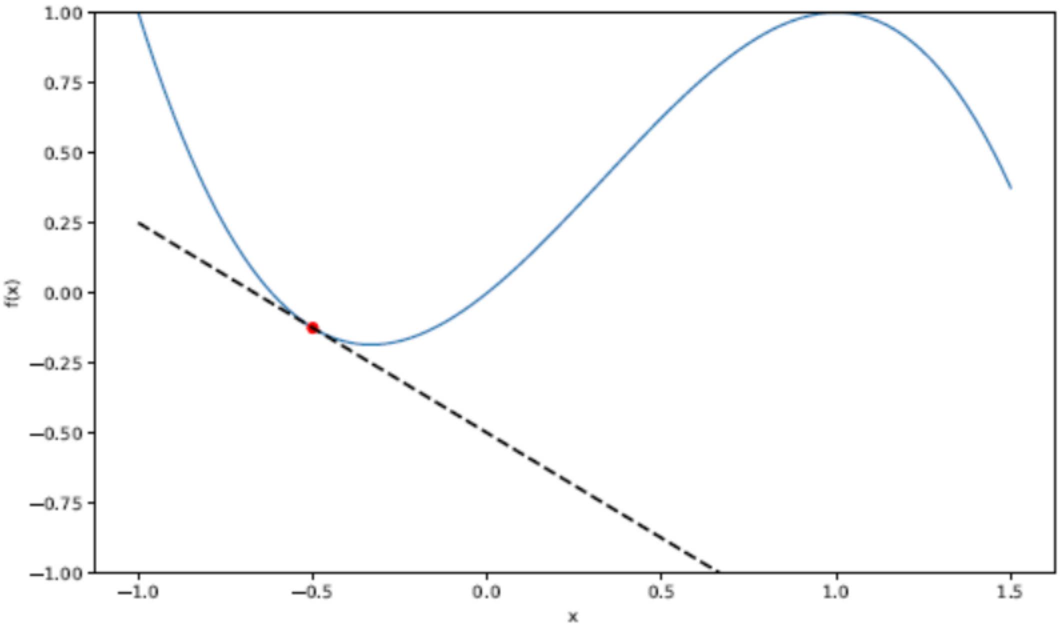
$$\lim_{x \to 64} \frac{\sqrt[3]{x} - 4}{\sqrt{x} - 8}$$

Scalar Differentiation $f: \mathrm{R} \rightarrow \mathrm{R}$

- Derivative is defined as the limit of the difference quotient

$$f'(x) = \frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- Slope of the secant line through $f(x)$ and $f(x+h)$



Examples of Scalar Differentiation

$$
\begin{aligned}
f(x) &= x^n & \quad\Longrightarrow\quad & f'(x) = nx^{n-1} \\
f(x) &= \sin(x) & & f'(x) = \cos(x) \\
f(x) &= \tanh(x) & & f'(x) = 1 - \tanh^2(x) \\
f(x) &= \exp(x) & & f'(x) = \exp(x) \\
f(x) &= \log(x) & & f'(x) = \frac{1}{x}
\end{aligned}
$$

Rules of Scalar Differentiation

- Sum Rule

$$\left(f(x)+g(x)\right)' = f'(x) + g'(x) = \frac{df}{dx} + \frac{dg}{dx}$$

Rules of Scalar Differentiation

- Product Rule

$$(f(x)g(x))' = f'(x)g(x) + f(x)g'(x) = \frac{df}{dx}g(x) + f(x)\frac{dg}{dx}$$

Rules of Scalar Differentiation

- Chain Rule

$$(g \circ f)'(x) = \left(g(f(x))\right)' = g'(f(x))f'(x) = \frac{dg}{df}\frac{df}{dx}$$

L'Hôpital’s Rule

For functions *f* and *g* which are differentiable on an open interval **I** except possibly at a point *c* contained in **I**, if

$\lim_{x\to c} f(x) = \lim_{x\to c} g(x) = 0$ or $\pm\infty$, and $g'(x) \neq 0$ for all x in I with $x \neq c$,

and $\lim_{x\to c} \frac{f'(x)}{g'(x)}$ exists, then

$$\lim_{x\to c} \frac{f(x)}{g(x)} = \lim_{x\to c} \frac{f'(x)}{g'(x)}$$

Examples for L'Hôpital's Rule

$f(x) = \sin(x)$
$g(x) = -0.5x$

What is the limit of the function $h(x) = f(x)/g(x)$ when x → 0?

We cannot directly calculate it because both $f(x)$ and $g(x)$ are approaching 0. However, we can calculate it using the L'Hôpital's Rule:

$h(0) = f(0)/g(0) = f'(0)/g'(0) = -2$

Multivariate Differentiation $f : \mathrm{R}^N \to \mathrm{R}$

- Given

$$y = f(\boldsymbol{x}), \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} \in \mathbb{R}^N$$

- Partial derivative (change one coordinate at a time):

$$\frac{\partial f}{\partial x_i} = \lim_{h \to 0} \frac{f(x_1, \ldots, x_{i-1}, x_i + h, x_{i+1}, \ldots, x_N) - f(\boldsymbol{x})}{h}$$

- Jacobian vector (gradient) collects all partial derivatives:

$$\frac{df}{d\boldsymbol{x}} = \left[\frac{\partial f}{\partial x_1} \quad \cdots \quad \frac{\partial f}{x_N}\right]^T \in \mathbb{R}^N$$

- ***Note: This is a vector, not a scalar***

Exercise of Multivariate Differentiation

$$f : \mathbb{R}^2 \to \mathbb{R}$$
$$f(x_1, x_2) = x_1^2 x_2 + x_1 x_2^3 \in \mathbb{R}$$

- Partial derivative:

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 2x_1 x_2 + x_2^3$$
$$\frac{\partial f(x_1, x_2)}{\partial x_2} = x_1^2 + 3x_1 x_2^2$$

- Gradient:

$$\frac{df}{d\boldsymbol{x}} = \begin{bmatrix} \frac{\partial f(x_1,x_2)}{\partial x_1} & \frac{\partial f(x_1,x_2)}{\partial x_2} \end{bmatrix}^T = \begin{bmatrix} 2x_1 x_2 + x_2^3 & x_1^2 + 3x_1 x_2^2 \end{bmatrix}^T \in \mathbb{R}^2$$

Rules of Multivariate Differentiation

- Sum Rule

$$\frac{\partial}{\partial \boldsymbol{x}}\left(f(\boldsymbol{x})+g(\boldsymbol{x})\right)=\frac{\partial f}{\partial \boldsymbol{x}}+\frac{\partial g}{\partial \boldsymbol{x}}$$

- Product Rule

$$\frac{\partial}{\partial \boldsymbol{x}}\left(f(\boldsymbol{x})g(\boldsymbol{x})\right)=\frac{\partial f}{\partial \boldsymbol{x}}g(\boldsymbol{x})+f(\boldsymbol{x})\frac{\partial g}{\partial \boldsymbol{x}}$$

- Chain Rule

$$\frac{\partial}{\partial \boldsymbol{x}}(g \circ f)(\boldsymbol{x})=\frac{\partial}{\partial \boldsymbol{x}}\left(g(f(\boldsymbol{x}))\right)=\frac{\partial g}{\partial f}\frac{\partial f}{\partial \boldsymbol{x}}$$

Differentiation in Vector Field $f : R^N \rightarrow R^M$

- Given

$$y = f(x) \in \mathbb{R}^M, \quad x \in \mathbb{R}^N$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix} = \begin{bmatrix} f_1(x) \\ \vdots \\ f_M(x) \end{bmatrix} = \begin{bmatrix} f_1(x_1, \ldots, x_N) \\ \vdots \\ f_M(x_1, \ldots, x_N) \end{bmatrix}$$

- Jacobian matrix (collection of all partial derivatives):

$$\begin{bmatrix} \frac{dy_1}{dx} \\ \vdots \\ \frac{dy_M}{dx} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \vdots & & \vdots \\ \frac{\partial f_M}{\partial x_1} & \cdots & \frac{\partial f_M}{\partial x_N} \end{bmatrix} \in \mathbb{R}^{M \times N}$$

Example of Differentiation in Vector Field

- Given

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}, \qquad \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^M, \quad \boldsymbol{A} \in \mathbb{R}^{M \times N}, \quad \boldsymbol{x} \in \mathbb{R}^N$$

- Compute the gradient $\frac{df}{dx}$.

Chain Rule of Differentiation in Vector Field

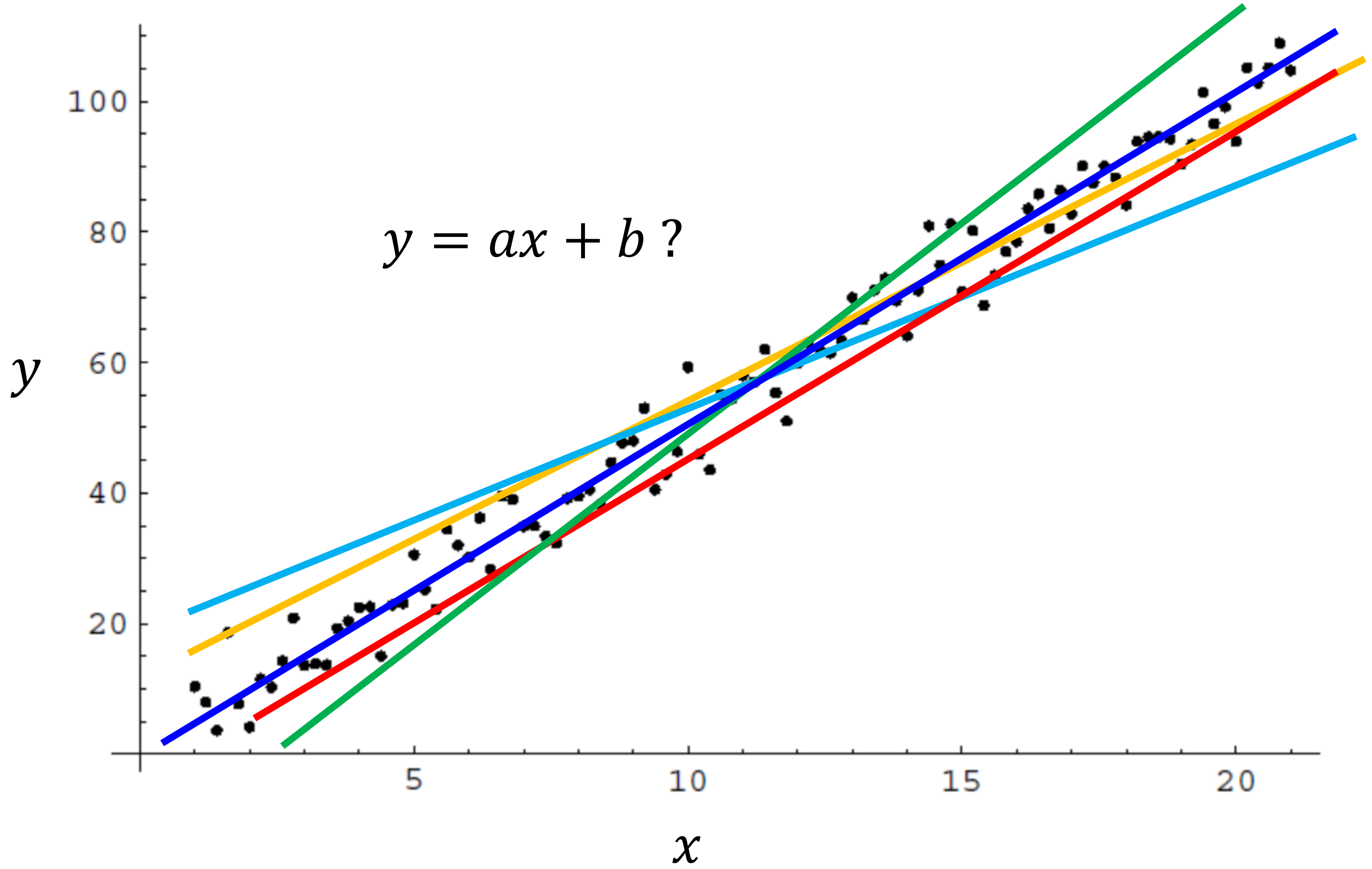
$$\frac{\partial}{\partial \boldsymbol{x}}(g \circ f)(\boldsymbol{x}) = \frac{\partial}{\partial \boldsymbol{x}}\big(g(f(\boldsymbol{x}))\big) = \frac{\partial g}{\partial f}\frac{\partial f}{\partial \boldsymbol{x}}$$

- Consider $f : \mathbb{R}^2 \to \mathbb{R}, \quad \boldsymbol{x} : \mathbb{R} \to \mathbb{R}^2$

$$f(\boldsymbol{x}) = f(x_1, x_2) = x_1^2 + 2x_2\,,$$

$$\boldsymbol{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} \sin(t) \\ \cos(t) \end{bmatrix}$$

Regression via Least Squares Method



Regression via Least Squares Method

Given data $\{(x_1, y_1), \ldots, (x_N, y_N)\}$, we may define the error associated to saying $y = ax + b$ by

$$E(a,b) \;=\; \sum_{n=1}^{N} \left(y_n - (ax_n + b)\right)^2.$$

The goal is to find values of a and b that minimize the error. In multivariable calculus we learn that this requires us to find the values of (a, b) such that

$$\frac{\partial E}{\partial a} \;=\; 0, \quad \frac{\partial E}{\partial b} \;=\; 0.$$

Differentiating $E(a, b)$ yields

$$\begin{aligned}
\frac{\partial E}{\partial a} \;&=\; \sum_{n=1}^{N} 2\left(y_n - (ax_n + b)\right) \cdot (-x_n) \\
\frac{\partial E}{\partial b} \;&=\; \sum_{n=1}^{N} 2\left(y_n - (ax_n + b)\right) \cdot 1.
\end{aligned}$$

Regression via Least Squares Method

Setting $\partial E/\partial a = \partial E/\partial b = 0$ (and dividing by 2) yields

$$\begin{aligned}\sum_{n=1}^{N}(y_n - (ax_n + b)) \cdot x_n &= 0\\ \sum_{n=1}^{N}(y_n - (ax_n + b)) &= 0.\end{aligned}$$

We may rewrite these equations as

$$\begin{aligned}\left(\sum_{n=1}^{N} x_n^2\right) a + \left(\sum_{n=1}^{N} x_n\right) b &= \sum_{n=1}^{N} x_n y_n\\ \left(\sum_{n=1}^{N} x_n\right) a + \left(\sum_{n=1}^{N} 1\right) b &= \sum_{n=1}^{N} y_n.\end{aligned}$$

Regression via Least Squares Method

We have obtained that the values of a and b which minimize the error satisfy the following matrix equation:

$$\begin{pmatrix} \sum_{n=1}^N x_n^2 & \sum_{n=1}^N x_n \\ \sum_{n=1}^N x_n & \sum_{n=1}^N 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^N x_n y_n \\ \sum_{n=1}^N y_n \end{pmatrix}.$$

We will show the matrix is invertible, which implies

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^N x_n^2 & \sum_{n=1}^N x_n \\ \sum_{n=1}^N x_n & \sum_{n=1}^N 1 \end{pmatrix}^{-1} \begin{pmatrix} \sum_{n=1}^N x_n y_n \\ \sum_{n=1}^N y_n \end{pmatrix}.$$

Gradient Descent

Optimization Using Gradient Descent

- Consider the problem of solving for the minimum of a loss function:

$$\min_x f(x)$$

where $x \in R^d$ and $f: R^d \to R$ is the objective function.

- Assume that:
  - Function f is differentiable ☺
  - We are unable to analytically find a solution in closed form ☹

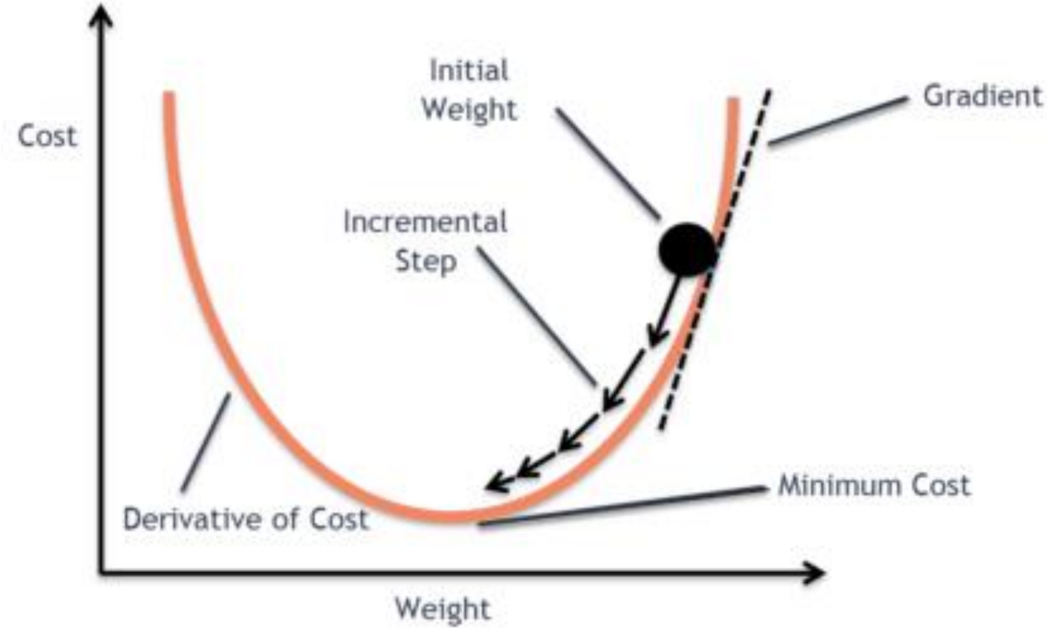
What should we do now?

Gradient descent !

Optimization Using Gradient Descent

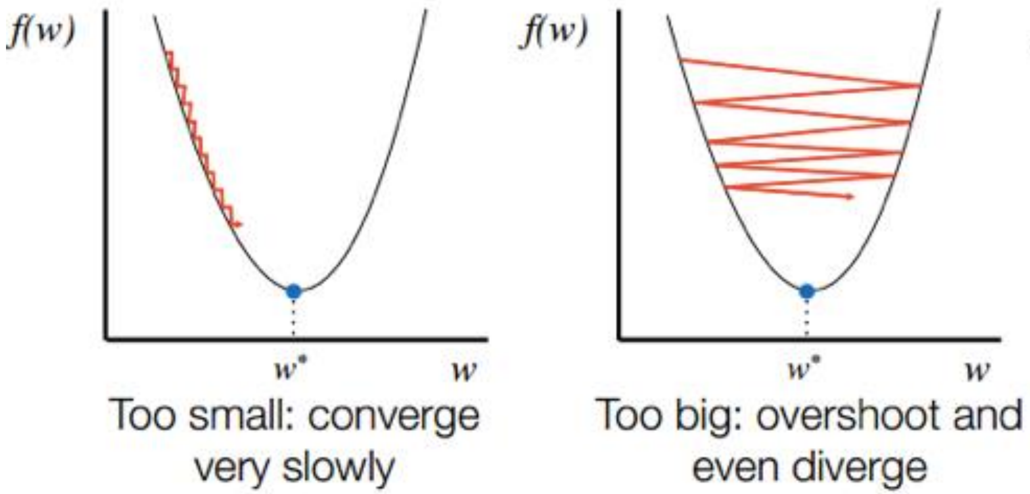
- Gradient descent is one of the most widely used optimization algorithms in machine learning
- It is a first-order iterative optimization algorithm: take steps proportional to the negative of the gradient of the function at the current point

$$x_{t+1} = x_t - \alpha \frac{df}{dx_t}$$



Step Size (Learning Rate)

- Choosing a good step-size is important in gradient descent



- Heuristics
  - When the function value increases after a gradient step, the step-size was too large. Undo the step and decrease the step-size.
  - When the function value decreases the step could have been larger. Try to increase the step-size.
- Algorithm: exact (or backtracking) line search: $\min_{\alpha \in R} f(x_t - \alpha \nabla x_t)$

Gradient Descent Algorithm

- Given a start point x_0
- For $t = 0, \ldots, T$

1. Calculate gradient: $\nabla x_t \leftarrow \frac{df}{dx_t}$
2. Line search: choose the step size (learning rate) α via exact (or backtracking) line search: $\min_{\alpha \in R} f(x_t - \alpha \nabla x_t)$
3. Update x: $x_{t+1} \leftarrow x_t - \alpha \nabla x_t$
4. $t \leftarrow t + 1$
5. Repeat steps 1-4 until convergence: $|\nabla x_t| < \epsilon$ or $|x_{t+1} - x_t| < \epsilon$

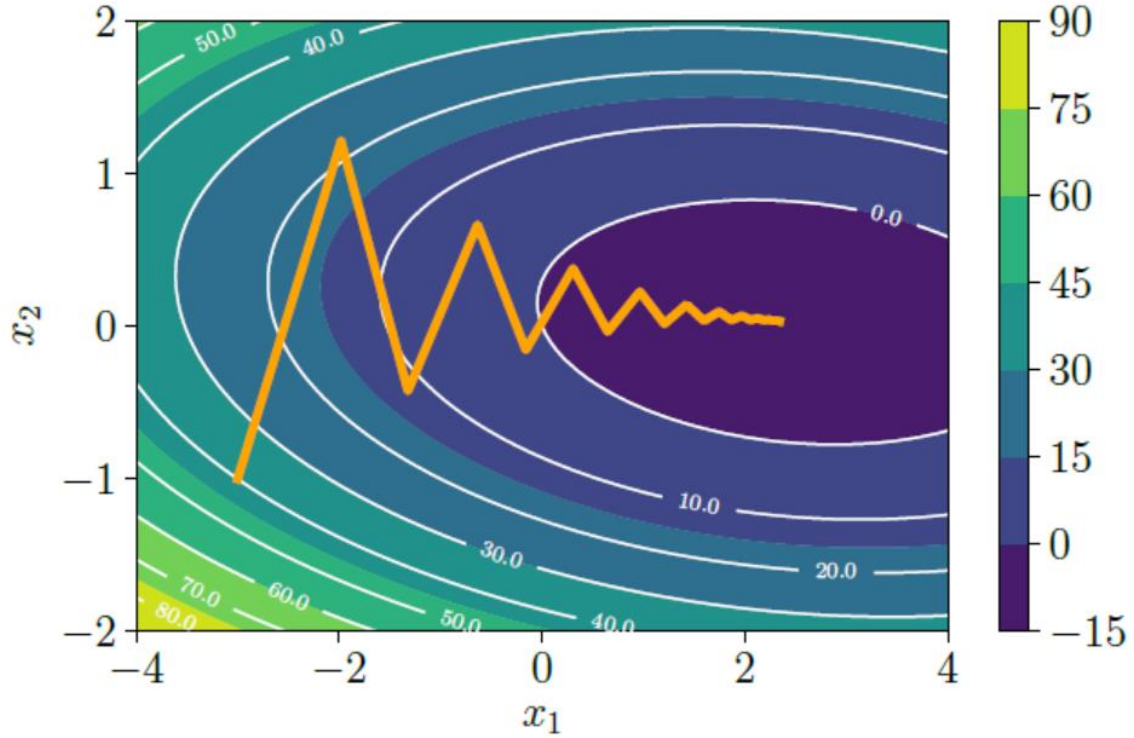
Example

$$\min_{x_1,x_2} f(x_1,x_2) = x_1^2 + 10x_2^2 + x_1x_2 + 5x_1 + 3x_2$$

$$= \frac{1}{2}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 5 \\ 3 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

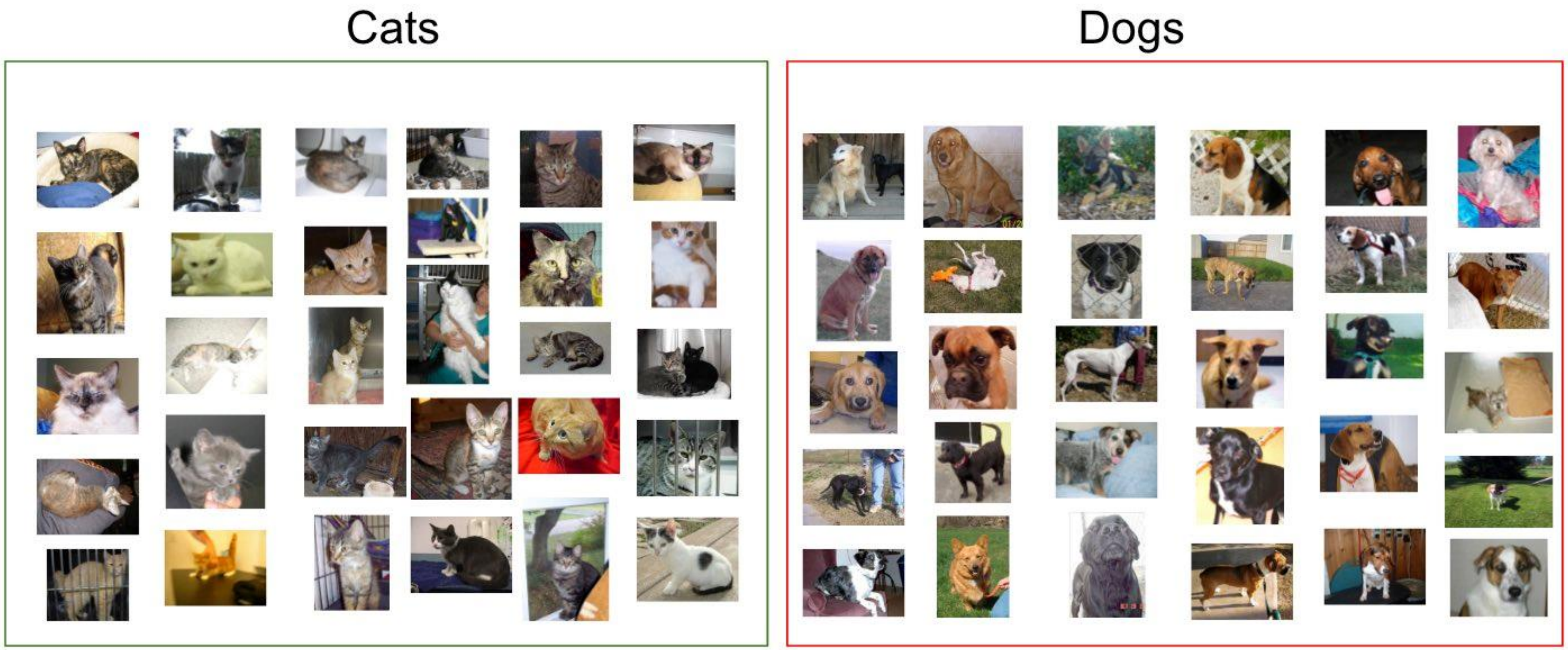
$$\nabla f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right)$$

$$= \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 5 \\ 3 \end{bmatrix}$$



Perceptron Learning

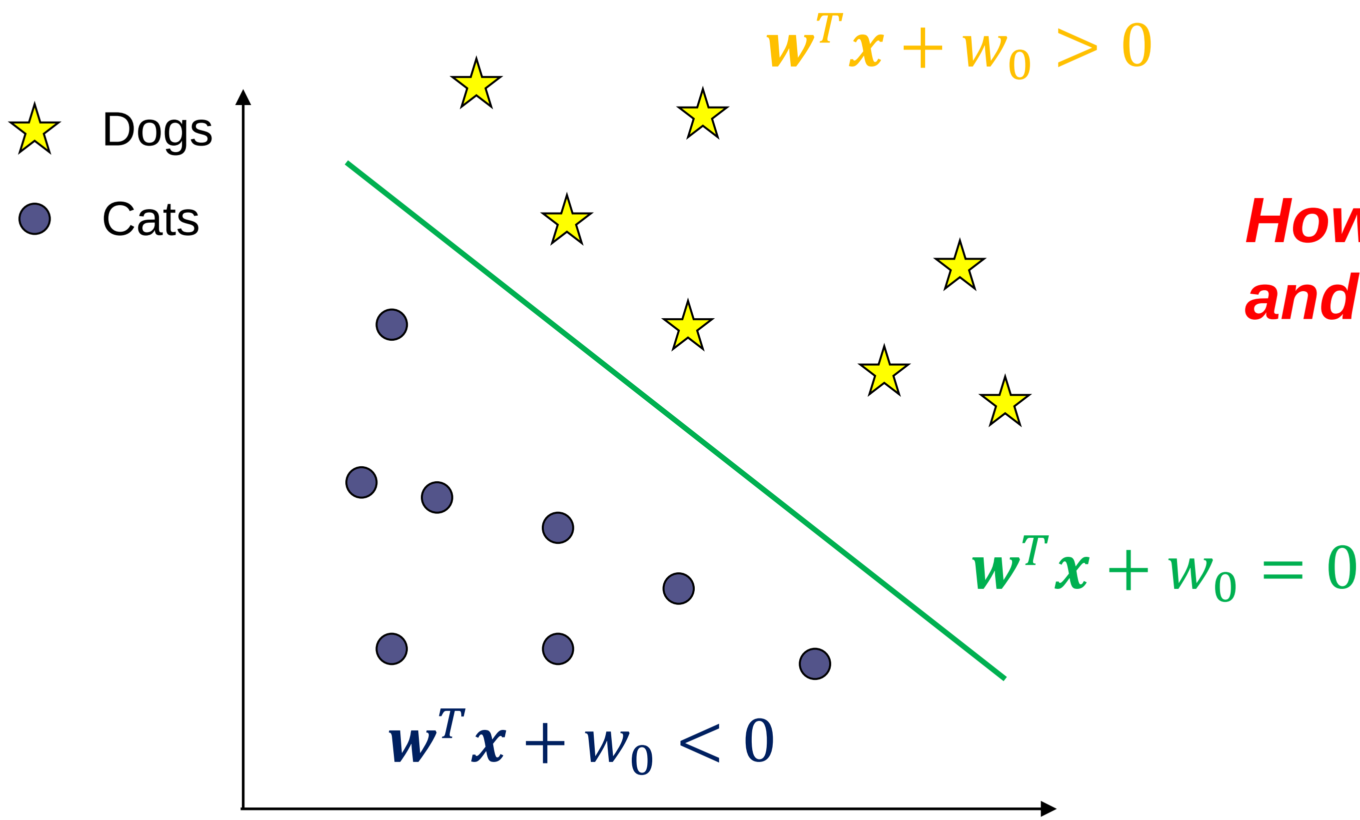
A Machine Learning Problem: Classification



Sample of cats & dogs images from Kaggle Dataset

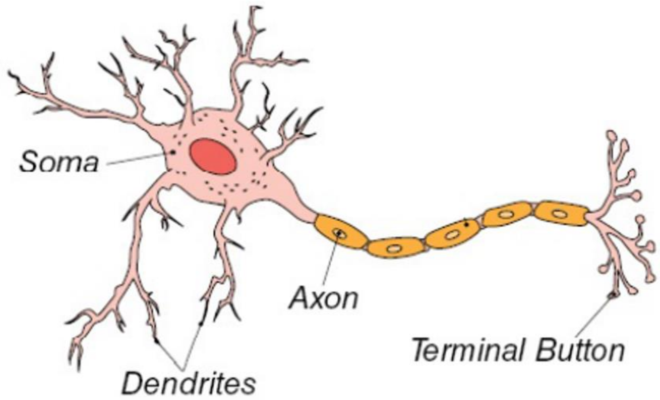
https://www.kaggle.com/c/dogs-vs-cats

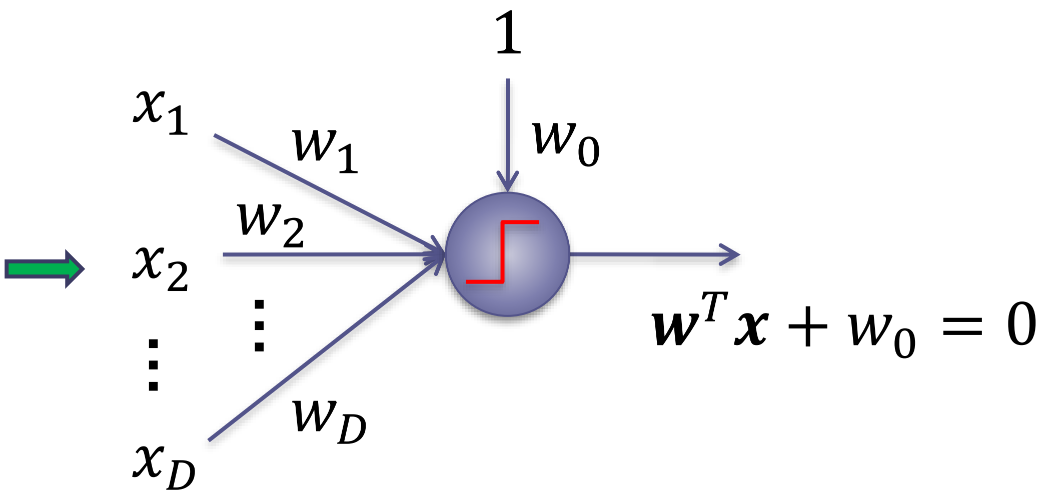
Geometric Interpretation of Classification Problem



How to find w and w_0 ??

Motivation of Perceptron Learning





- Neurons
  - accept information from multiple inputs
  - transmit information to other neurons
- Multiply inputs by weights along edges
- Apply some function to the set of inputs at each node

Mathematical Definition

- Given training data $\{(x_i, y_i): 1 \le i \le n\}$
- Hypothesis $f_w(x) = w^T x$
  - $y = +1$ if $w^T x > 0$
  - $y = -1$ if $w^T x < 0$
- Prediction: $y = \text{sign}(f_w(x)) = \text{sign}(w^T x)$

- Goal: minimize classification error

Perceptron Learning Algorithm

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.

Intuition: correct the current mistake

- If mistake on a positive example

$$w_{t+1}^T x = (w_t + x)^T x = w_t^T x + x^T x = w_t^T x + 1$$

- If mistake on a negative example

$$w_{t+1}^T x = (w_t - x)^T x = w_t^T x - x^T x = w_t^T x - 1$$

Perceptron Theorem

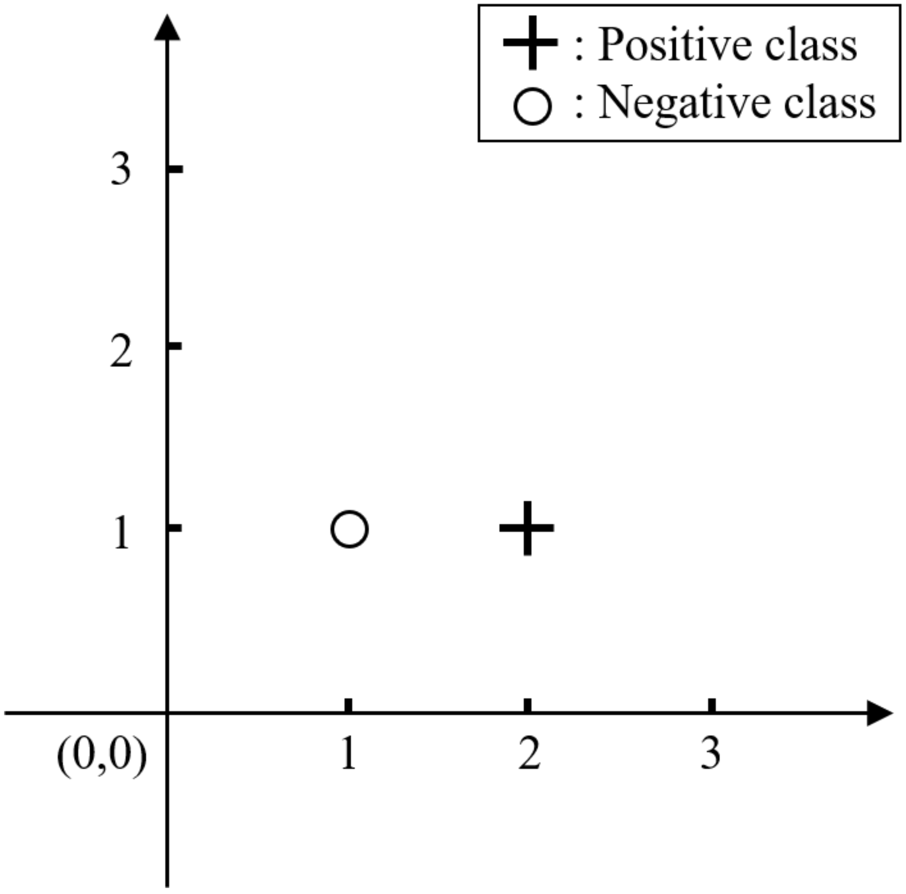
- Suppose there exists w^* that correctly classifies $\{(x_i, y_i)\}$
- all x_i and w^* have length 1, so the minimum distance of any example to the decision boundary is

$$\gamma = \min_i |(w^*)^T x_i|$$

- Then Perceptron makes at most $\left(\frac{1}{\gamma}\right)^2$ mistakes

<u>Can you proof this theorem?</u>

An Example of Perceptron Learning



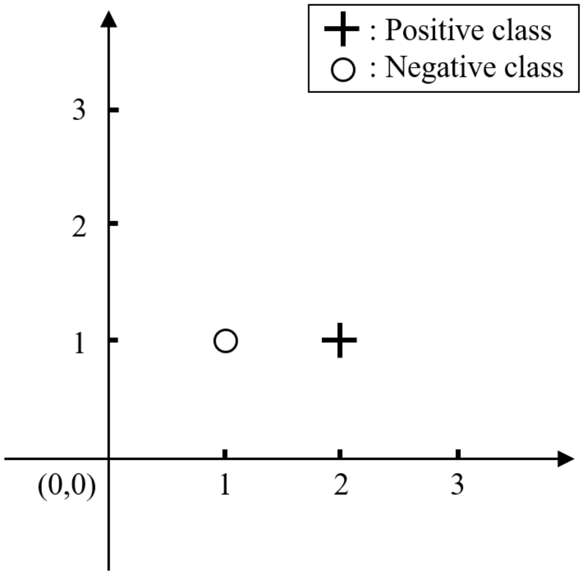
Given two classes of data points: for the positive class, there is one data point: (2,1); for the negative class, there is one data point: (1,1).

Assume that we start with the all-zero weight vector, use the Perceptron Learning Algorithm to find out the decision hyperplane (equation) that can correctly classify all the data points.

An Example of Perceptron Learning

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.



x+: (2,1,1) ; x–: (1,1,1)

y = sign($w^T x$), where w = [w_1, w_2, w_0] = [0,0,0]

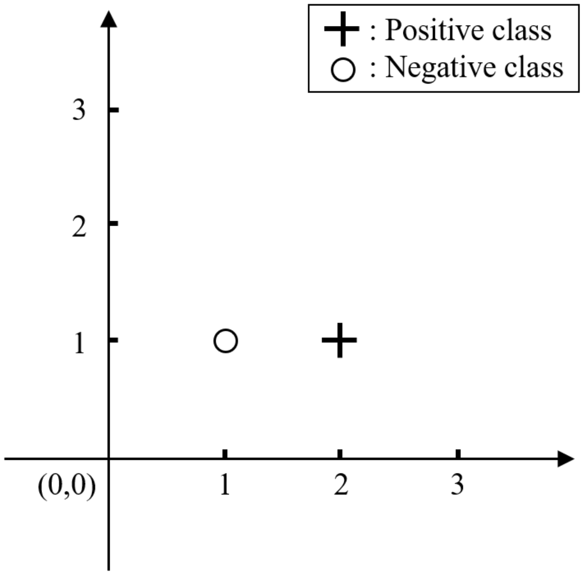
For x+, we have y = 0, not correctly classified. So we need to update the weights:

w_new = w_old + x = [0,0,0] + [2,1,1] = [2,1,1].

An Example of Perceptron Learning

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.



x+: (2,1,1) ; x–: (1,1,1)

y = sign($w^T x$), where w = [w_1, w_2, w_0] = [2,1,1]

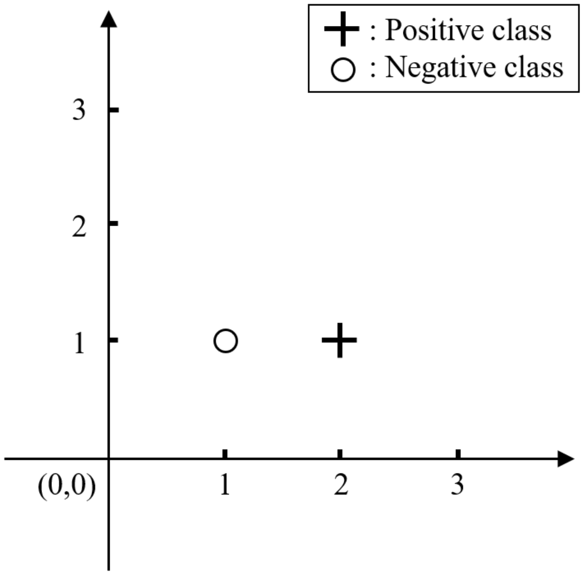
For x+, we have y = 1, correctly classified; but for x–, we have y =1 not correctly classified. So we need to update the weights:

w_new = w_old – x = [2,1,1] - [1,1,1] = [1,0,0].

An Example of Perceptron Learning

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.



x+: (2,1,1) ; x–: (1,1,1)

y = sign($w^T x$), where w = [w_1, w_2, w_0] = [1,0,0]

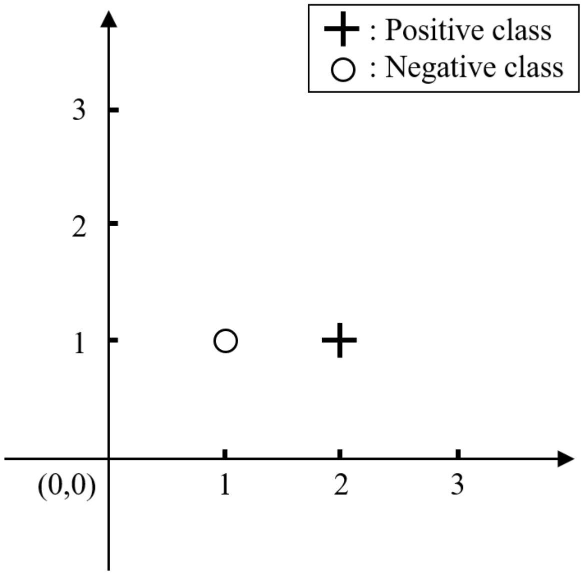
For x+, we have y = 1, correctly classified; but for x–, we still have y =1 not correctly classified. So we need to update the weights:

w_new = w_old – x = [1,0,0] - [1,1,1] = [0,-1,-1].

An Example of Perceptron Learning

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.



x+: (2,1,1) ; x–: (1,1,1)

y = sign($w^T x$), where w = [w_1, w_2, w_0] = [0,-1,-1]

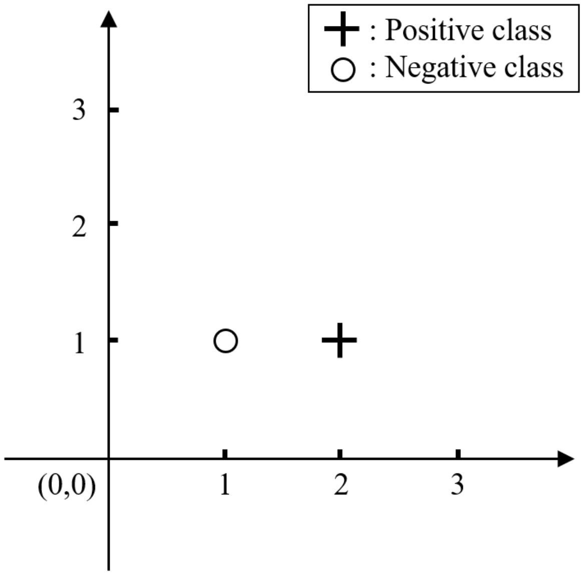
For x+, we have y = -1, not correctly classified. So we need to update the weights:

w_new = w_old + x = [0,-1,-1] + [2,1,1] = [2,0,0].

An Example of Perceptron Learning

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.



x+: (2,1,1) ; x–: (1,1,1)

y = sign($w^T x$), where w = [w_1, w_2, w_0] = [2,0,0]

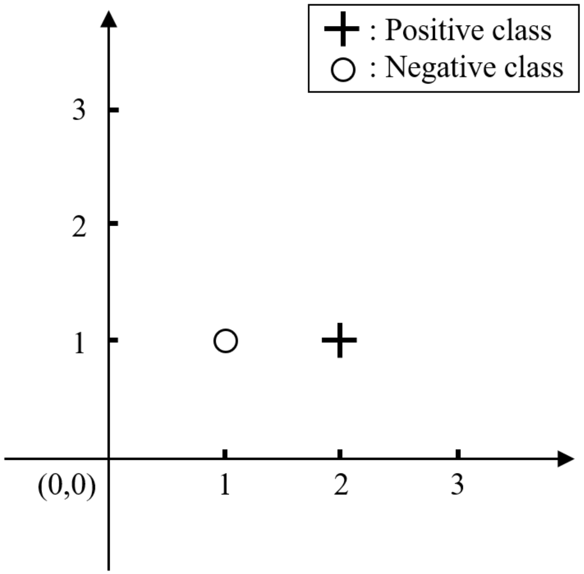
For x+, we have y = 1, correctly classified; but for x–, we have y = 1 not correctly classified. So we need to update the weights:

w_new = w_old – x = [2,0,0] – [1,1,1] = [1,-1,-1].

An Example of Perceptron Learning

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.



x+: (2,1,1) ; x–: (1,1,1)

y = sign($w^T x$), where w = [w_1, w_2, w_0] = [1,-1,-1]

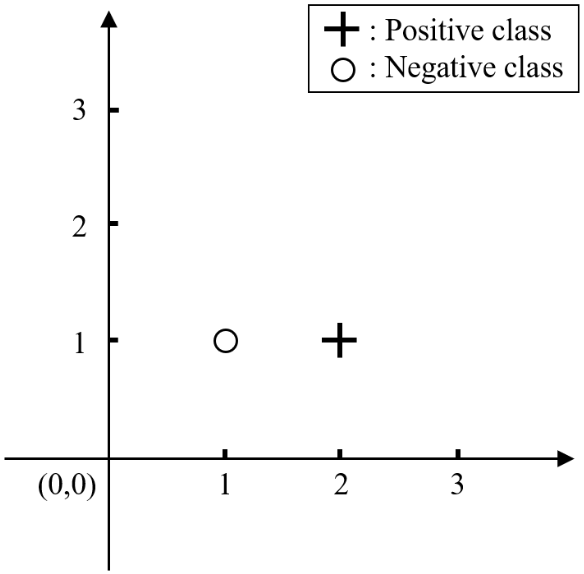
For x+, we have y = 0, not correctly classified. So we need to update the weights:

w_new = w_old + x = [1,-1,-1] + [2,1,1] = [3,0,0].

An Example of Perceptron Learning

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.



x+: (2,1,1) ; x–: (1,1,1)

y = sign($w^T x$), where w = [w_1, w_2, w_0] = [3,0,0]

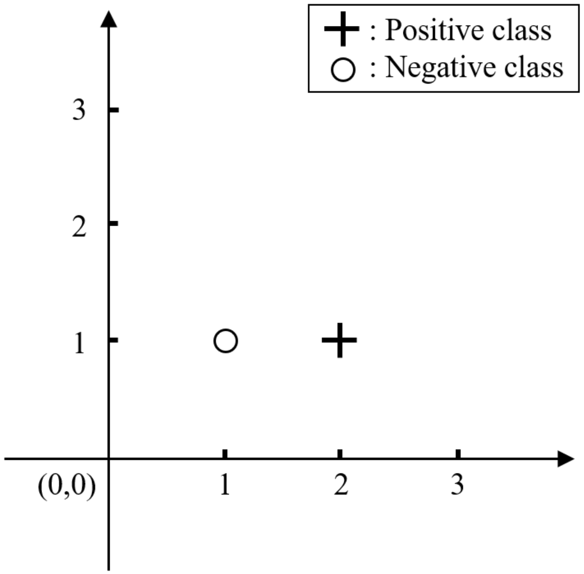
For x+, we have y = 1, correctly classified, but for x–, we have y = 1 not correctly classified. So we need to update the weights:

w_new = w_old – x = [3,0,0] – [1,1,1] = [2,-1,-1].

An Example of Perceptron Learning

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.



x+: (2,1,1) ; x–: (1,1,1)

y = sign($w^T x$), where w = [w_1, w_2, w_0] = [2,-1,-1]

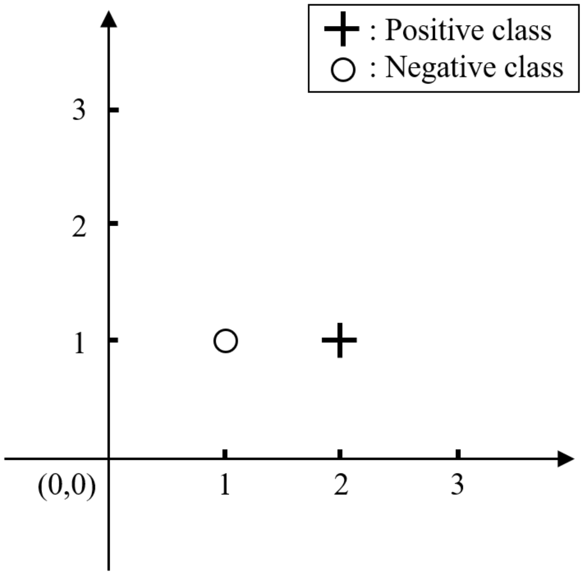
For x+, we have y = 1, correctly classified, but for x–, we have y = 0 not correctly classified. So we need to update the weights:

w_new = w_old – x = [2,-1,-1] – [1,1,1] = [1,-2,-2].

An Example of Perceptron Learning

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.



x+: (2,1,1) ; x–: (1,1,1)

y = sign($w^T x$), where w = [w_1, w_2, w_0] = [1,-2,-2]

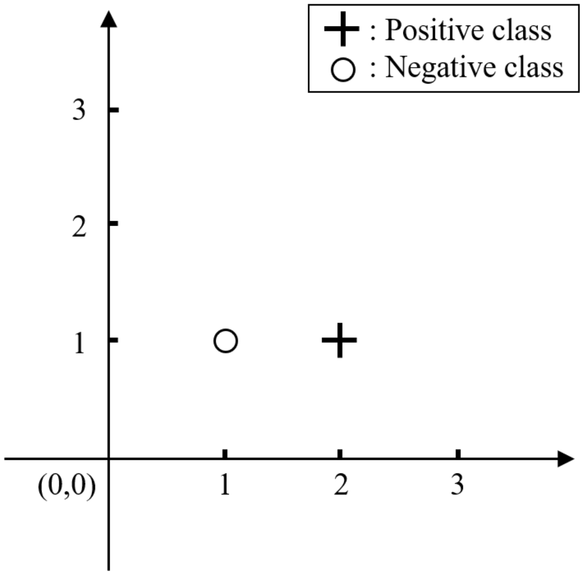
For x+, we have y = -1, not correctly classified. So we need to update the weights:

w_new = w_old + x = [1,-2,-2] + [2,1,1] = [3,-1,-1].

An Example of Perceptron Learning

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.



x+: (2,1,1) ; x–: (1,1,1)

y = sign($w^T x$), where w = [w_1, w_2, w_0] = [3,-1,-1]

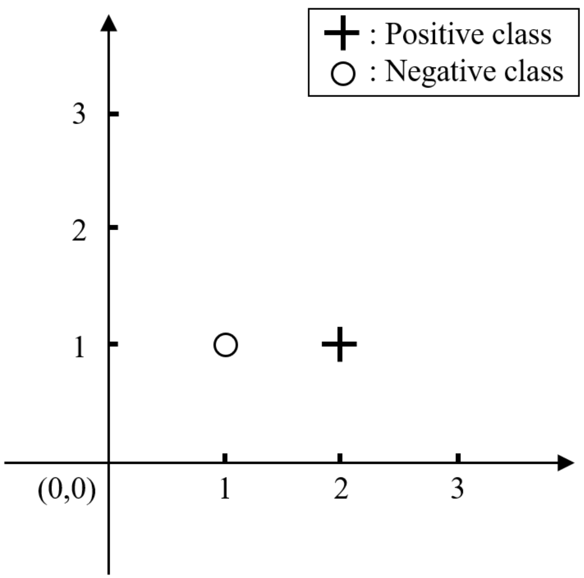
For x+, we have y = 1, correctly classified, but for x–, we have y = 1 not correctly classified. So we need to update the weights:

w_new = w_old – x = [3,-1,-1] – [1,1,1] = [2,-2,-2].

An Example of Perceptron Learning

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.



x+: (2,1,1) ; x–: (1,1,1)

y = sign($w^T x$), where w = [w_1, w_2, w_0] = [2,-2,-2]

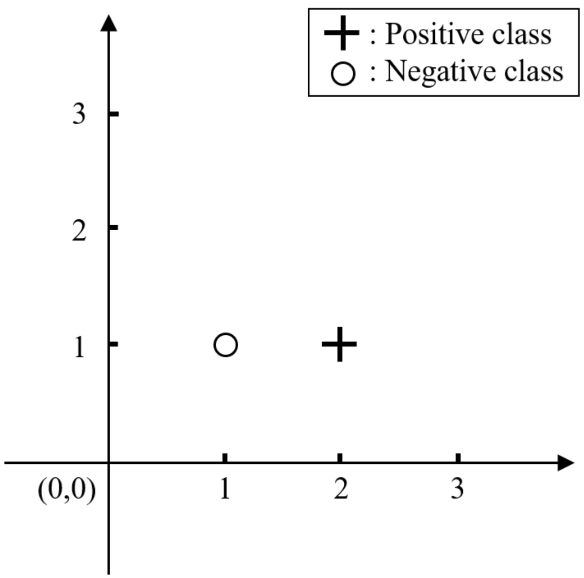
For x+, we have y = 0, not correctly classified. So we need to update the weights:

w_new = w_old + x = [2,-2,-2] + [2,1,1] = [4,-1,-1].

An Example of Perceptron Learning

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.



x+: (2,1,1) ; x–: (1,1,1)

y = sign($w^T x$), where w = [w_1, w_2, w_0] = [4,-1,-1]

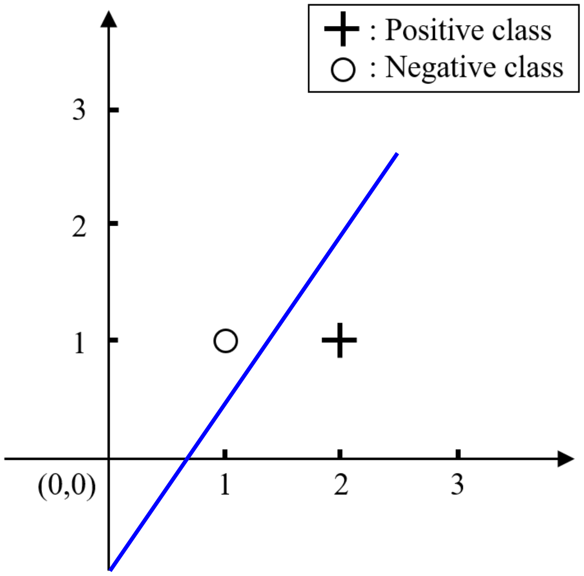
For x+, we have y = 1, correctly classified, but for x–, we have y = 1 not correctly classified. So we need to update the weights:

w_new = w_old – x = [4,-1,-1] – [1,1,1] = [3,-2,-2].

An Example of Perceptron Learning

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.



x+: (2,1,1) ; x–: (1,1,1)

y = sign($w^T x$), where w = [w_1, w_2, w_0] = [3,-2,-2]

For x+, we have y = 1, correctly classified, for x–, we have y = – 1 correctly classified.

Done!!

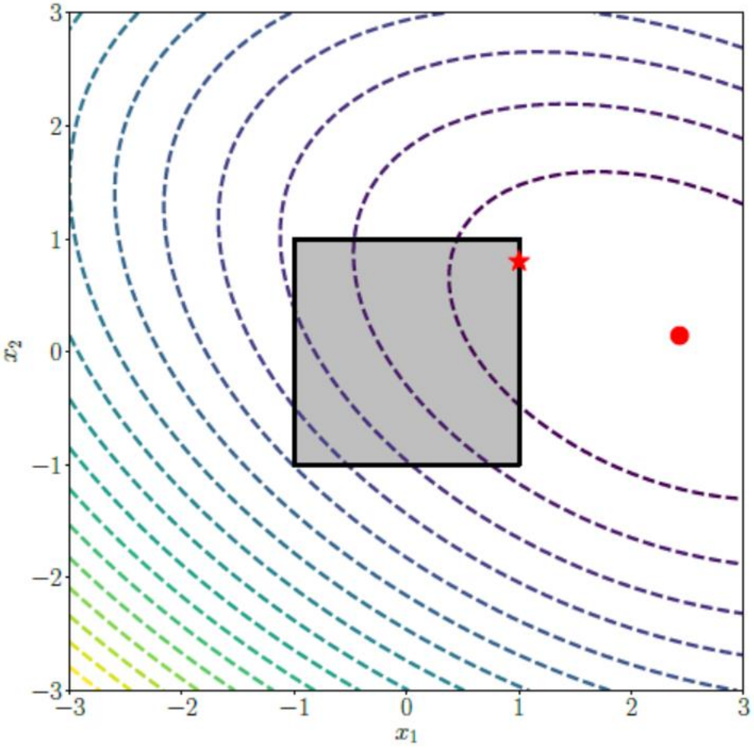
Convex Optimization

Constrained Optimization

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$

$$\text{subject to } g_i(\boldsymbol{x}) \leqslant 0 \quad \text{for all} \quad i = 1, \ldots, m$$

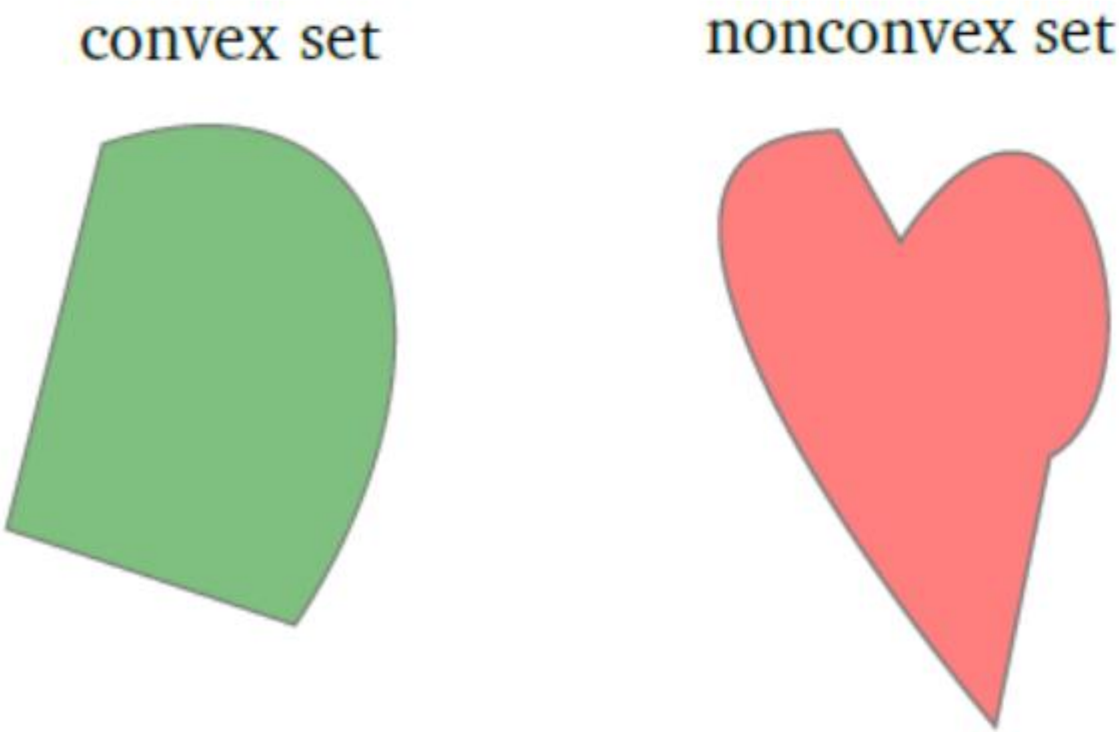
$$h_j(\boldsymbol{x}) = 0 \quad \text{for all} \quad j = 1, \ldots, n$$



Convex Set

Definition A set C is a *convex set* if for any $x, y \in C$ and for any scalar θ with $0 \leqslant \theta \leqslant 1$, we have

$$\theta x + (1-\theta)y \in C$$



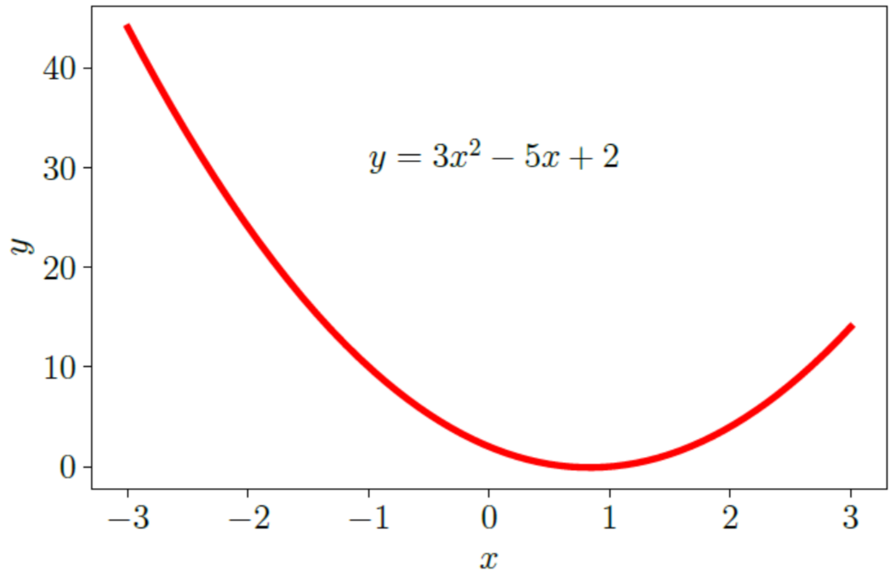
Convex Set

- Example: the solution set of linear equations $Ax = b$ is a convex set.

Convex Function

Definition Let function $f : \mathbb{R}^D \to \mathbb{R}$ be a function whose domain is a convex set. The function f is a *convex function* if for all $\boldsymbol{x}, \boldsymbol{y}$ in the domain of f, and for any scalar θ with $0 \leqslant \theta \leqslant 1$, we have

$$f(\theta \boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leqslant \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{y})$$

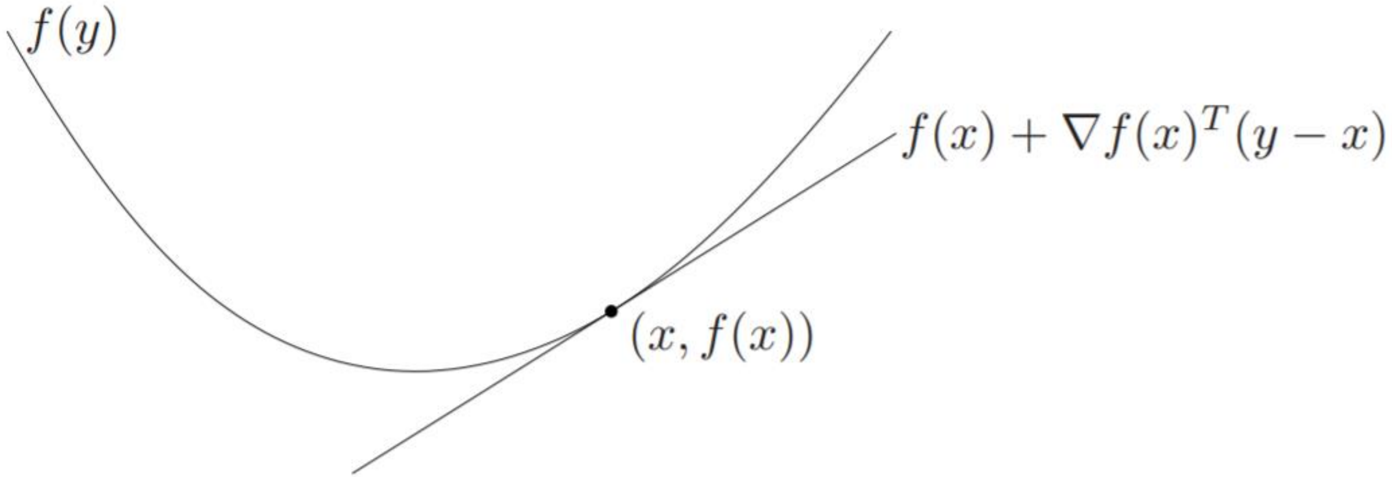


First Order Condition

Suppose f is differentiable (*i.e.*, its gradient ∇f exists at each point in **dom** f, which is open). Then f is convex if and only if **dom** f is convex and

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

holds for all $x,\ y \in \textbf{dom}\, f$.



Proof of First Order Condition ($n = 1$)

Second Order Condition

- For twice differentiable *f* with convex domain, *f* is convex if and only if the Hessian matrix $\nabla^2 f(x)$ is positive semi-definite. Here the Hessian matrix is defined as follows:

$$\nabla^2 f(x)_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}, \quad i, j = 1, \ldots, n$$

Proof of Second Order Condition ($n = 1$)

Example of convex function

- $f(x) = x^2$
- Definition:

$$f(\theta \boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leqslant \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{y})$$

- First order condition:

$$f(y) \geq f(x) + f'(x)(y-x)$$

- Second order condition:

$$f''(x) \geq 0$$

Convex Optimization Problem

In summary, a constrained optimization problem is called a *convex optimization problem* if

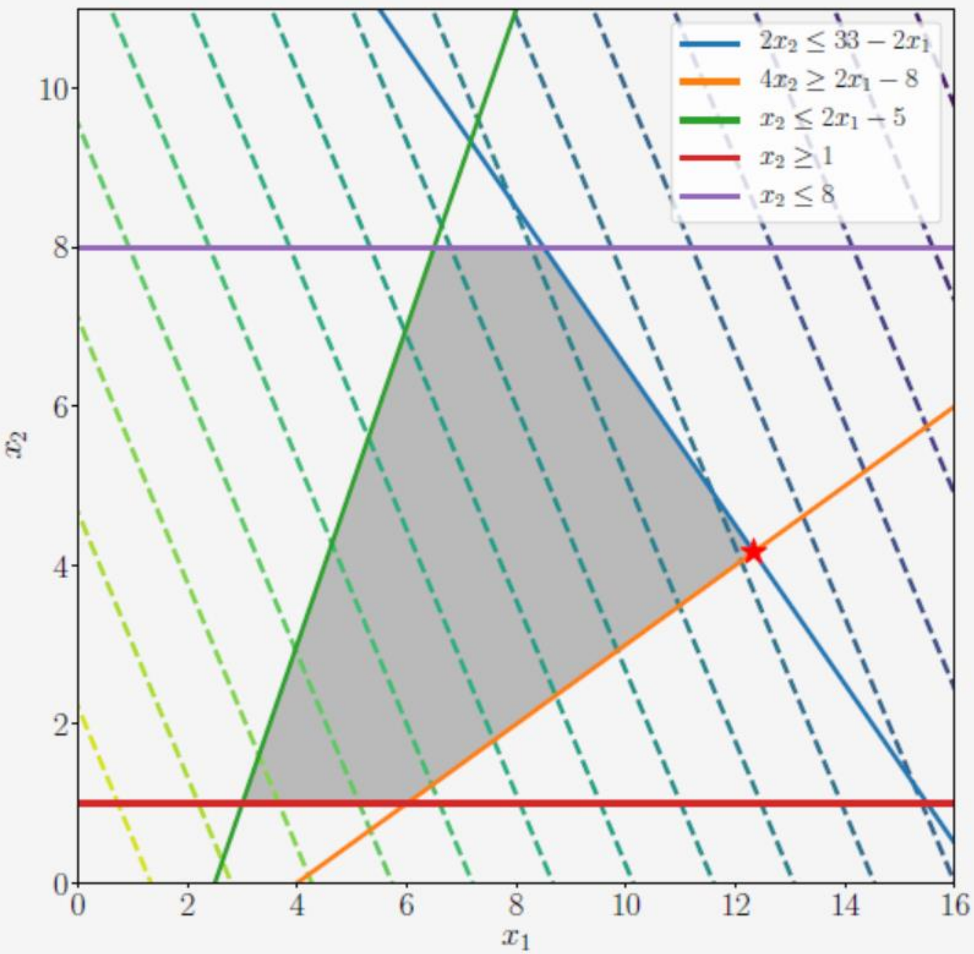
$$\begin{aligned} \min_{\boldsymbol{x}} & f(\boldsymbol{x}) \\ \text{subject to } & g_i(\boldsymbol{x}) \leqslant 0 \quad \text{for all} \quad i = 1, \ldots, m \\ & h_j(\boldsymbol{x}) = 0 \quad \text{for all} \quad j = 1, \ldots, n\,, \end{aligned}$$

where all functions $f(\boldsymbol{x})$ and $g_i(\boldsymbol{x})$ are convex functions, and all $h_j(\boldsymbol{x}) = 0$ are convex sets.

Example: Linear Programming

$$\min_{\boldsymbol{x}\in\mathbb{R}^2} \quad -\begin{bmatrix}5\\3\end{bmatrix}^\top \begin{bmatrix}x_1\\x_2\end{bmatrix}$$

$$\text{subject to} \quad \begin{bmatrix}2 & 2\\2 & -4\\-2 & 1\\0 & -1\\0 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} \leqslant \begin{bmatrix}33\\8\\5\\-1\\8\end{bmatrix}$$



Support Vector Machine

-We are given a set of n points (vectors) :
$x_1, x_2, \ldots\ldots x_n$ such that x_i is a vector of length m, and each belong to one of two classes we label them by "+1" and "-1".

-So our training set is:

$$(x_1, y_1), (x_2, y_2), \ldots.(x_n, y_n)$$

$$\forall i \quad x_i \in R^m, \; y_i \in \{+1, -1\}$$

So the decision function will be

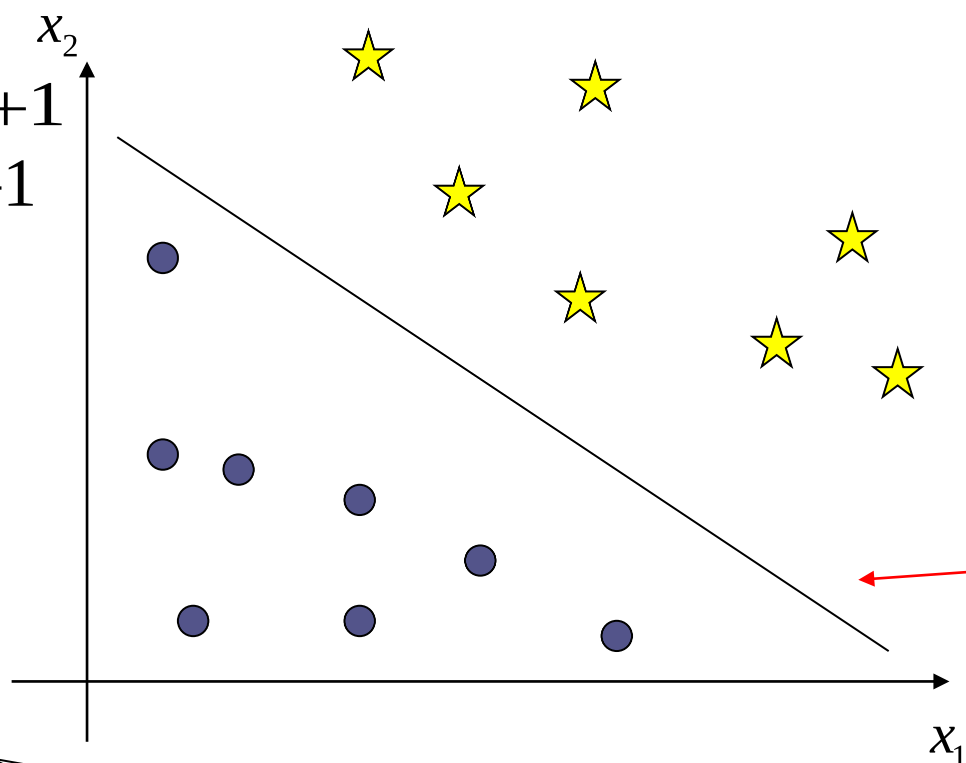
$$f(x) = sign(w \cdot x + b)$$

- We want to find a separating hyperplane $w \cdot x + b = 0$ that separates these points into the two classes. "The positives" (class "+1") and "The negatives" (class "-1"). (Assuming that they are linearly separable)

Separating Hyperplane

☆ $y_i = +1$

● $y_i = -1$



$$f(x) = sign(w \cdot x + b)$$

A separating hypreplane

$$w \cdot x + b = 0$$

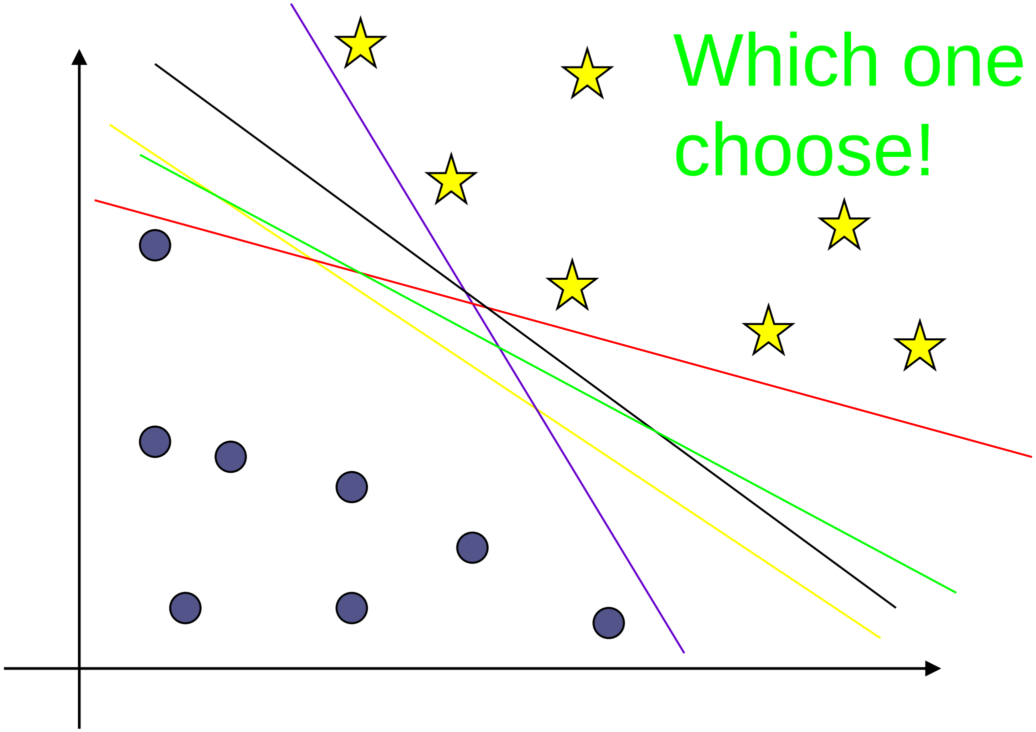
But there are many possibilities

for such hyperplanes !!

Separating Hyperplane

$y_i = +1$

$y_i = -1$



Which one should we choose!

Yes, There are many possible separating hyperplanes

It could be this one or this or this or maybe….!

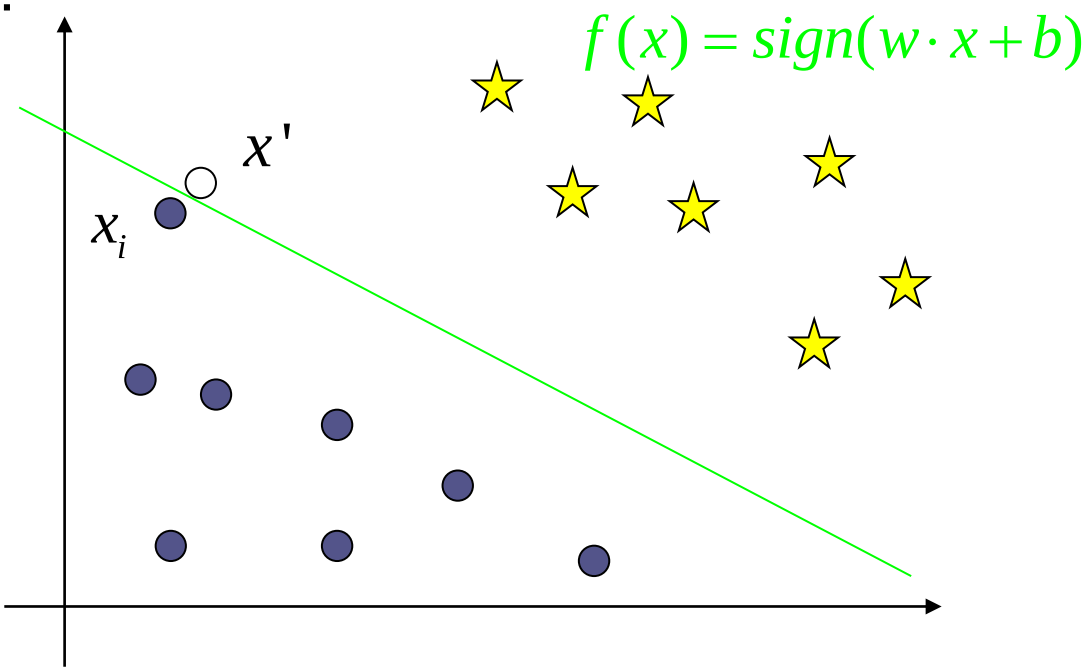
Choosing a separating hyperplane

-Suppose we choose the hyperplane (seen below) that is close to some sample x_i.

- Now suppose we have a new point x' that should be in class "-1" and is close to x_i. Using our classification function $f(x)$ this point is misclassified!

Poor generalization!

(Poor performance on unseen data)

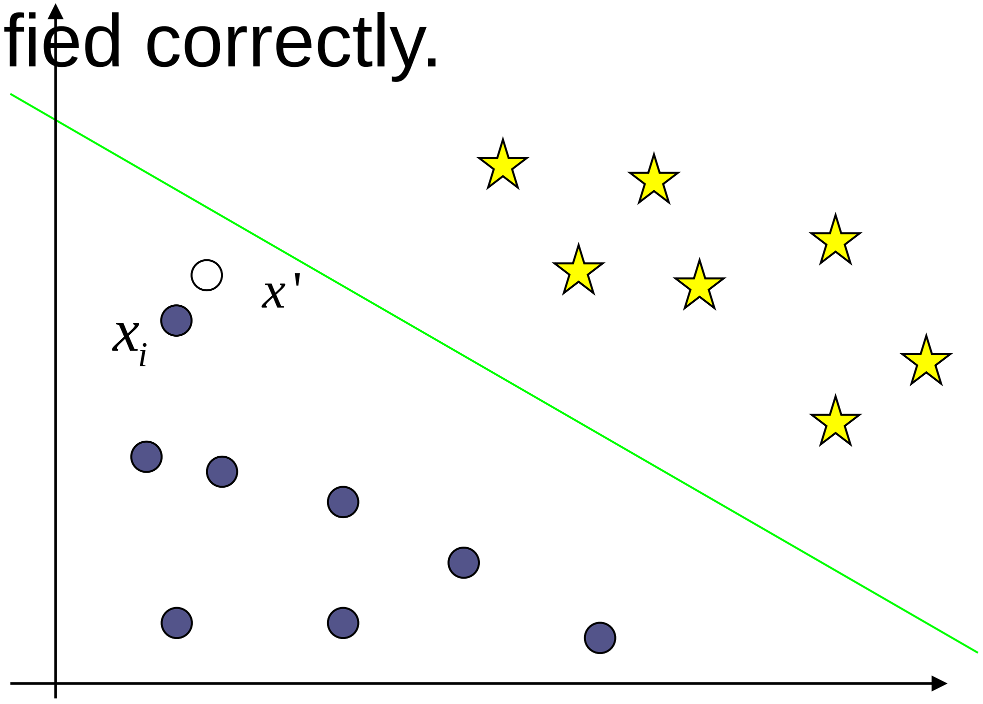


Choosing a separating hyperplane

-Hyperplane should be as far as possible from any sample point.

-This way a new data that is close to the old samples will be classified correctly.

Good generalization!



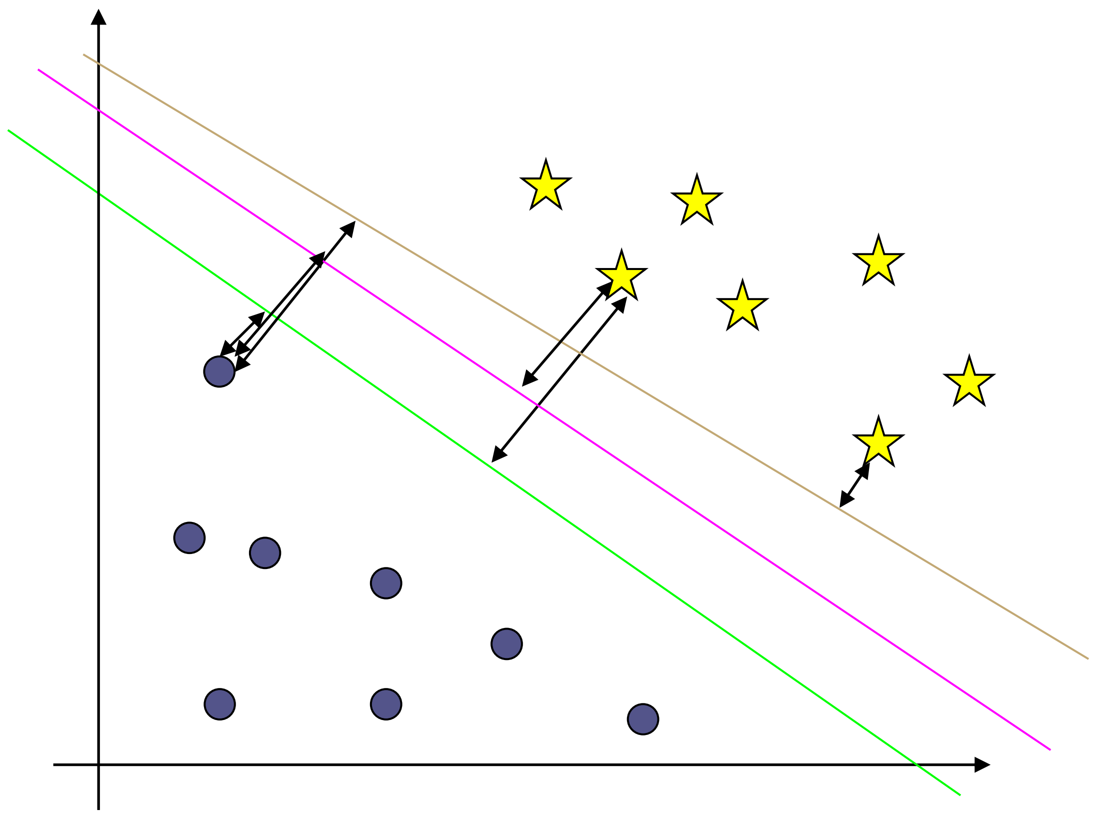
Choosing a separating hyperplane

-The SVM idea is to maximize the distance between
The hyperplane and the closest sample point.

In the optimal hyper-plane:

The distance to the closest negative point =

The distance to the closest positive point.

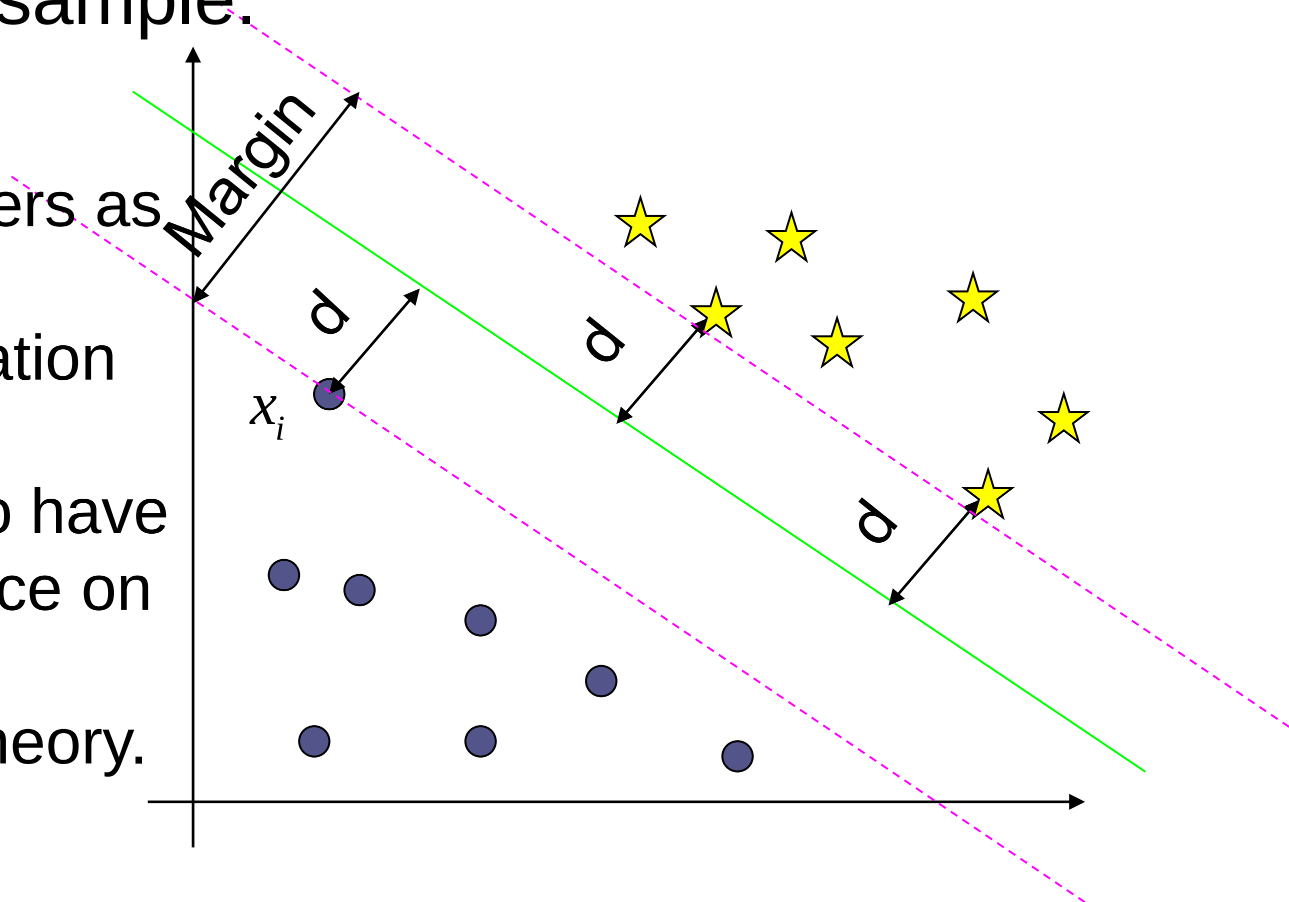


Support Vector Machine

SVM's goal is to maximize the Margin which is twice the distance "d" between the separating hyperplane and the closest sample.

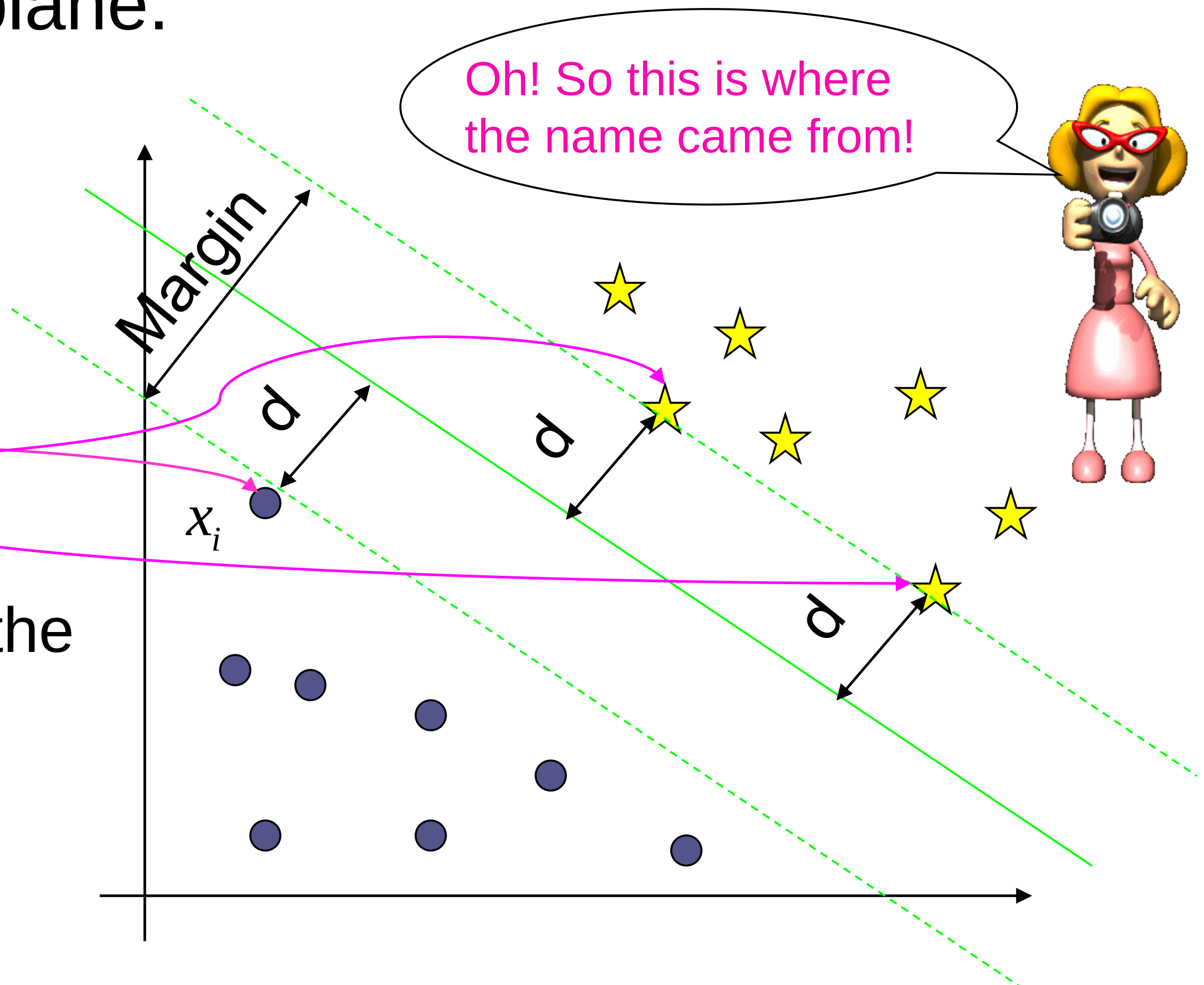
Why it is the best?

- -Robust to outliners as we saw and thus strong generalization ability.
- -It proved itself to have better performance on test data in both practice and in theory.



Support Vector Machine

Support vectors are the samples closest to the separating hyperplane.



These are Support Vectors

We will see later that the Optimal hyperplane is completely defined by the support vectors.

The Optimization Problem

$$\max \; d \quad \Longrightarrow \quad \max \left(\frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|} \right)$$

$$\Longrightarrow \quad \min \left(\frac{\|\mathbf{w}\|}{\mathbf{w}^T \mathbf{x}_i + b} \right) \quad \Longrightarrow \quad \begin{array}{c} \min \;\; \|\mathbf{w}\| \\ \text{s.t.} \;\; |\, \mathbf{w}^T \mathbf{x}_i + b \,| \geq 1 \end{array}$$

$$\Longrightarrow \quad \boxed{\begin{array}{c} \min \;\; \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \;\; y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \end{array}}$$