Caltech

Machine Learning & Data Mining

CMS/CS/CNS/EE 155

Lecture 2:

Perceptron & Stochastic Gradient Descent

Reminder: Gradescope & Piazza

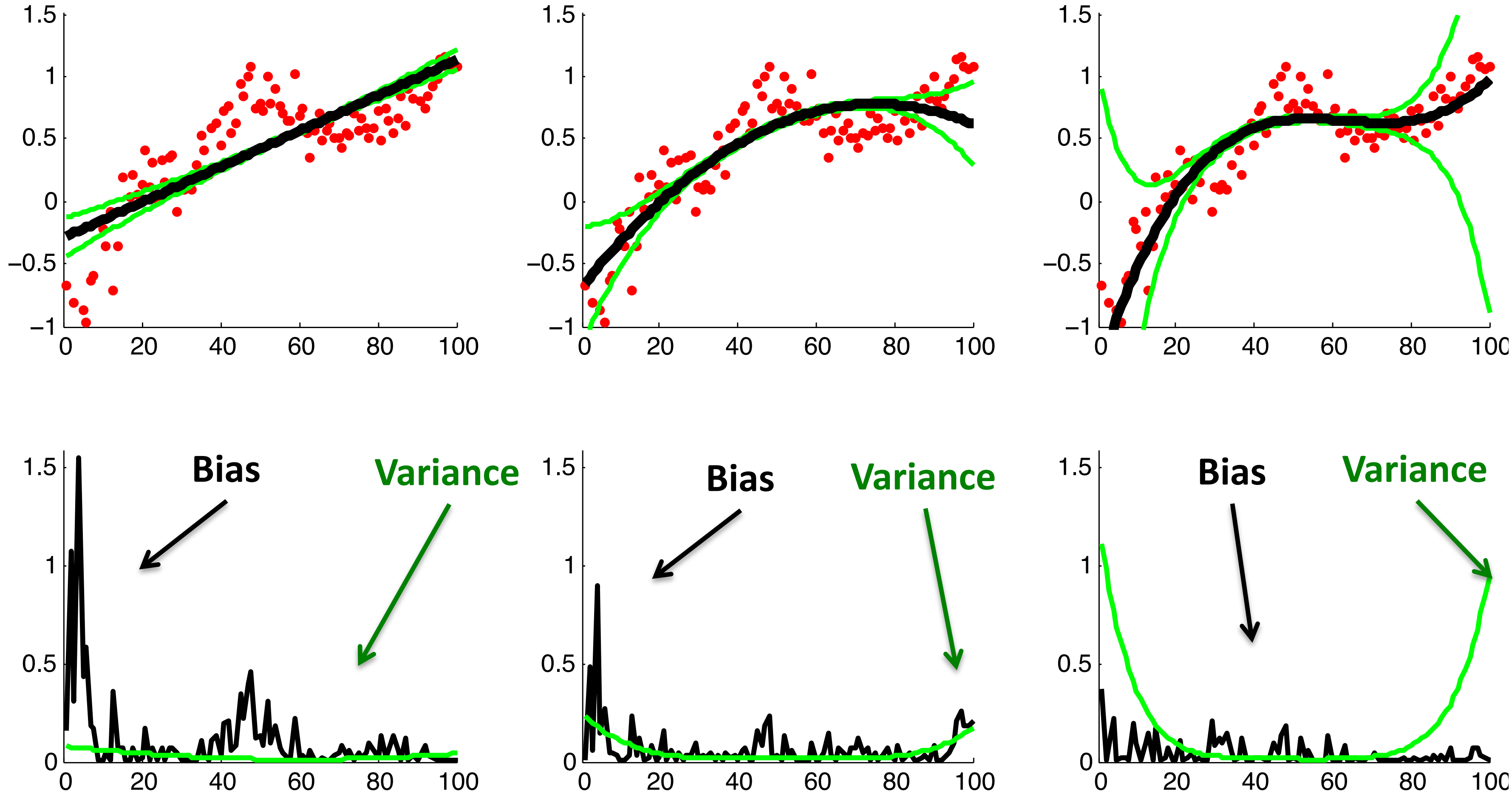
- Gradescope:
  - https://www.gradescope.com/courses/77190
  - Submission, Solutions, Grades
- Piazza
  - https://piazza.com/class/k4c8ma7gjuy21q
  - Course announcements
  - Q&A Forum (use it!)
- Lecture Videos
  - On YouTube (linked from course website)

Recap: Basic Recipe (supervised)

- Training Data: $S = \{(x_i, y_i)\}_{i=1}^{N}$ $\quad x \in R^D$ $\quad y \in \{-1,+1\}$
- Model Class: $f(x \mid w,b) = w^T x - b$ **Linear Models**
- Loss Function: $L(a,b) = (a-b)^2$ **Squared Loss**
- Learning Objective: $\operatorname*{argmin}_{w,b} \sum_{i=1}^{N} L\big(y_i, f(x_i \mid w,b)\big)$

Optimization Problem

Recap: Bias-Variance Trade-off



Recap: Complete Pipeline

$$S = \{(x_i, y_i)\}_{i=1}^{N}$$

Training Data

$$f(x \mid w, b) = w^T x - b$$

Model Class(es)

$$L(a,b) = (a-b)^2$$

Loss Function

$$\underset{w,b}{\operatorname{argmin}} \sum_{i=1}^{N} L\big(y_i, f(x_i \mid w, b)\big)$$

Cross Validation & Model Selection

Profit!

Today

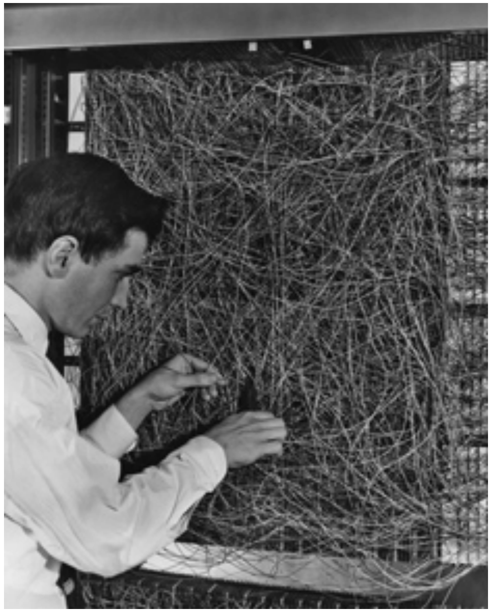
- **Two Basic Learning Algorithms**

- Perceptron Algorithm

- (Stochastic) Gradient Descent
  – Aka, actually solving the optimization problem

The Perceptron

- One of the earliest learning algorithms
  - 1957 by Frank Rosenblatt
- Still a great algorithm
  - Fast
  - Clean analysis
  - Precursor to Neural Networks



Frank Rosenblatt
with the Mark 1
Perceptron Machine

Perceptron Learning Algorithm
(Linear Classification Model)

- $w^1 = 0$, $b^1 = 0$
- For t = 1 ….
  - Receive example (x,y)
  - If $f(x|w^t,b^t) = y$
    - $[w^{t+1}, b^{t+1}] = [w^t, b^t]$
  - Else
    - $w^{t+1} = w^t + yx$
    - $b^{t+1} = b^t - y$

$$f(x \mid w) = sign(w^T x - b)$$

Training Set:

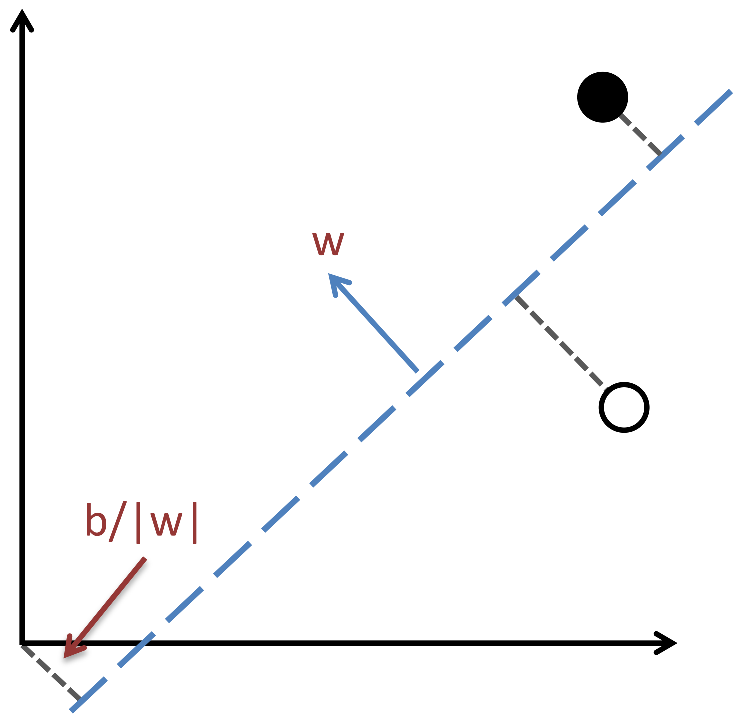
$$S = \left\{(x_i, y_i)\right\}_{i=1}^{N}$$

$$y \in \left\{+1, -1\right\}$$

Go through training set in arbitrary order (e.g., randomly)

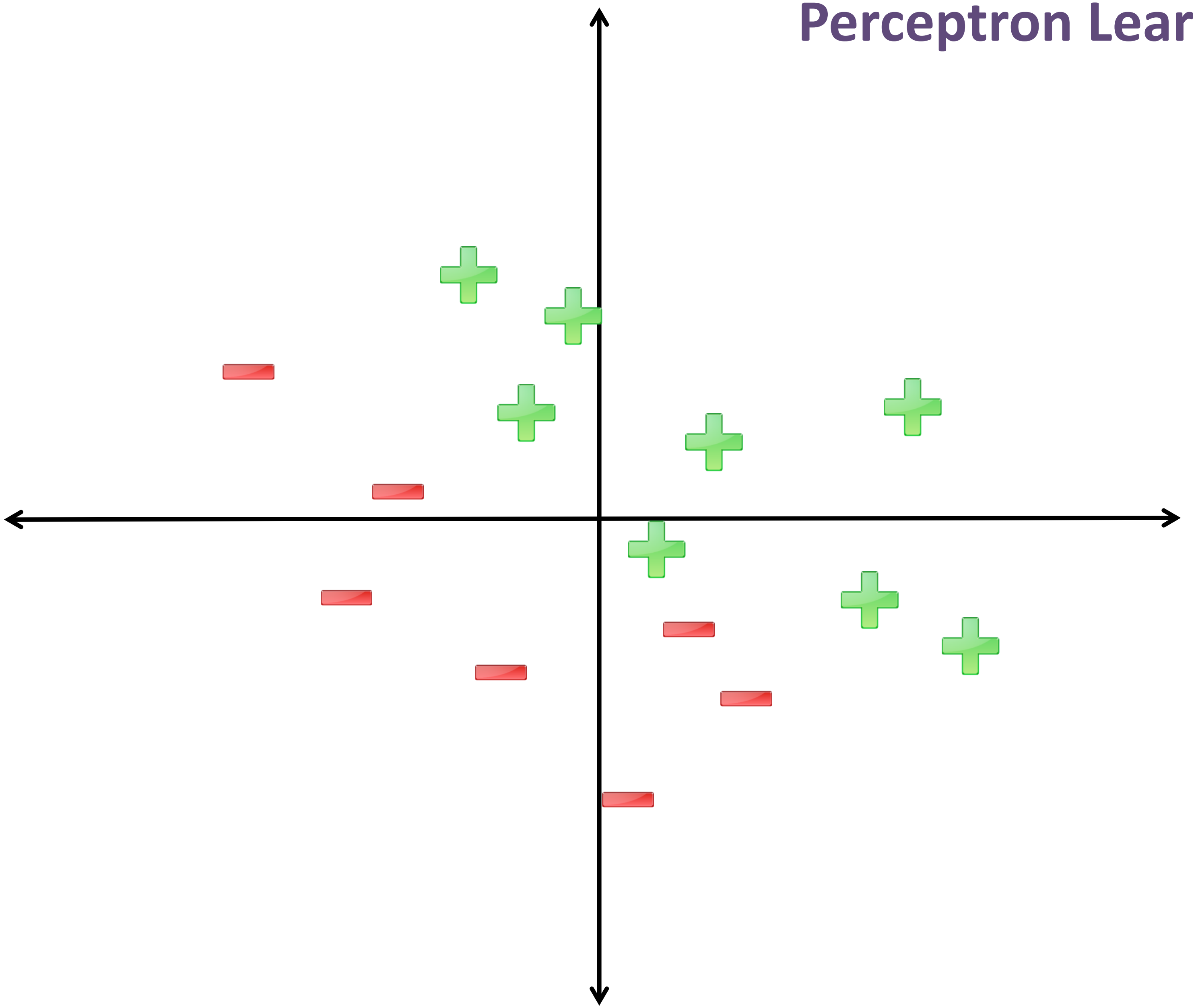
Aside: Hyperplane Distance

- Line is a 1D, Plane is 2D
- Hyperplane is many D
  - Includes Line and Plane
- Defined by (w,b)
- Distance: $$\frac{|w^T x - b|}{\|w\|}$$
- Signed Distance: $$\frac{w^T x - b}{\|w\|}$$

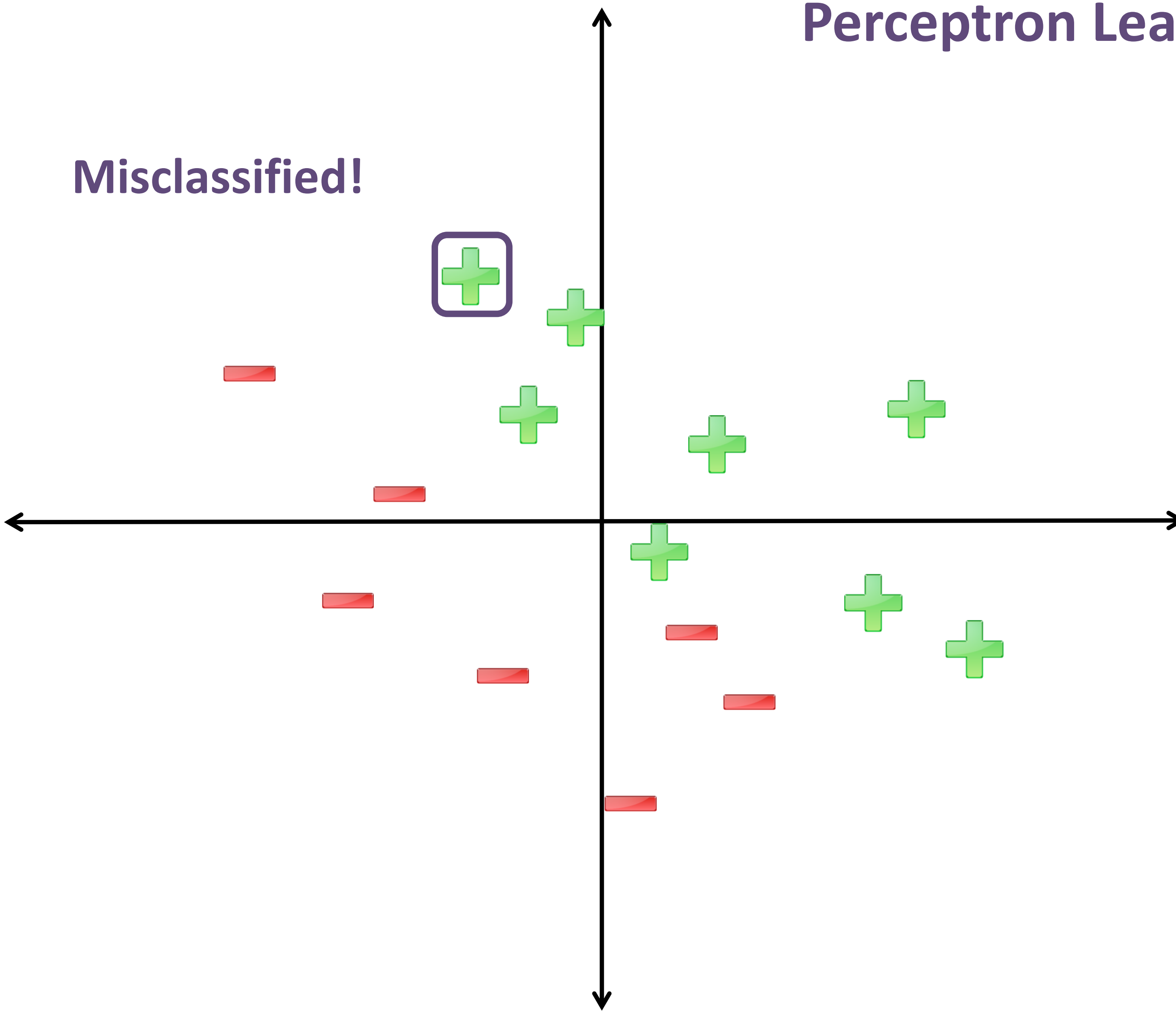


Linear Model = un-normalized signed distance!

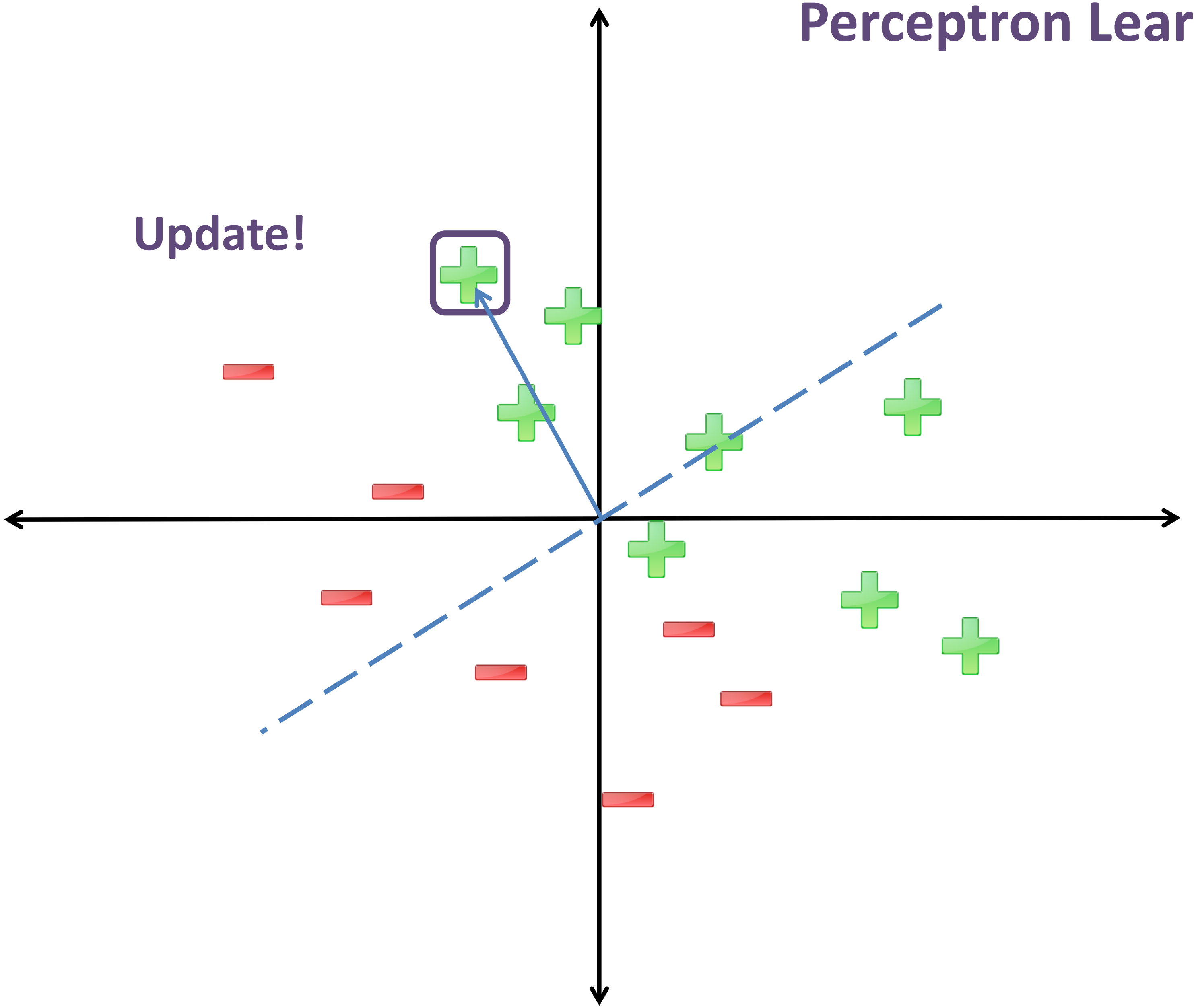
Perceptron Learning



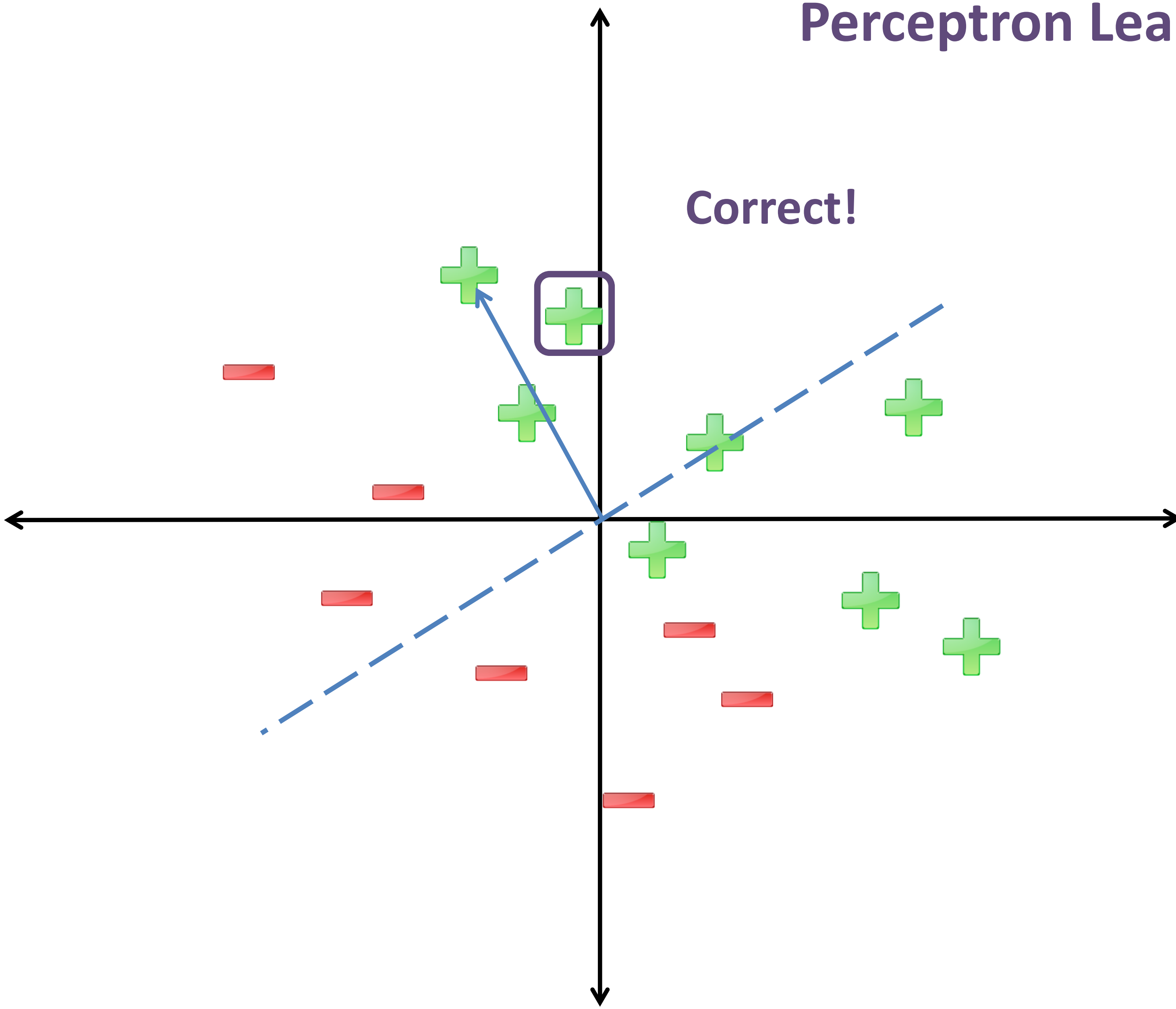
Perceptron Learning



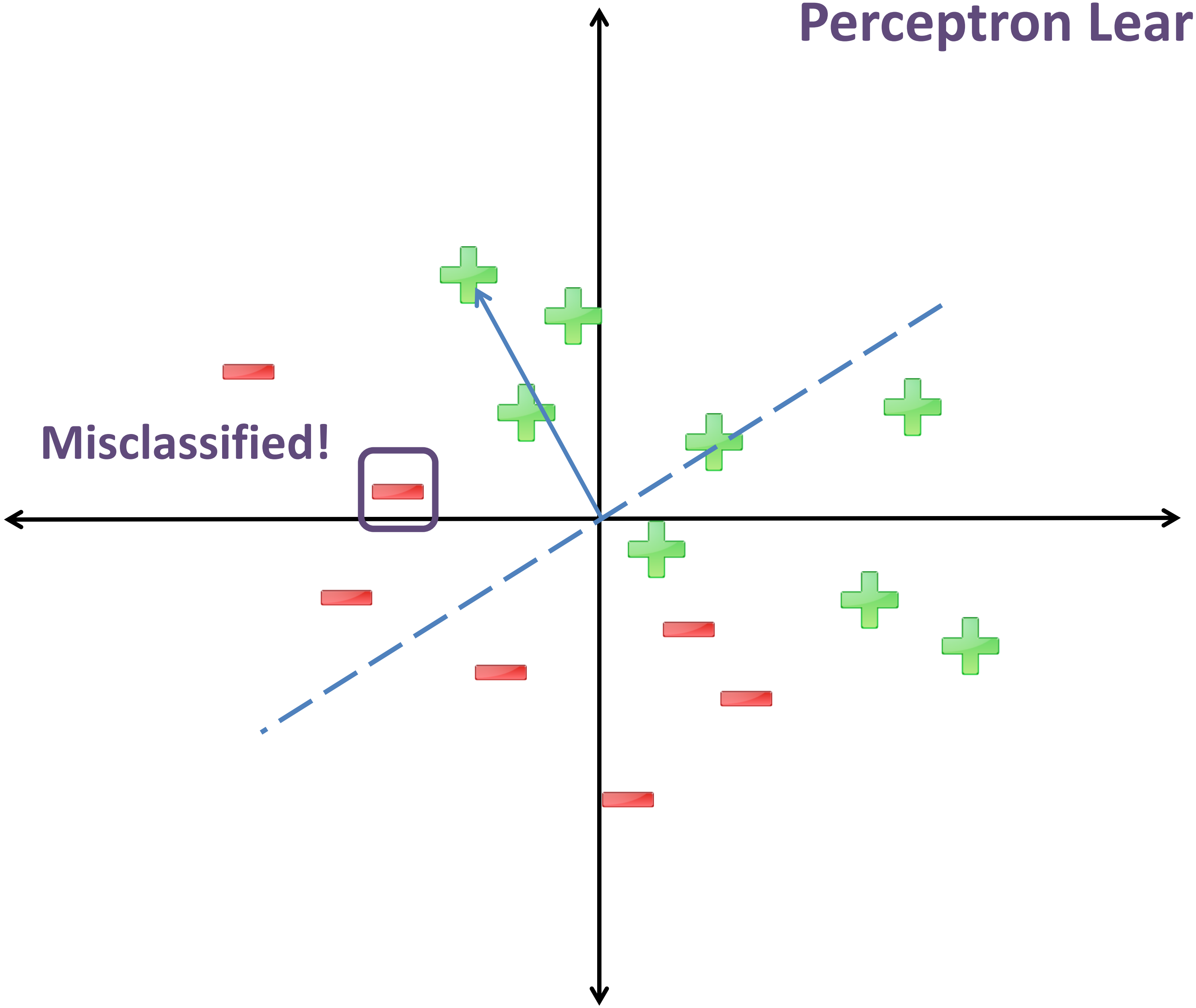
Perceptron Learning



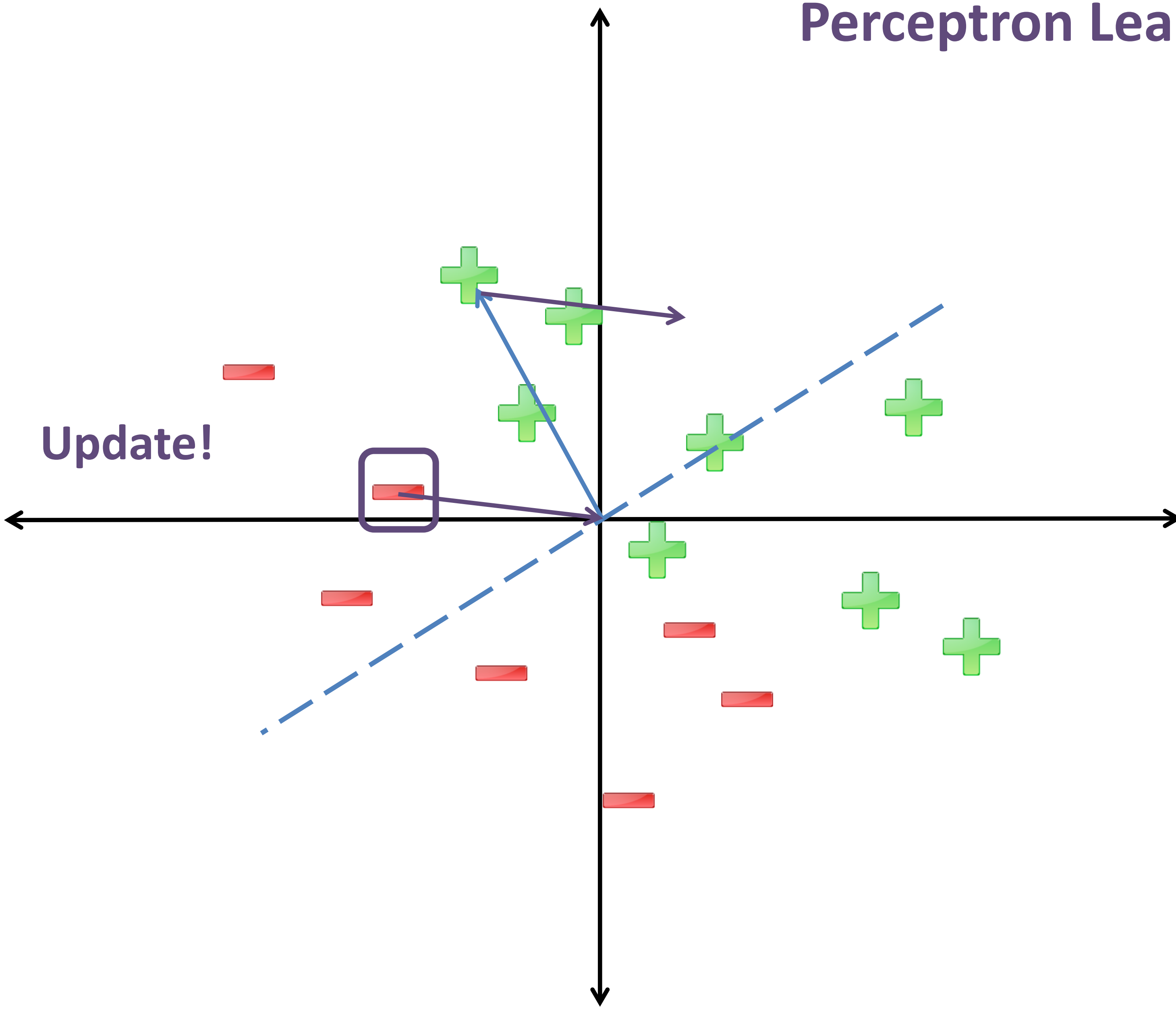
Perceptron Learning



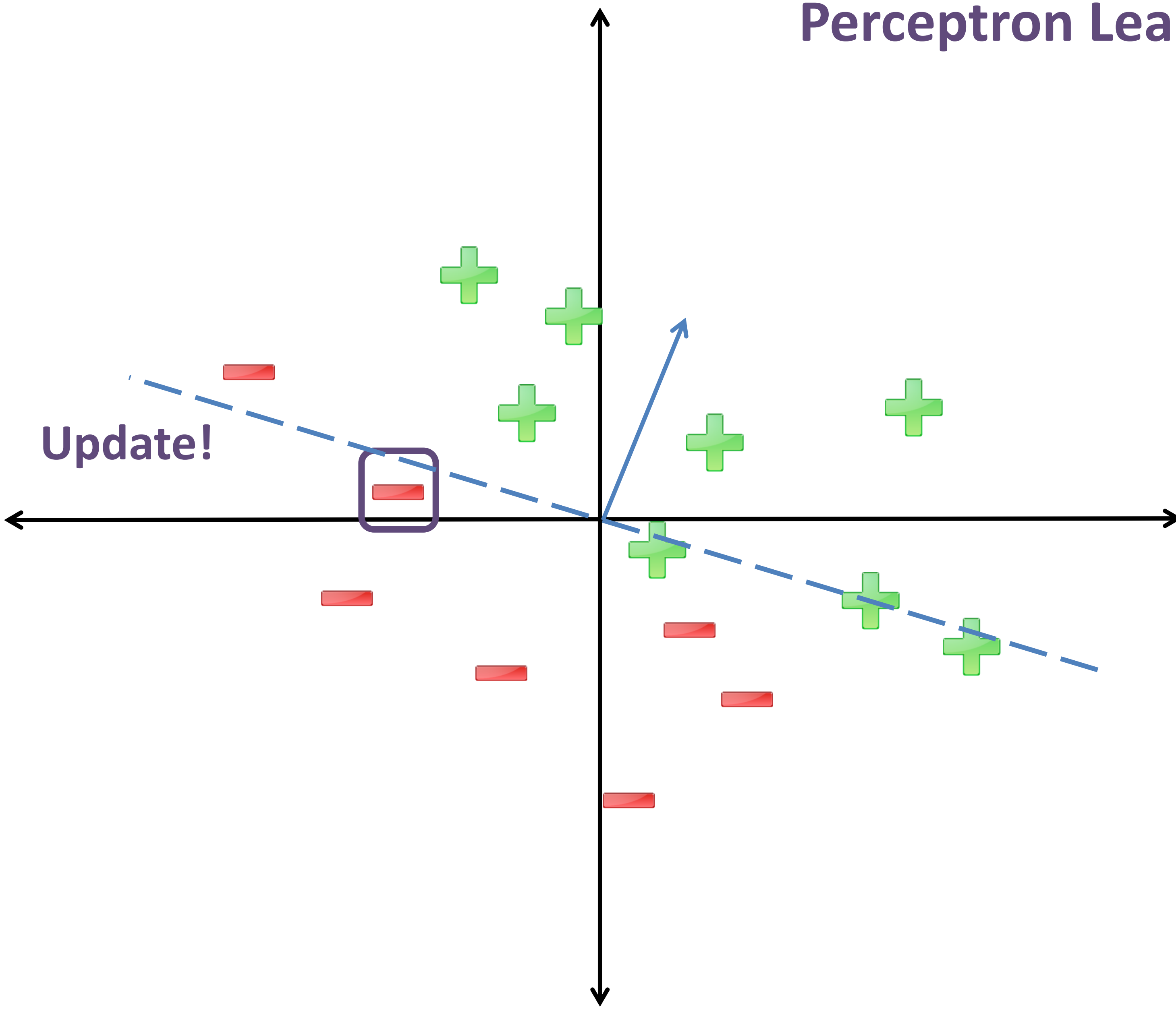
Perceptron Learning



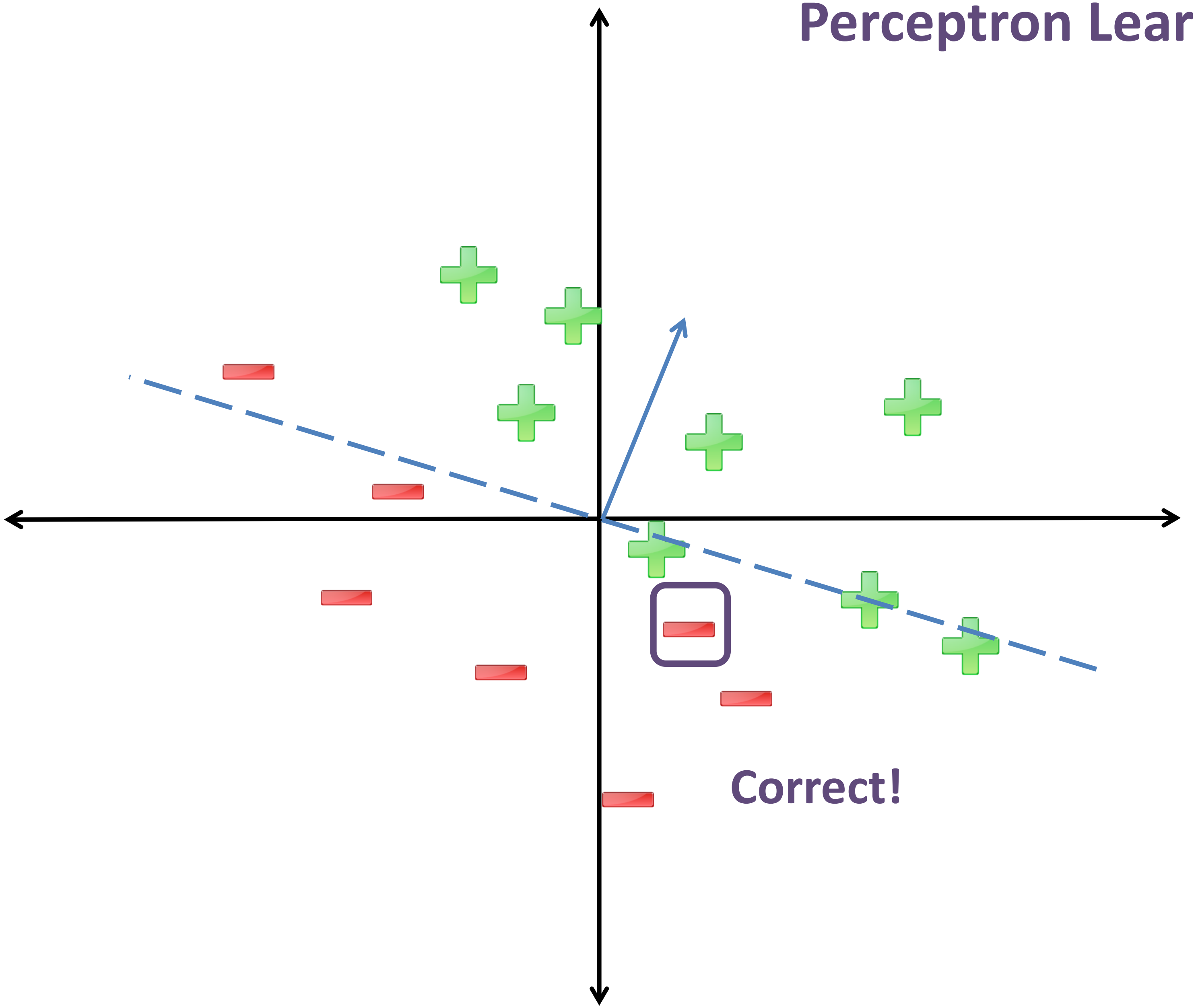
Perceptron Learning



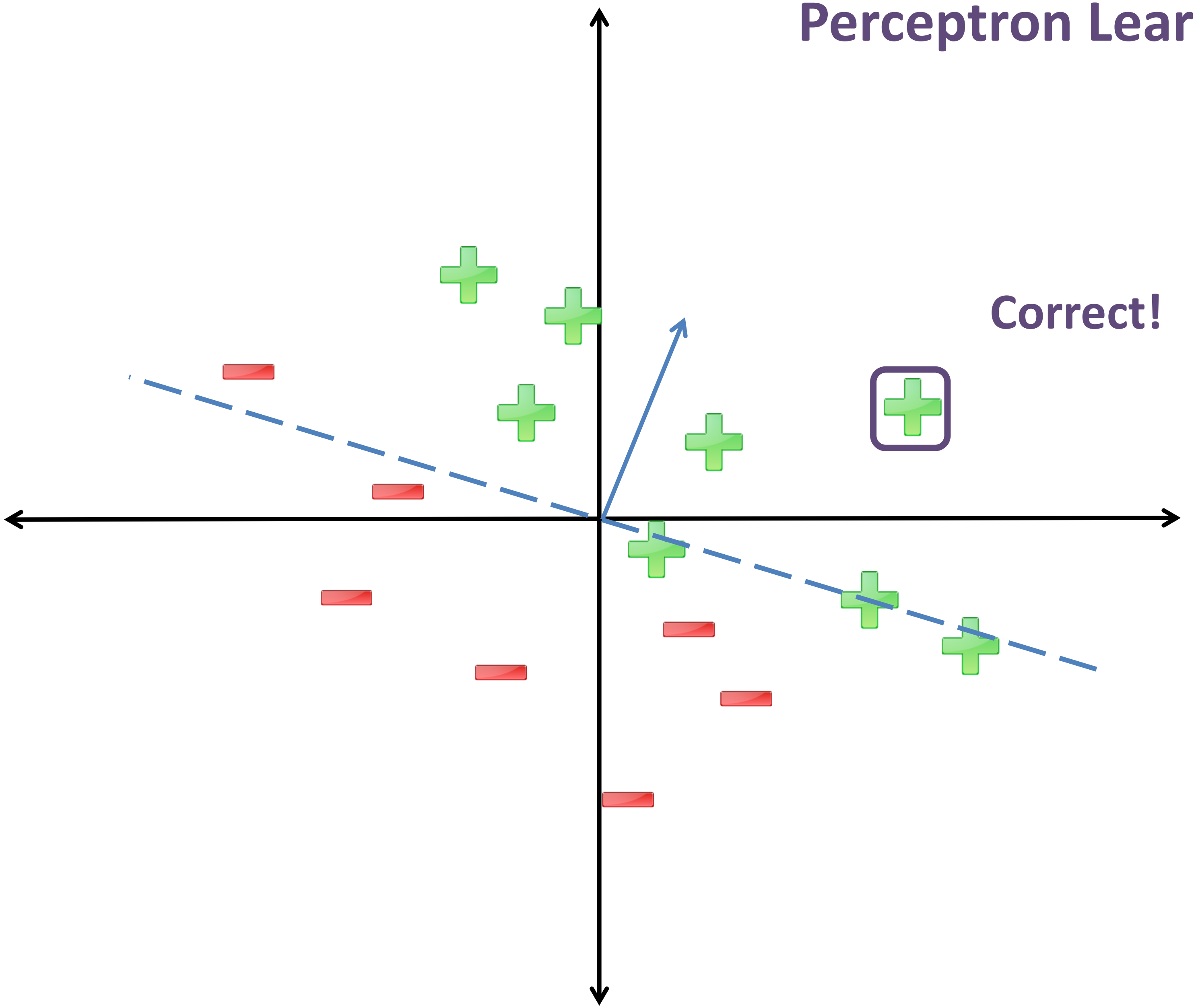
Perceptron Learning



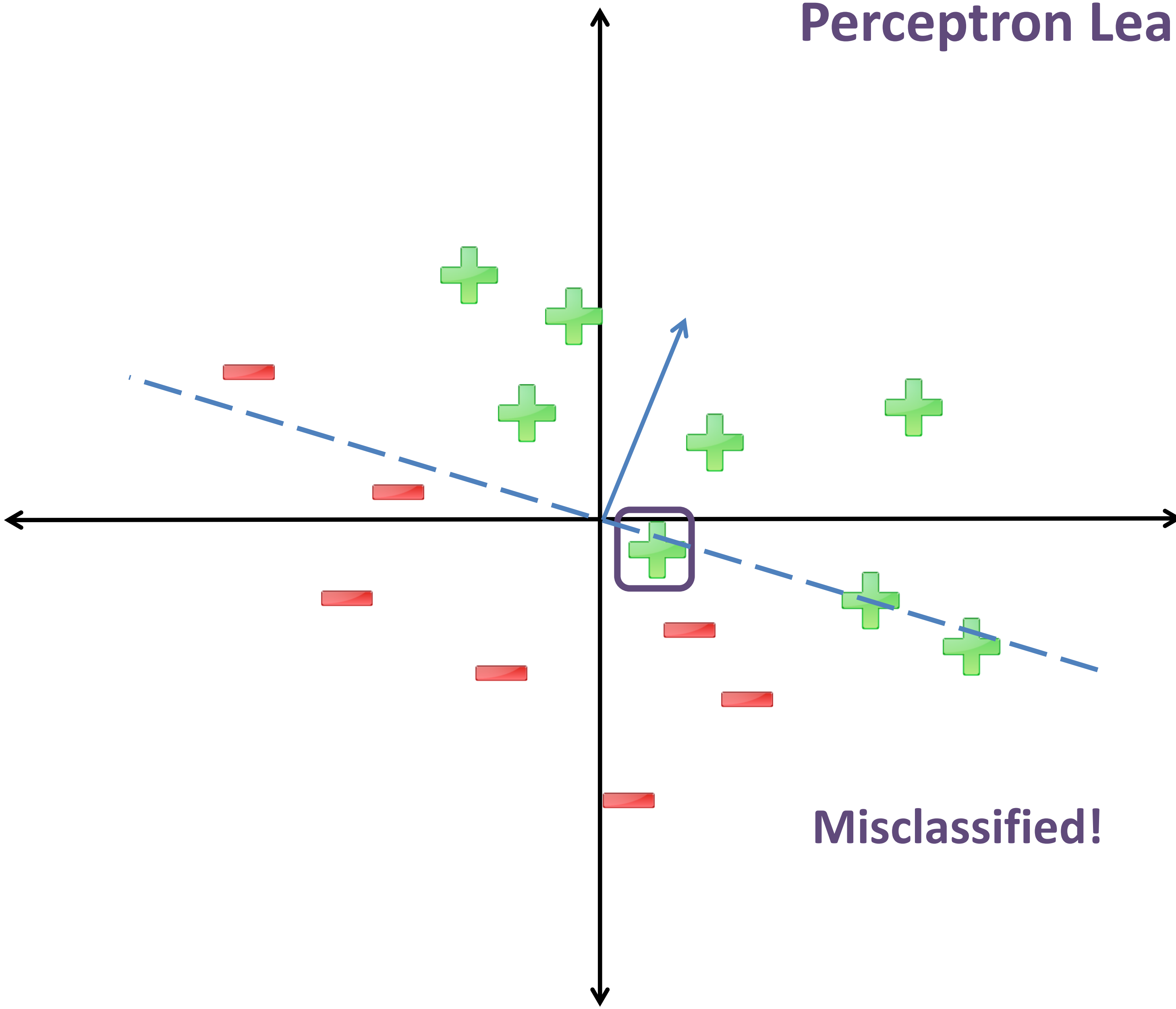
Perceptron Learning



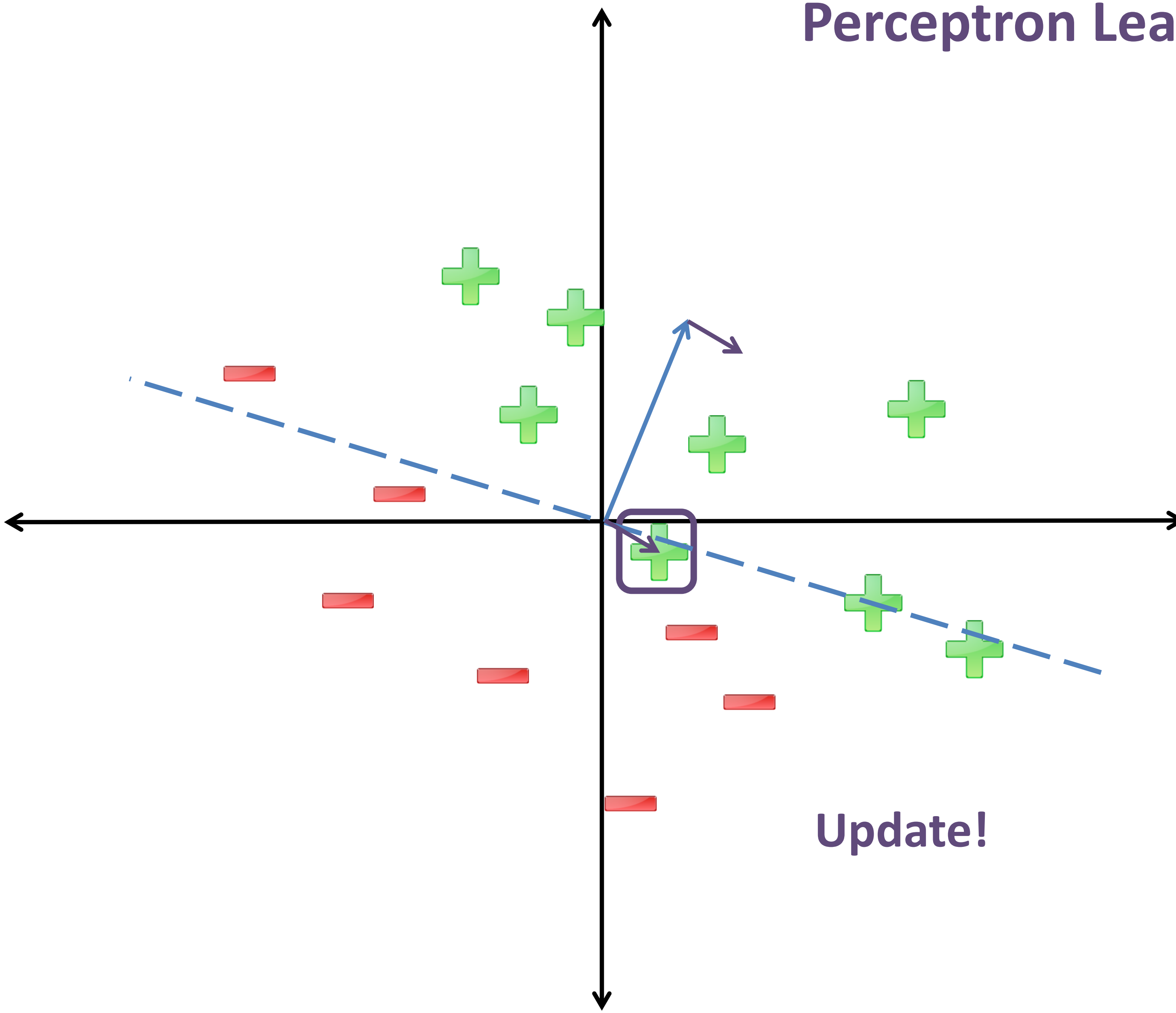
Perceptron Learning



Perceptron Learning

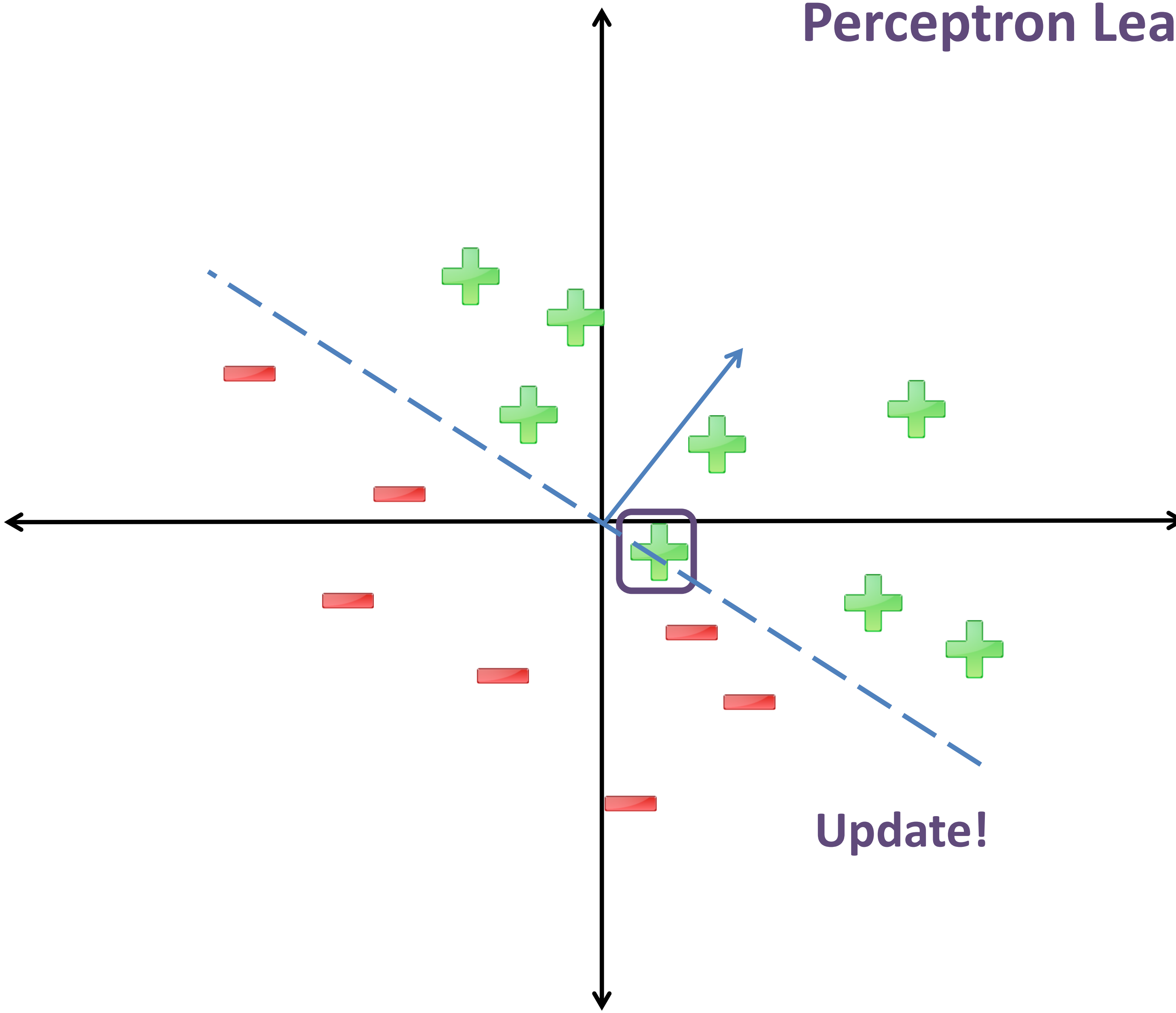


Perceptron Learning



Update!

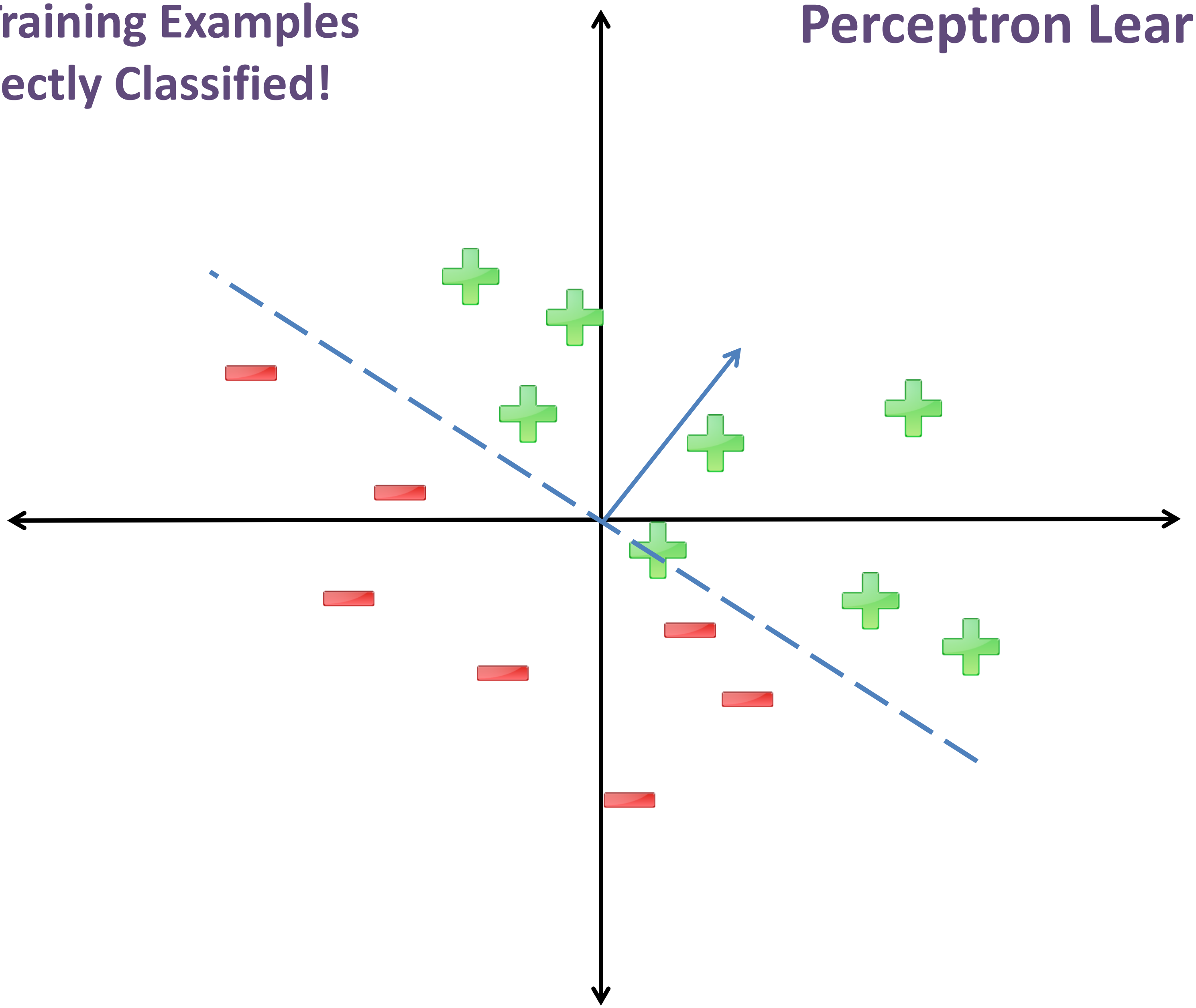
Perceptron Learning



Update!

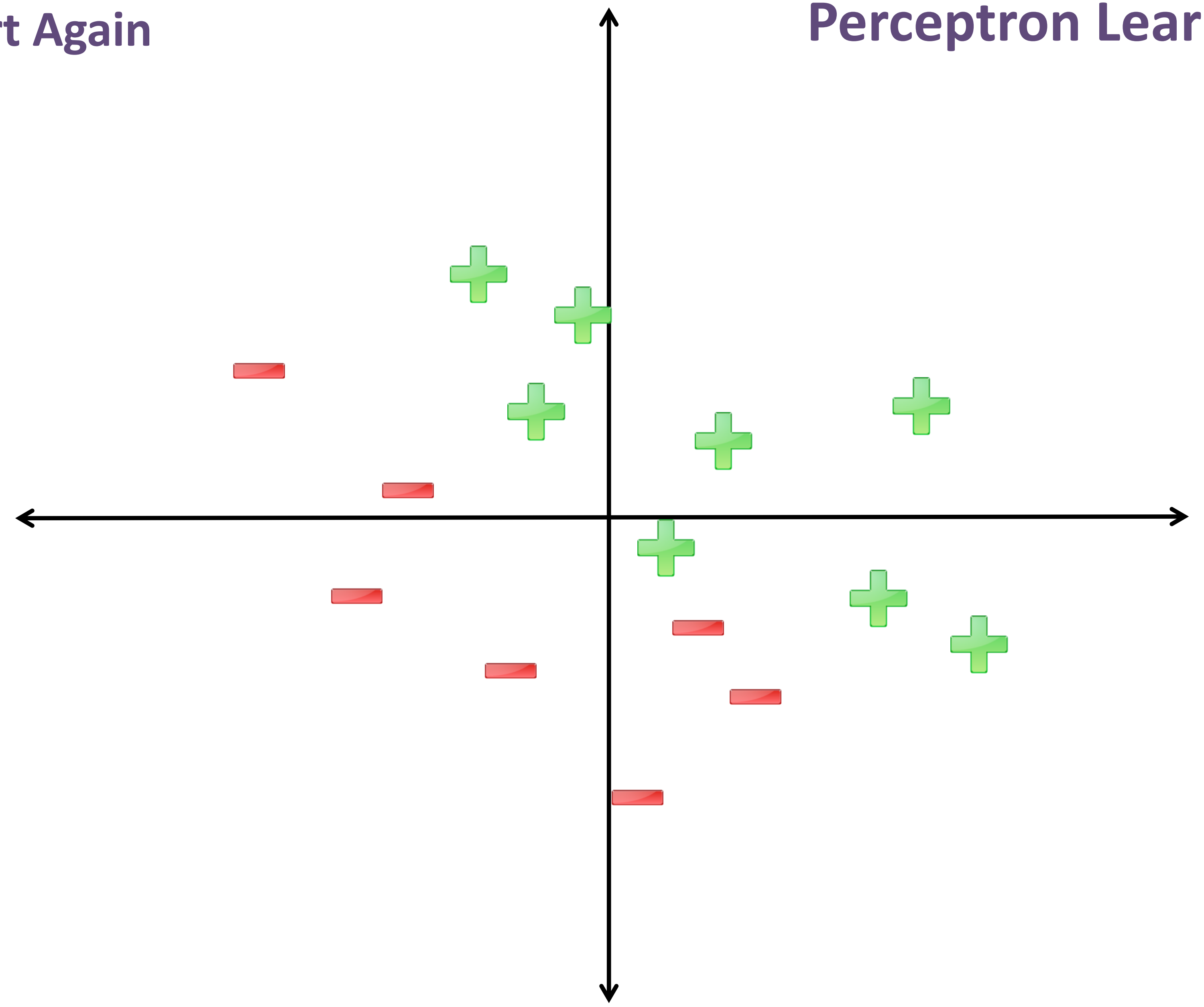
Perceptron Learning

All Training Examples Correctly Classified!

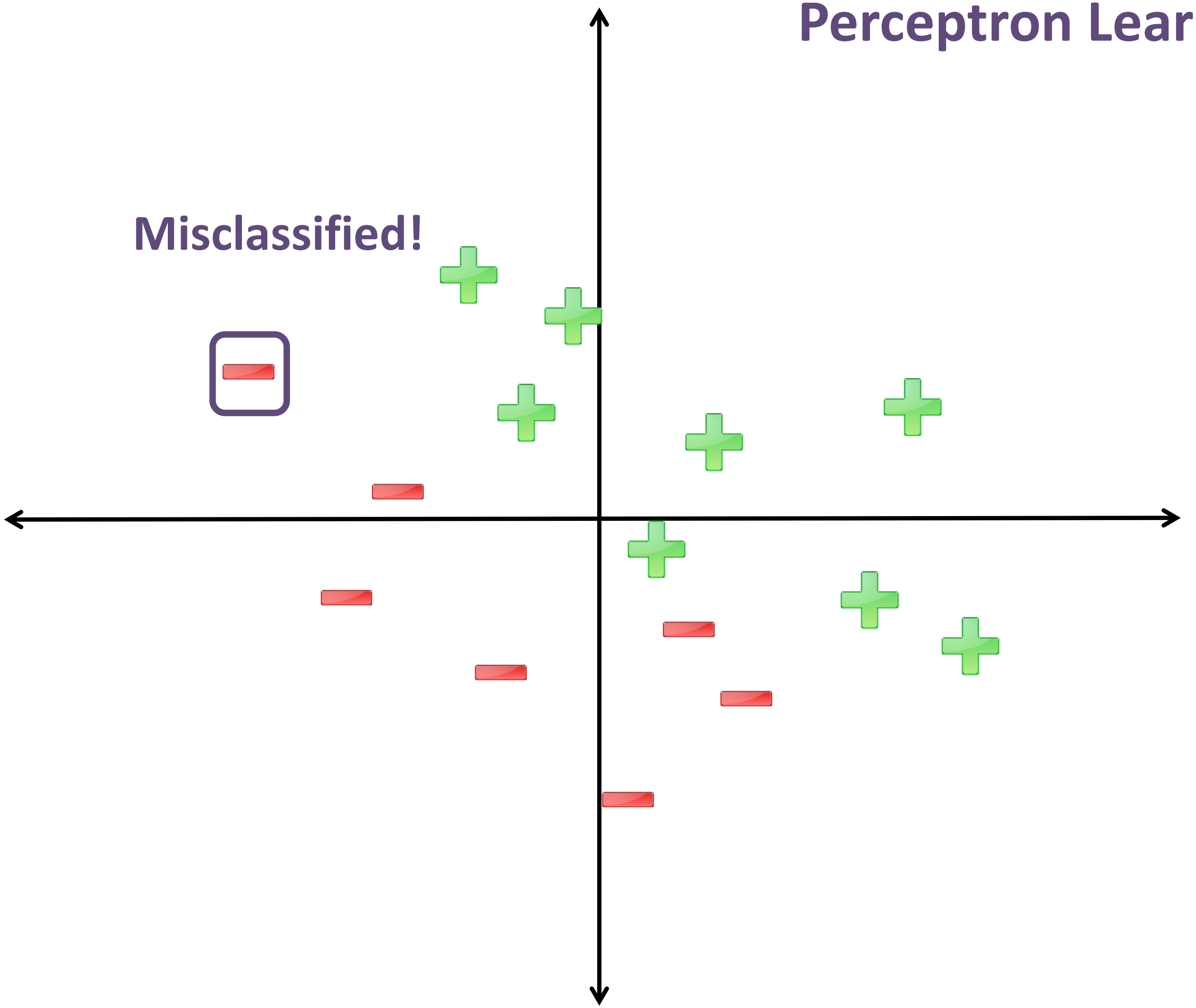


Start Again

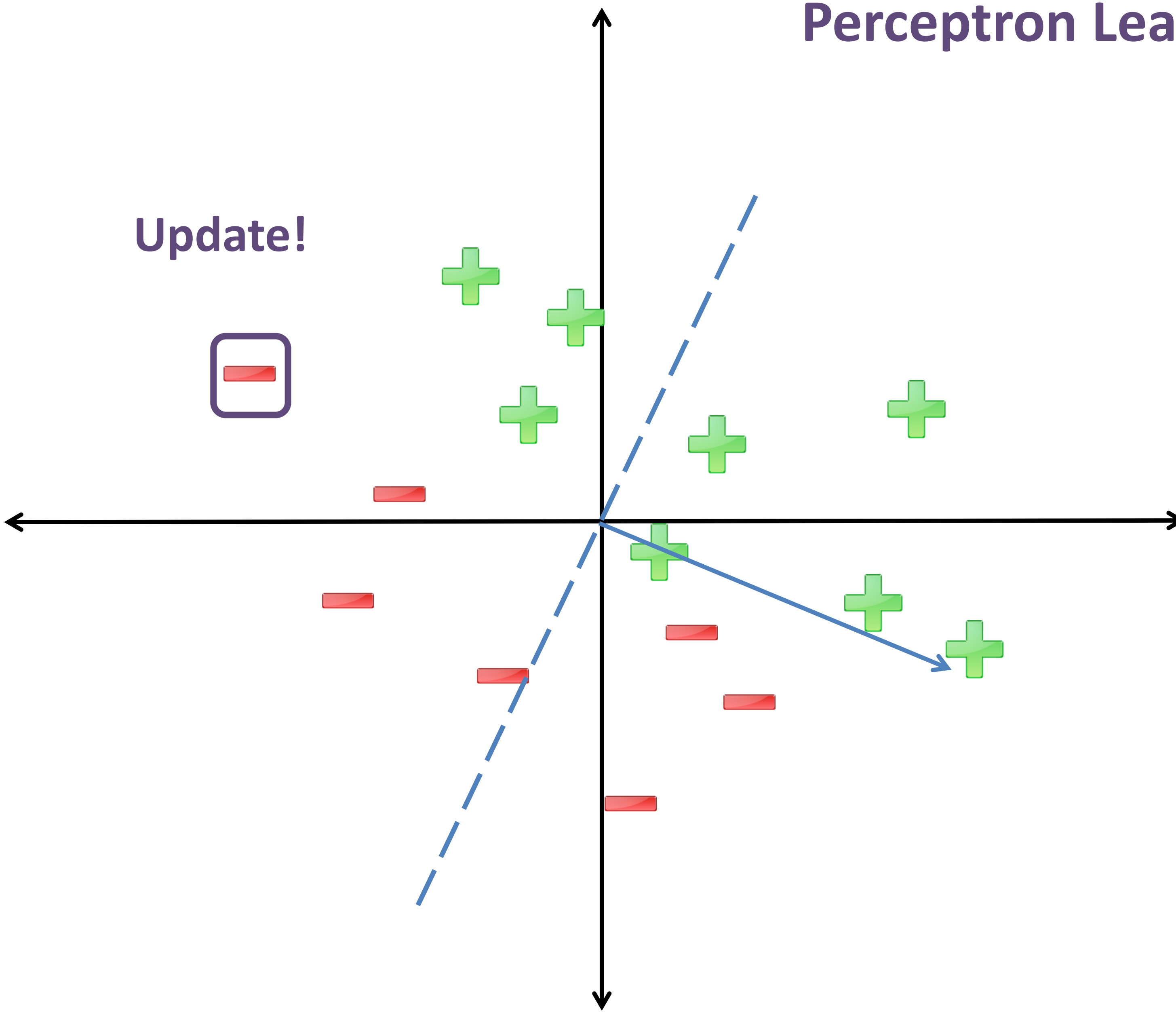
Perceptron Learning



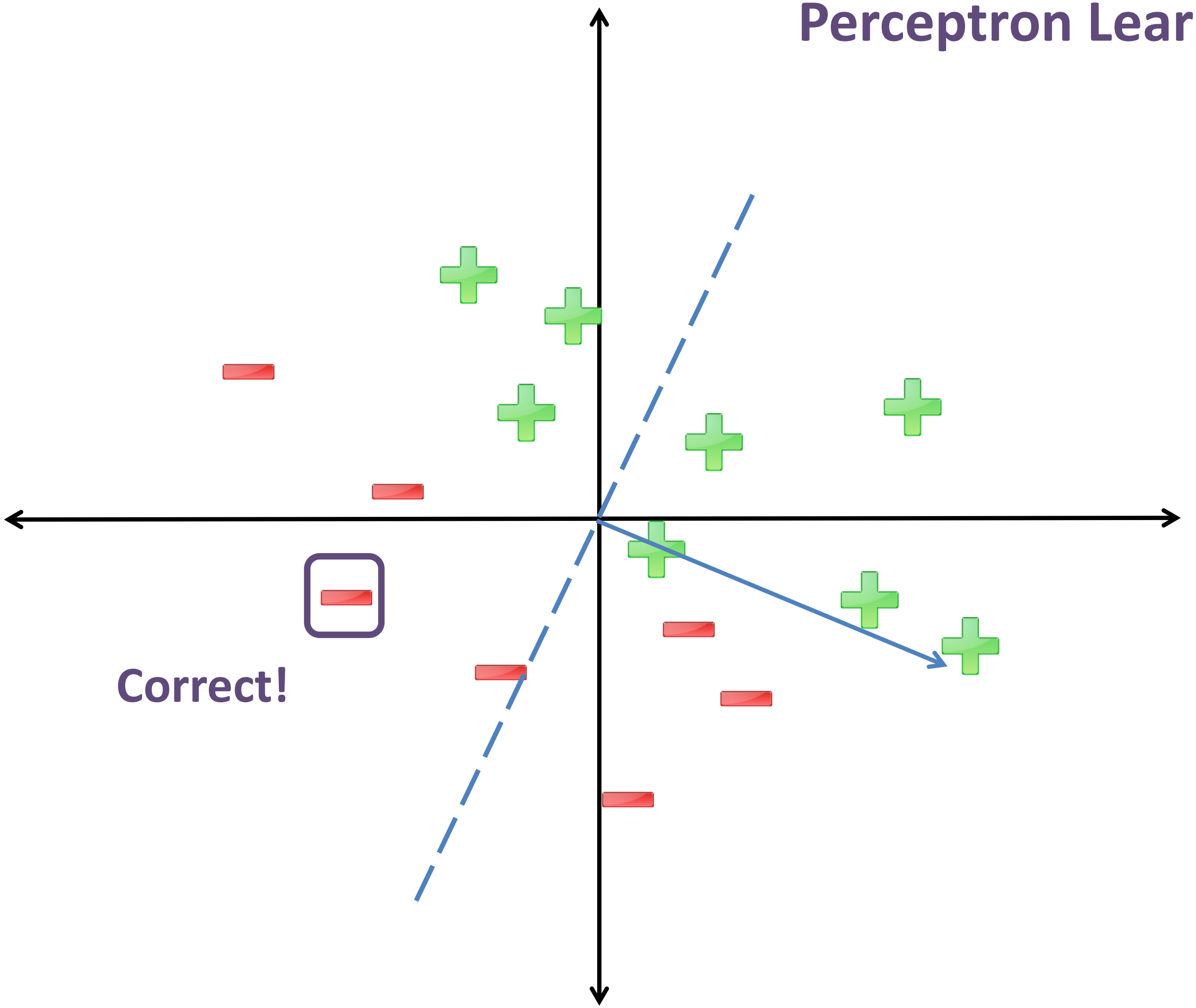
Perceptron Learning



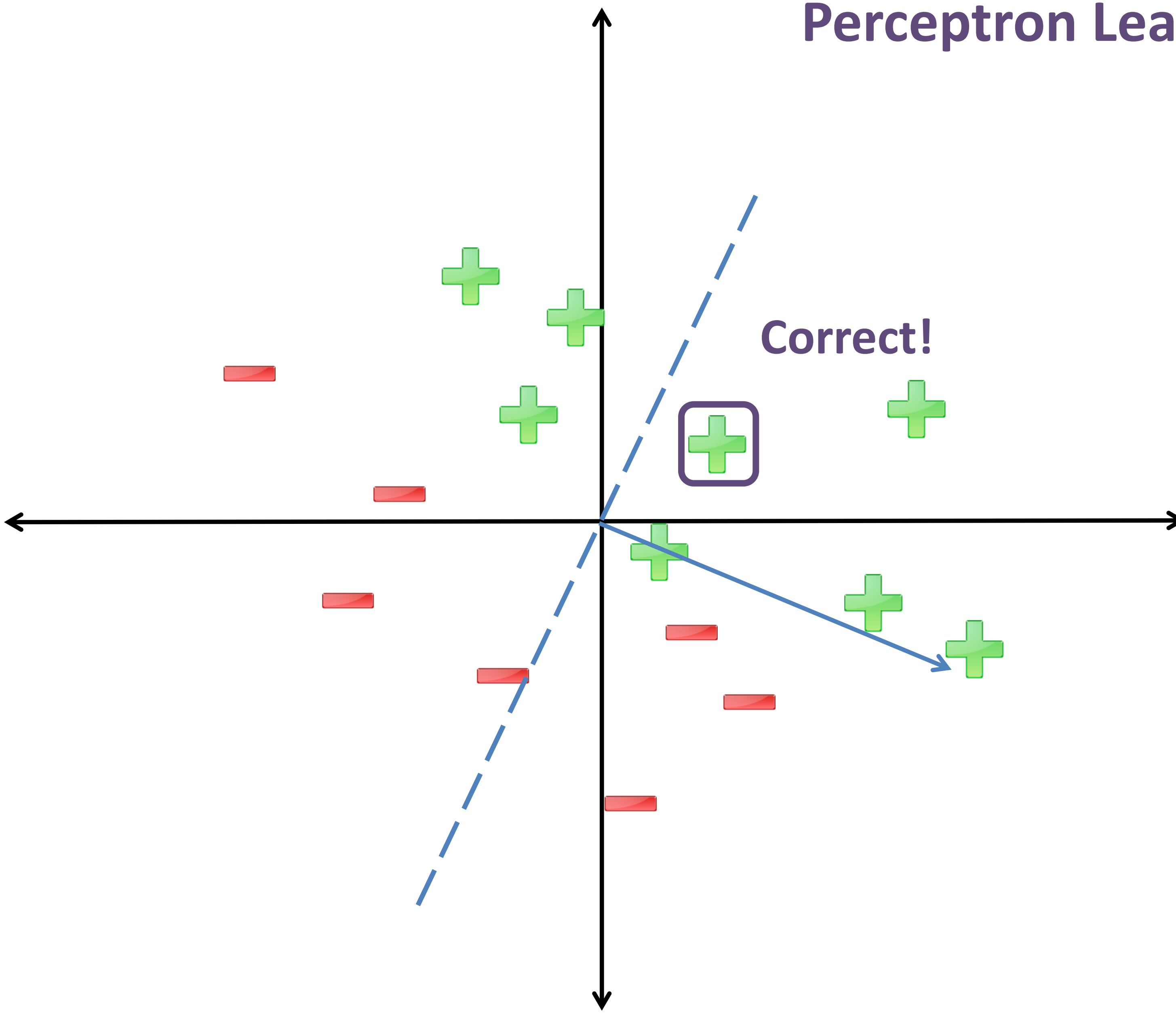
Perceptron Learning



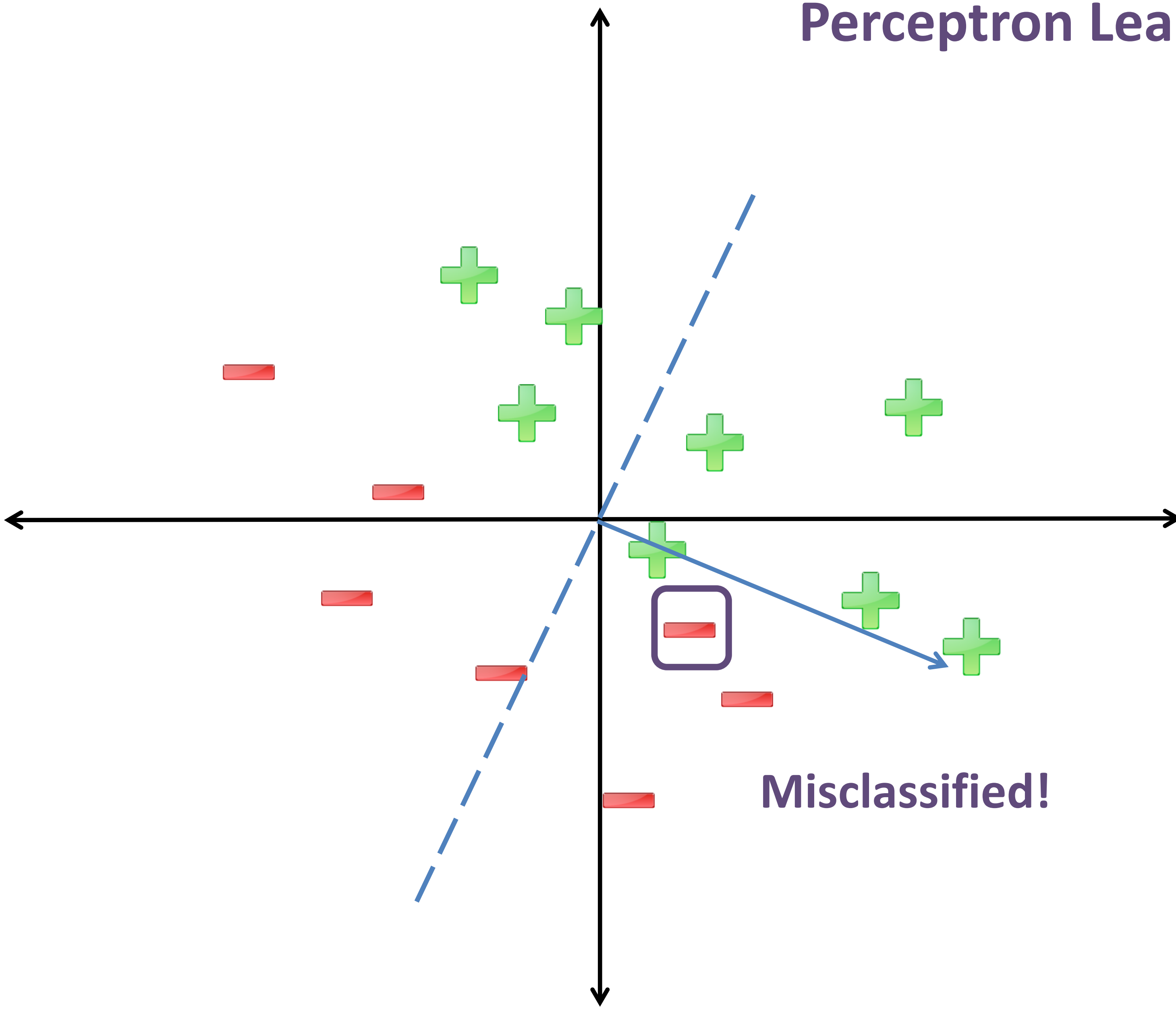
Perceptron Learning



Perceptron Learning

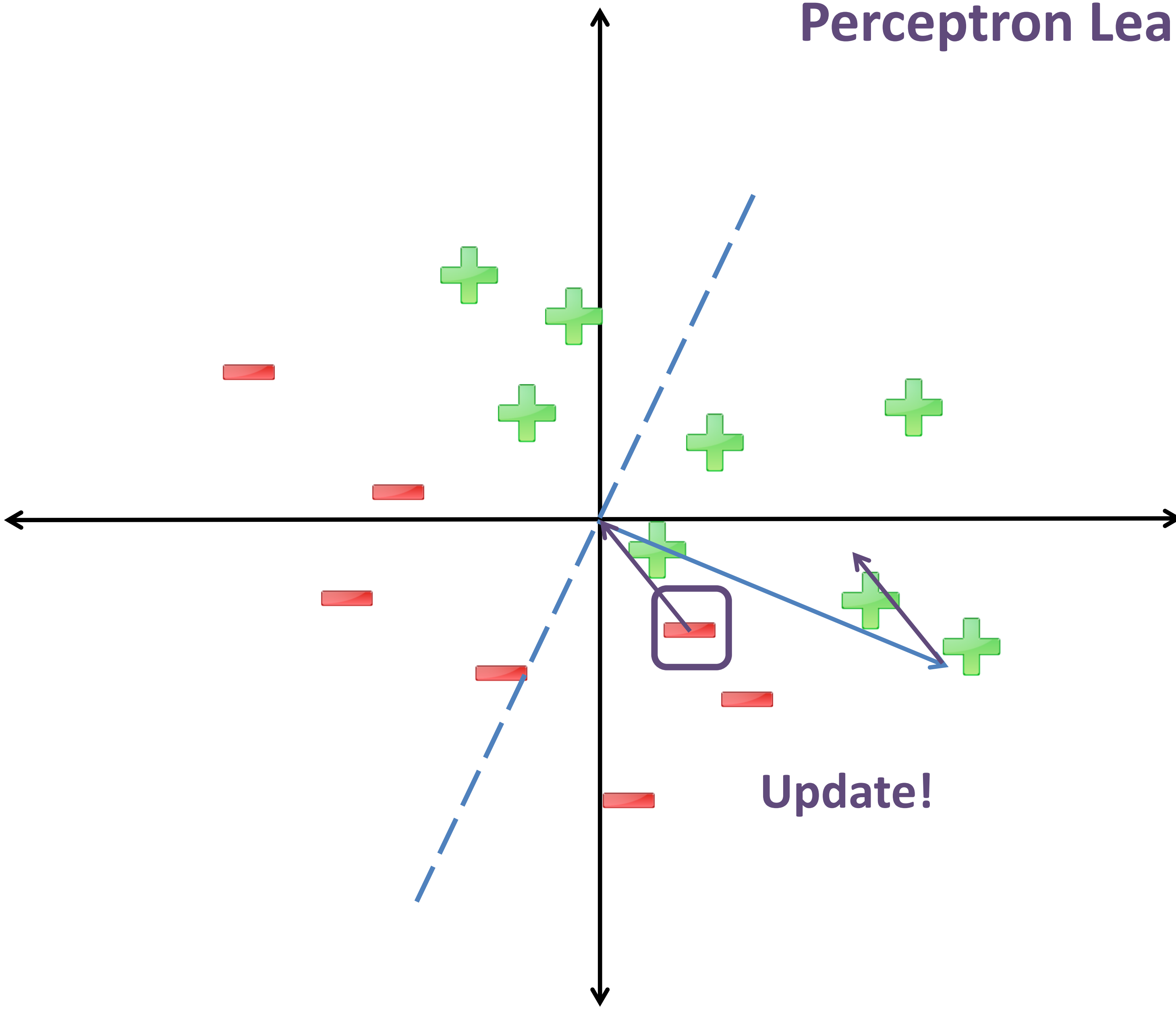


Perceptron Learning

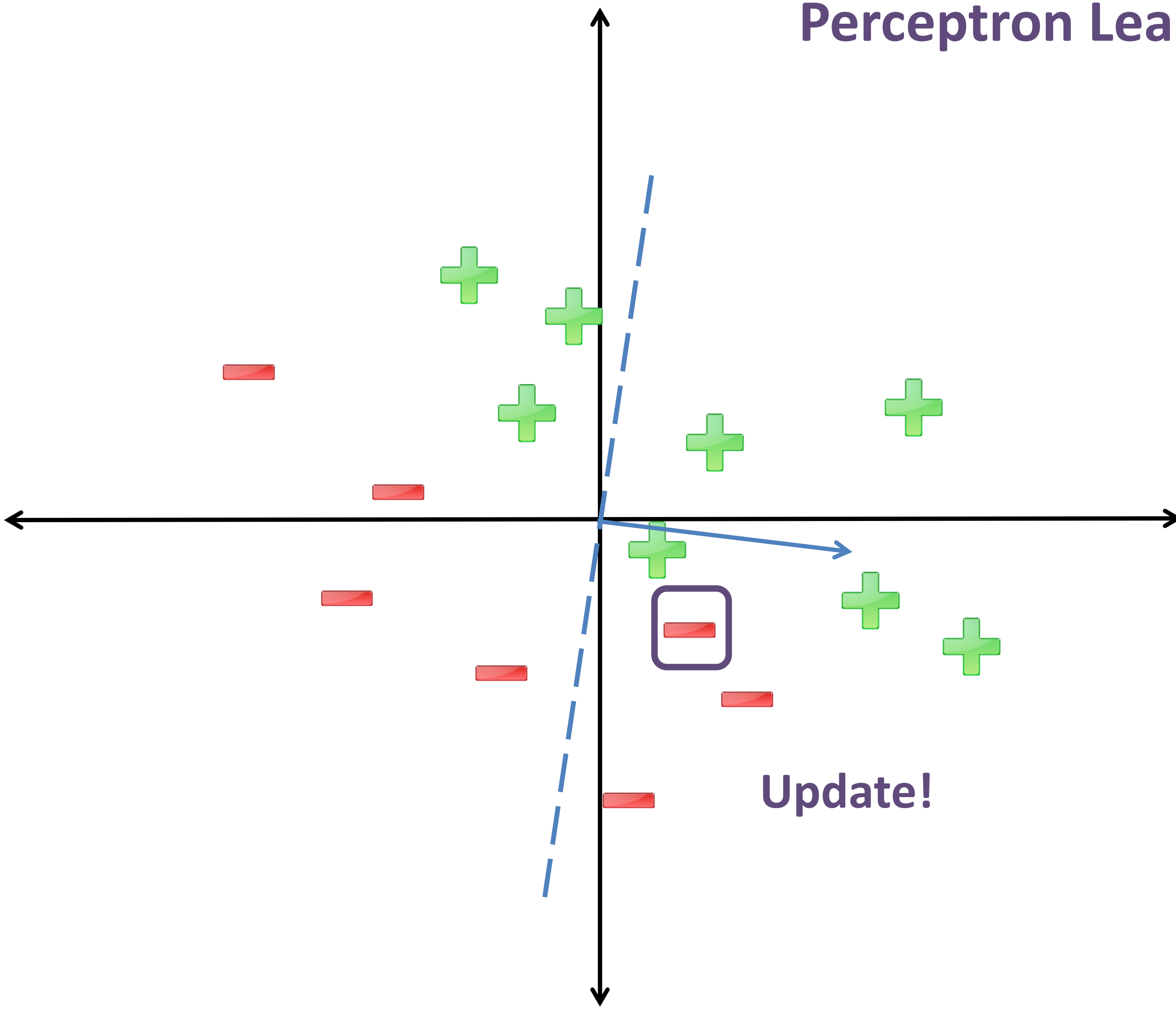


Misclassified!

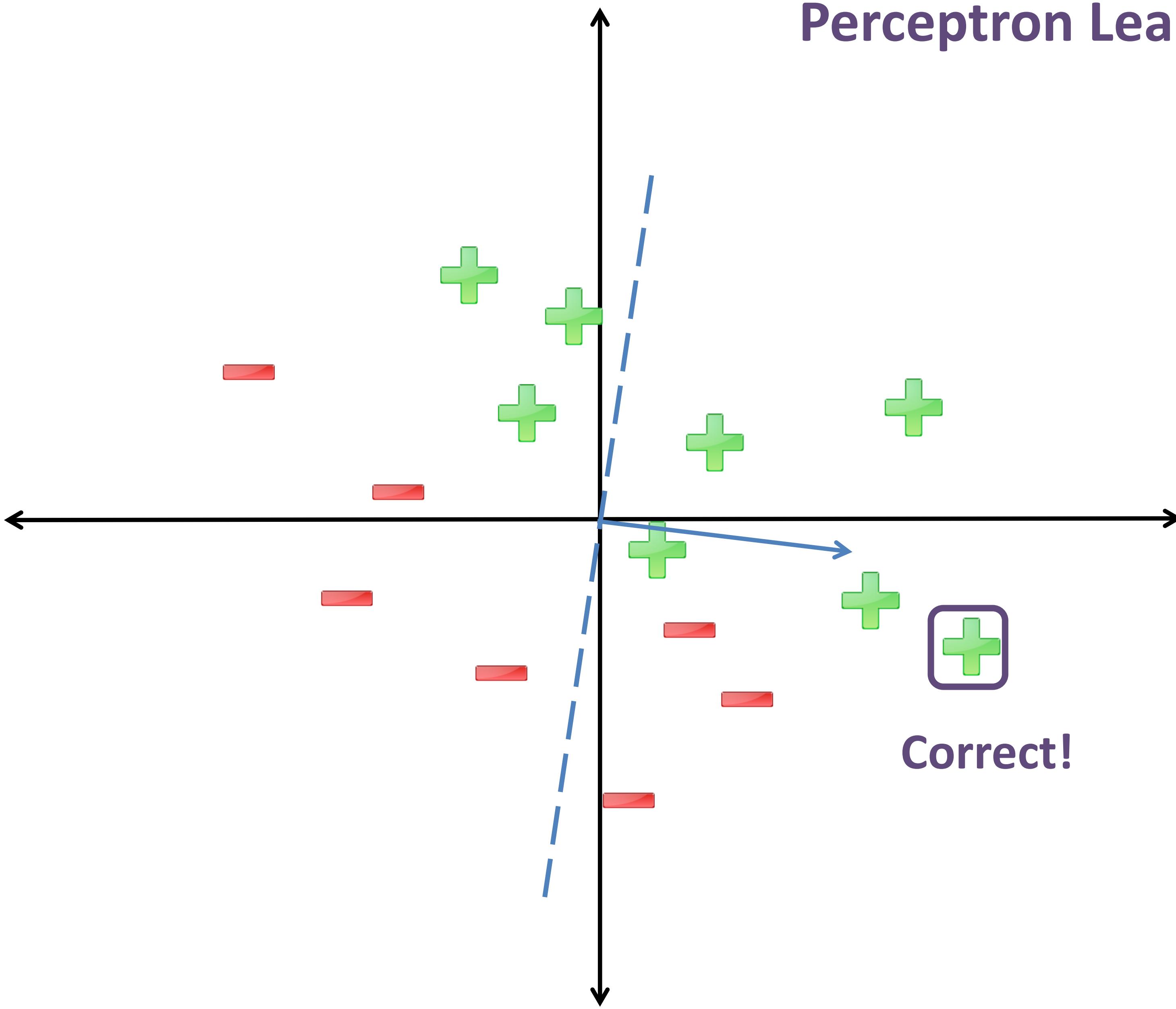
Perceptron Learning



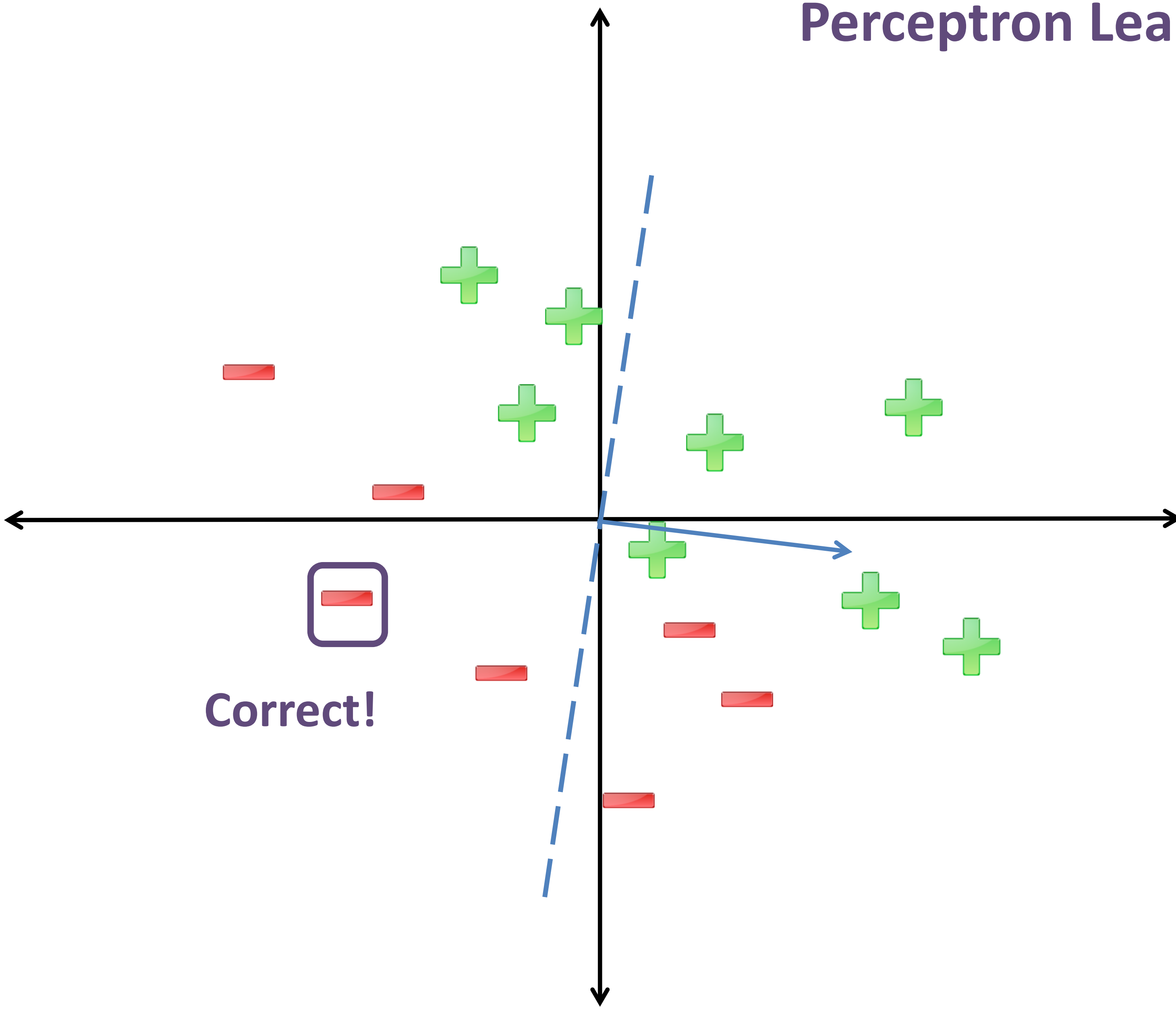
Perceptron Learning



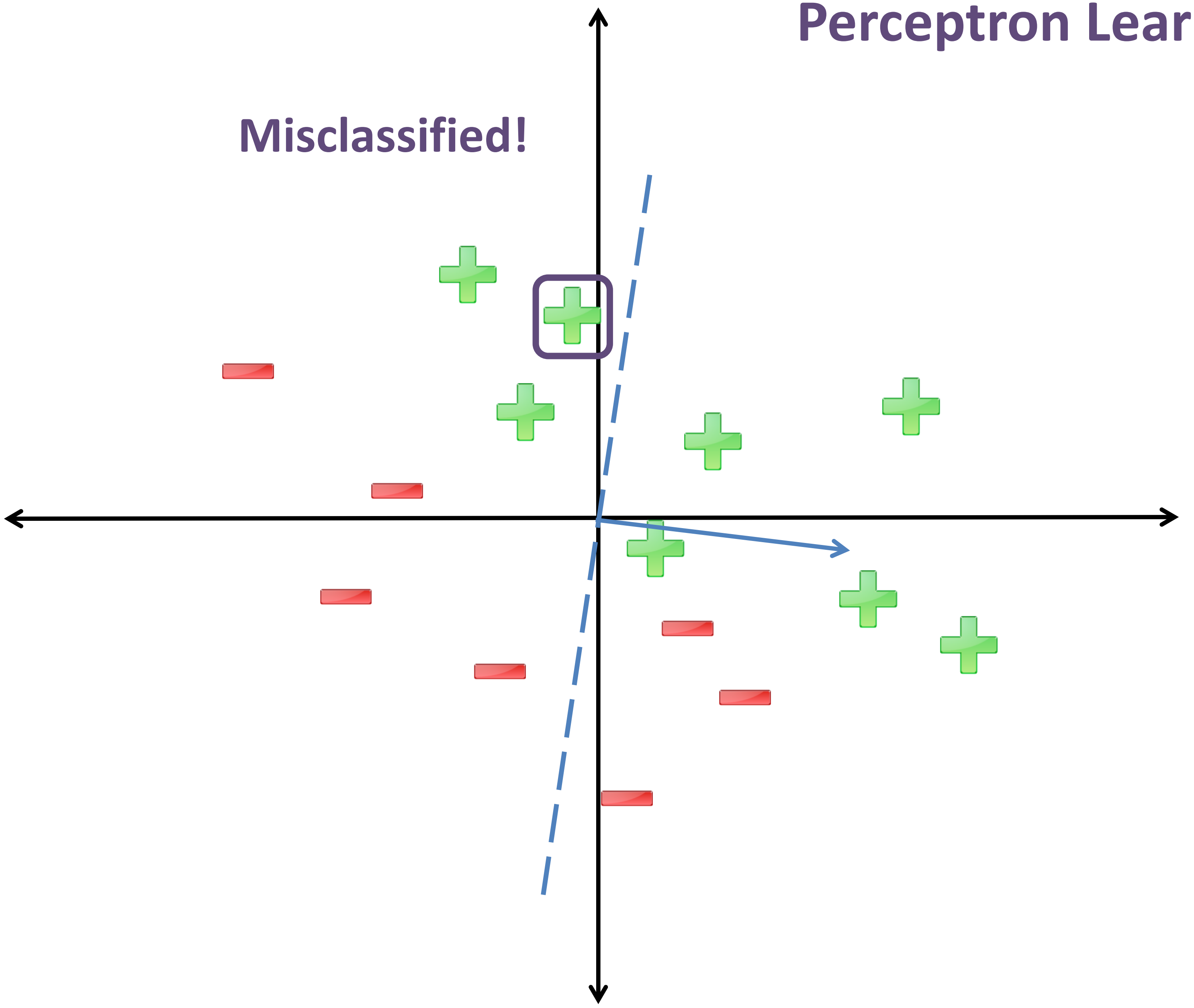
Perceptron Learning



Perceptron Learning

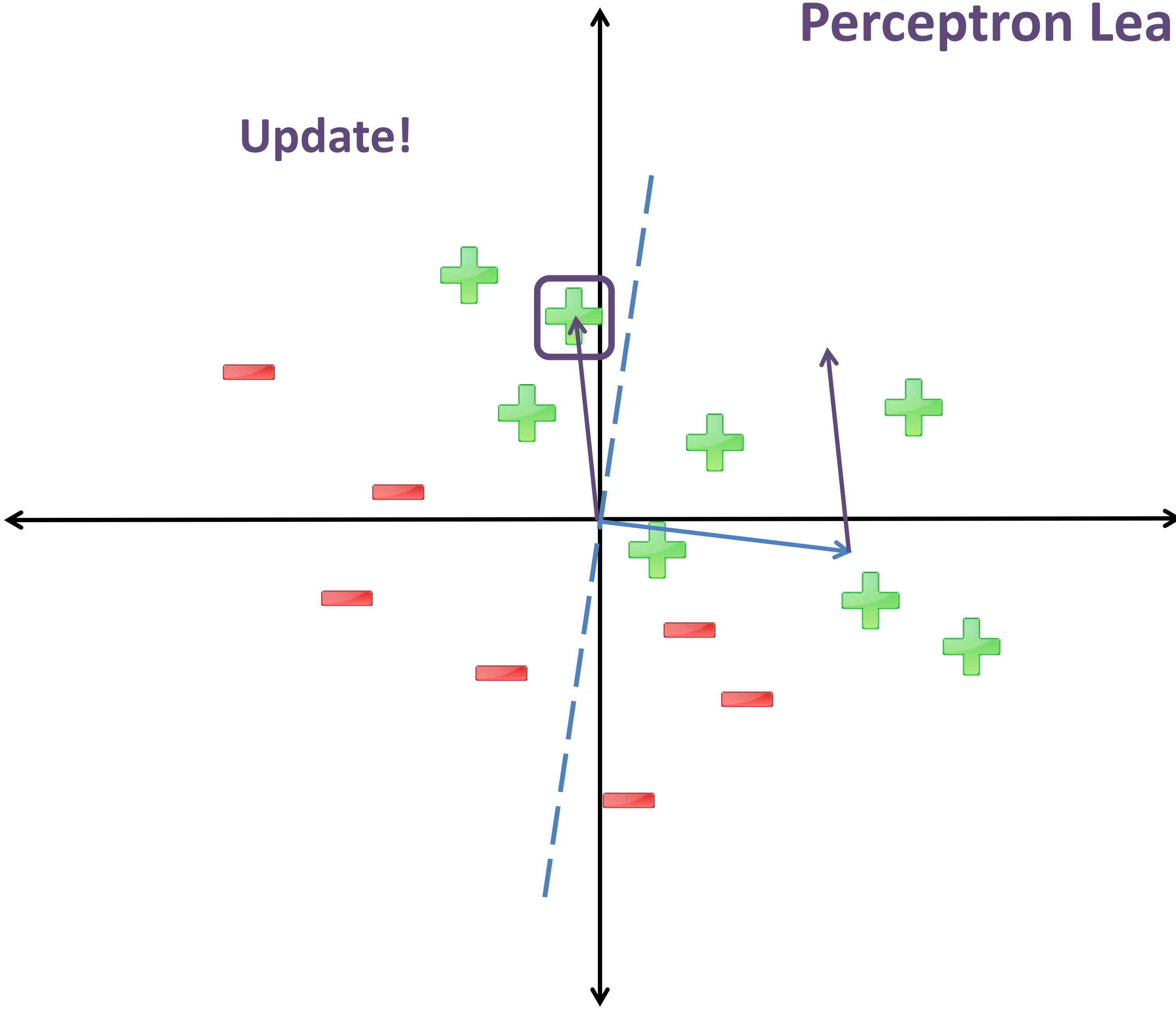


Perceptron Learning

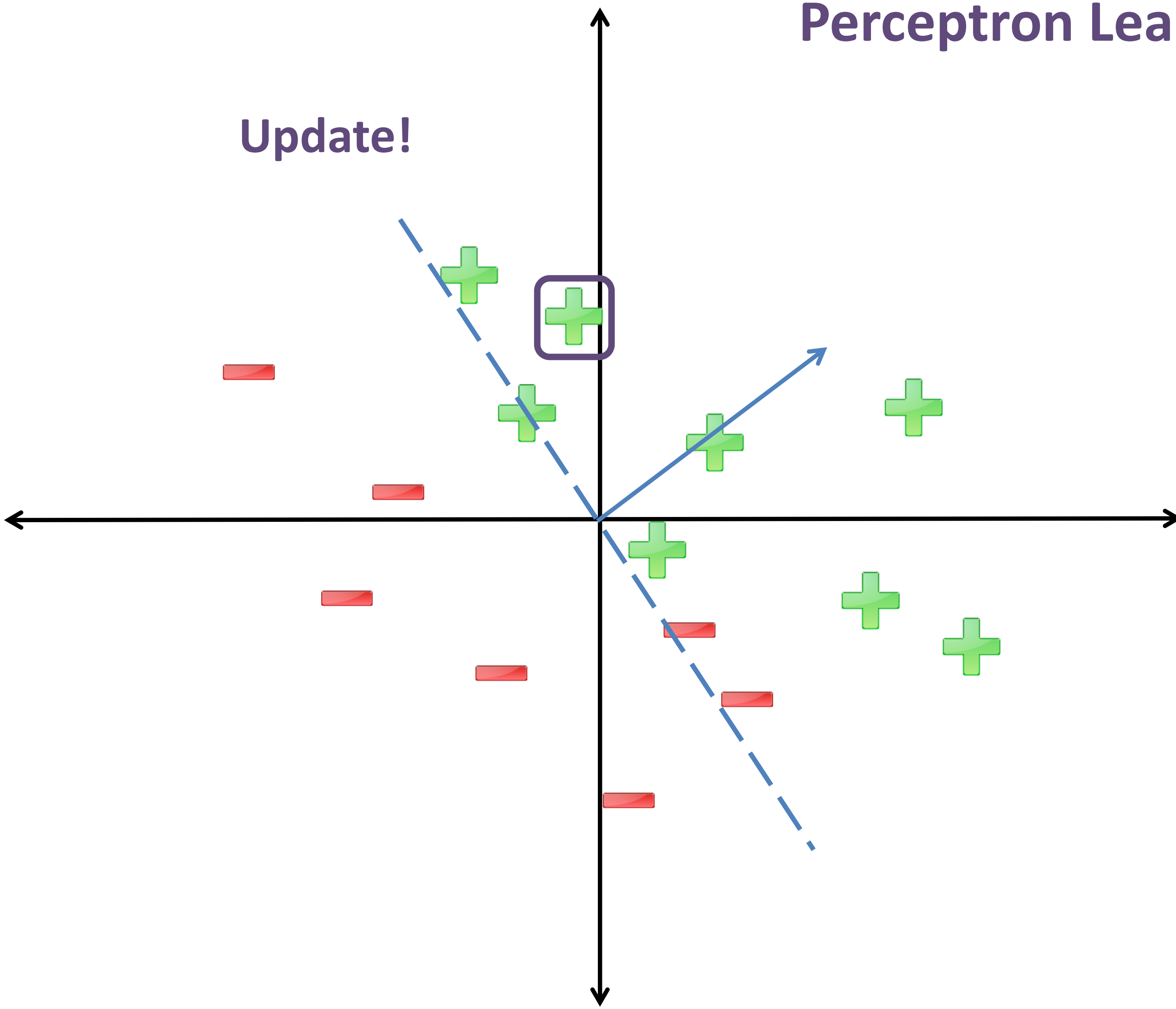


Perceptron Learning

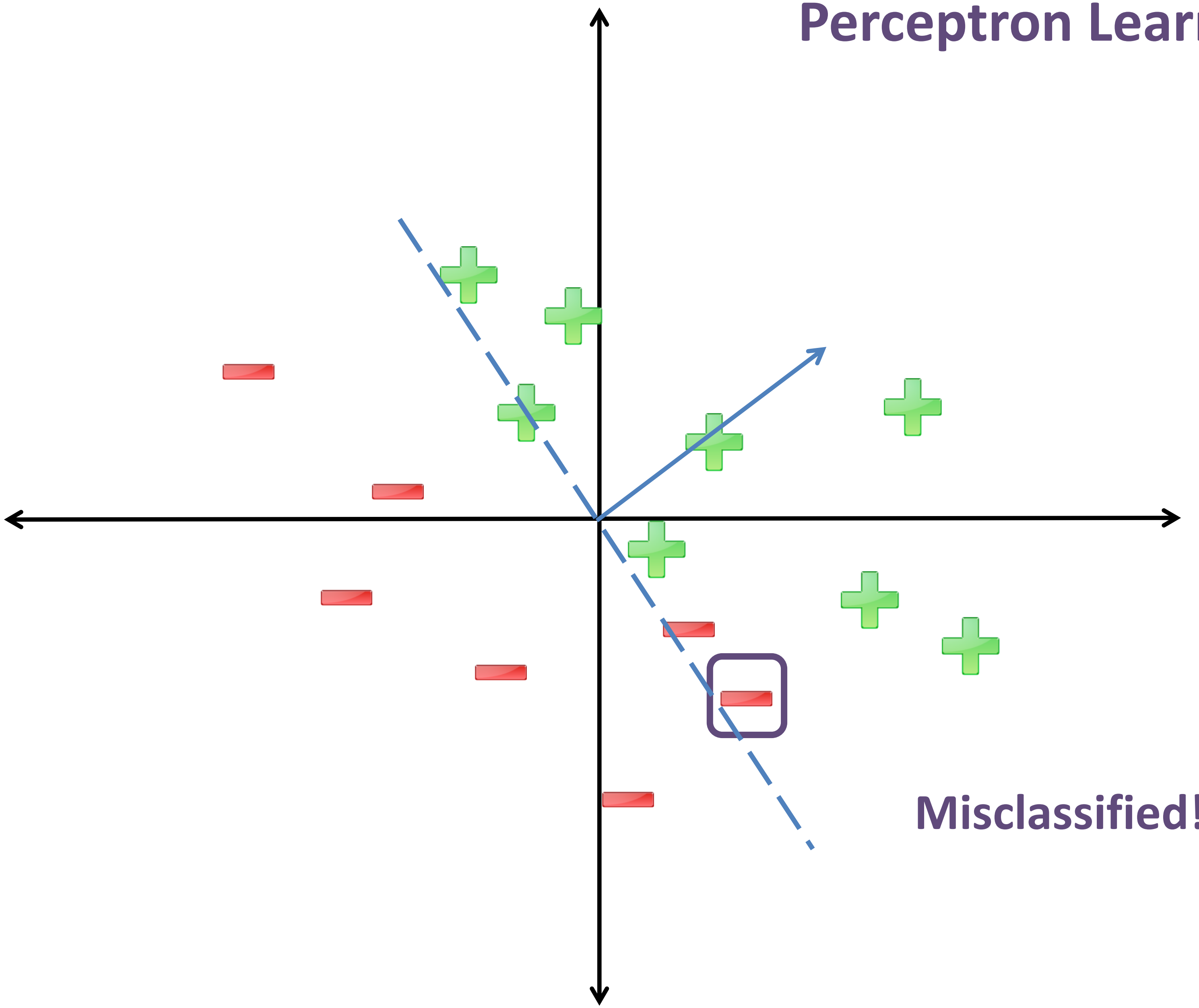
Update!



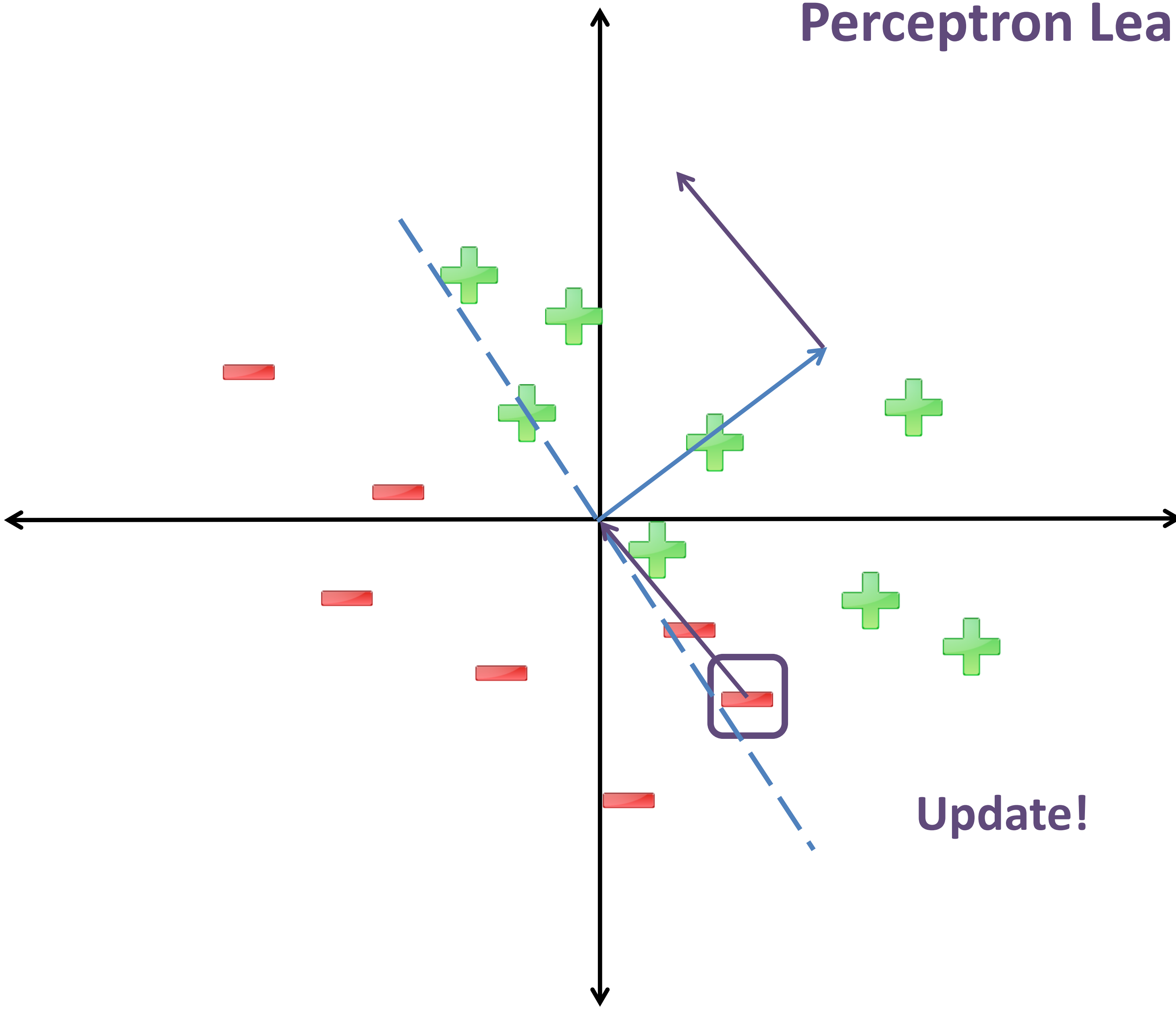
Perceptron Learning



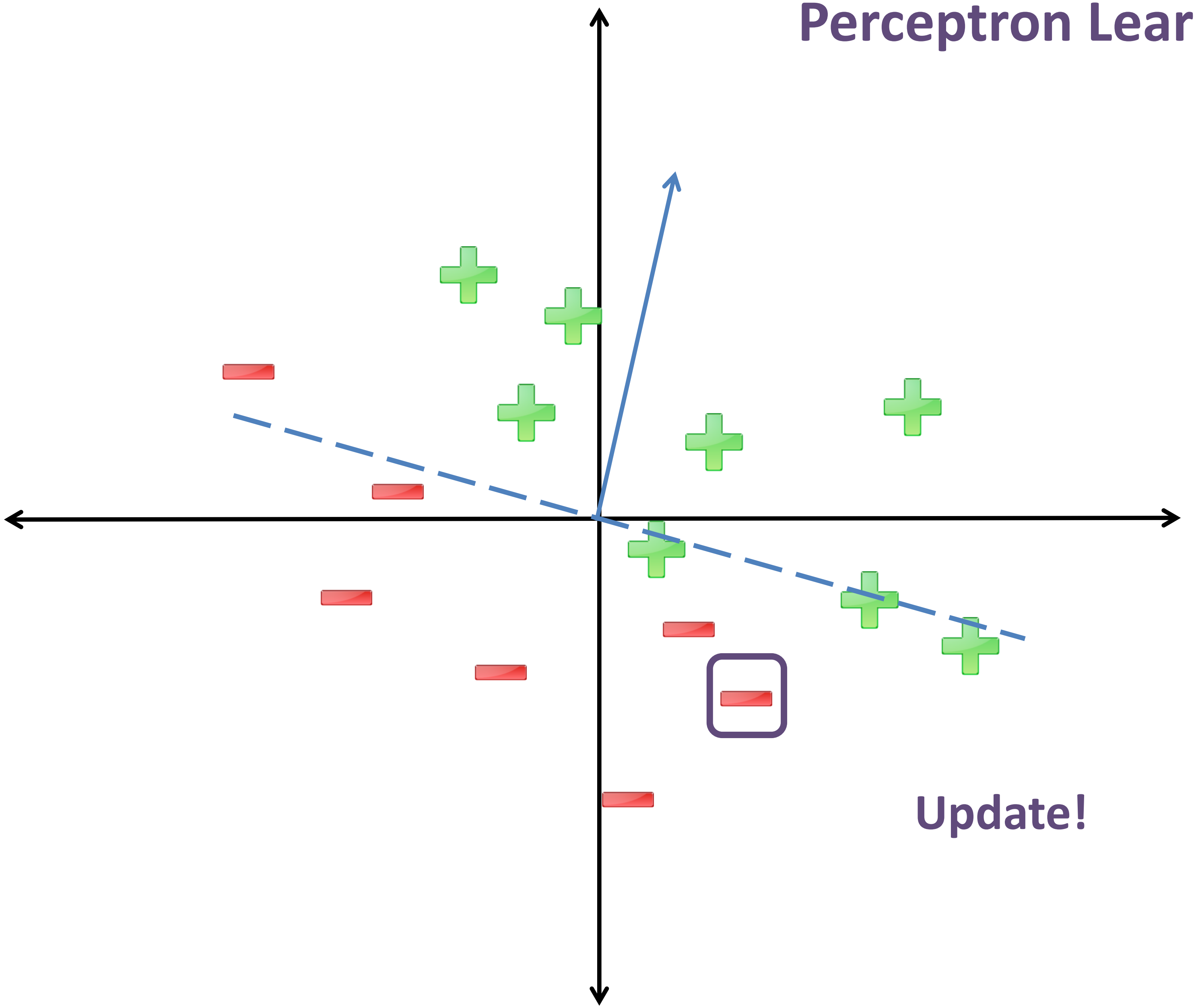
Perceptron Learning



Perceptron Learning

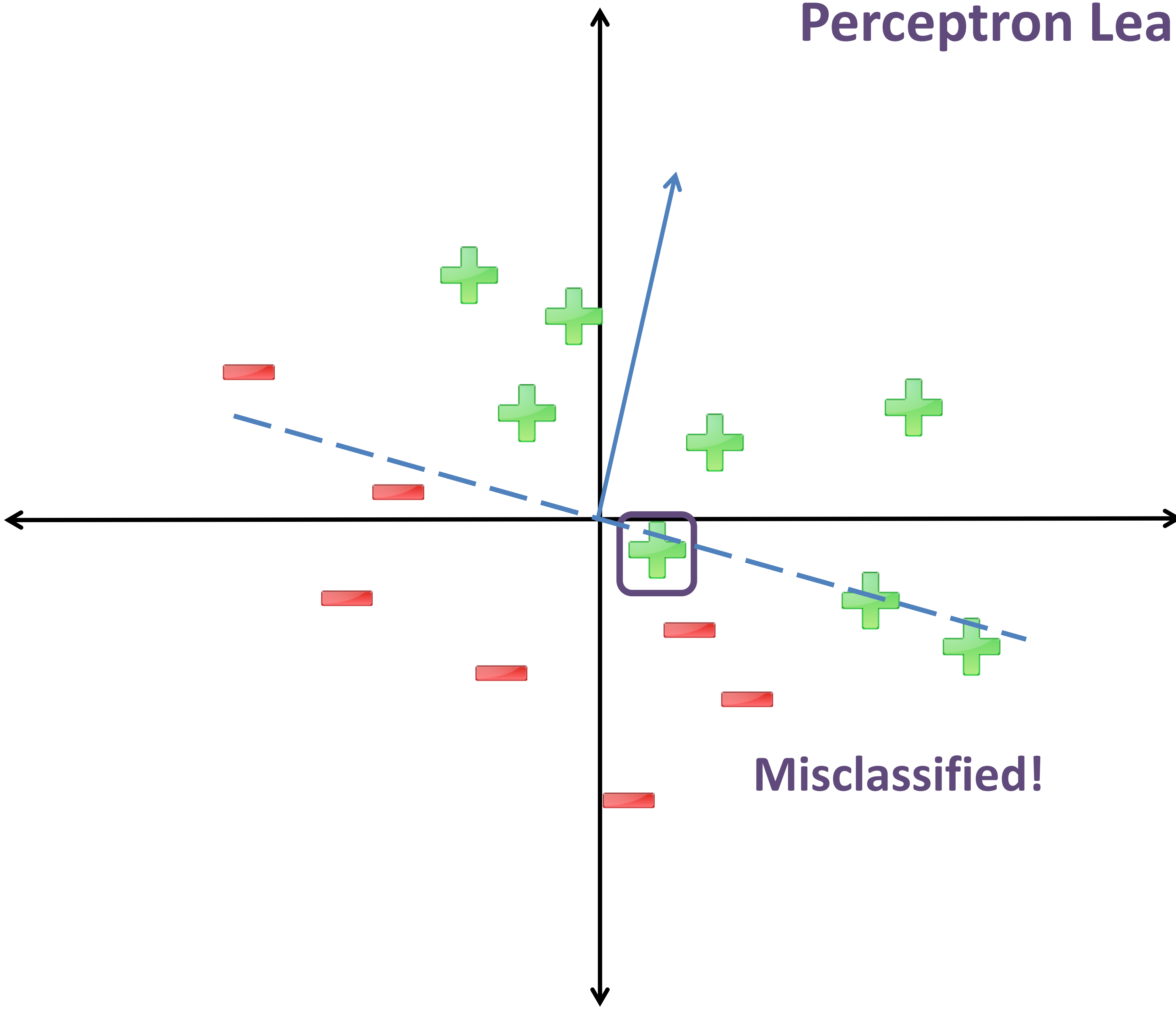


Perceptron Learning



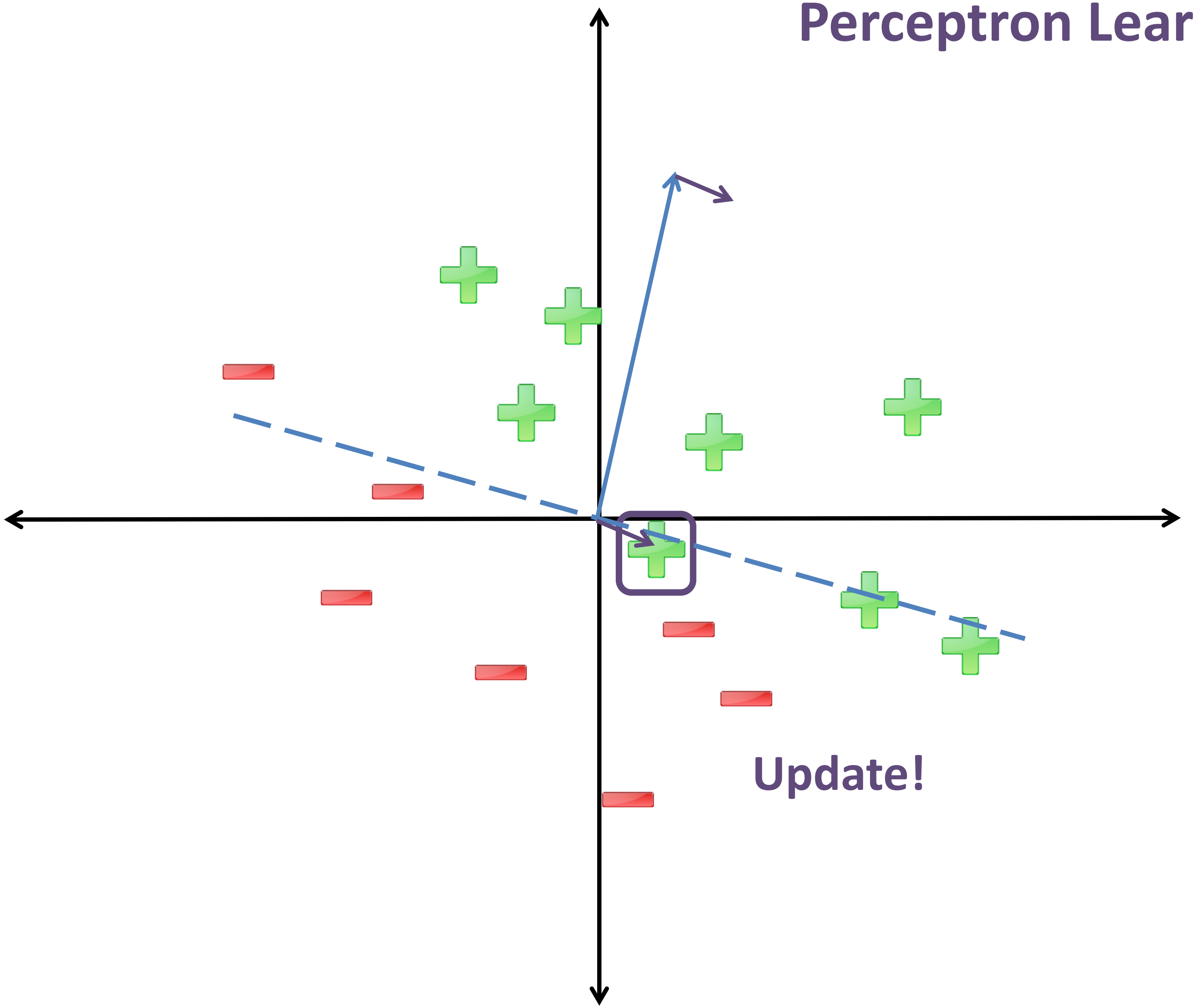
Update!

Perceptron Learning

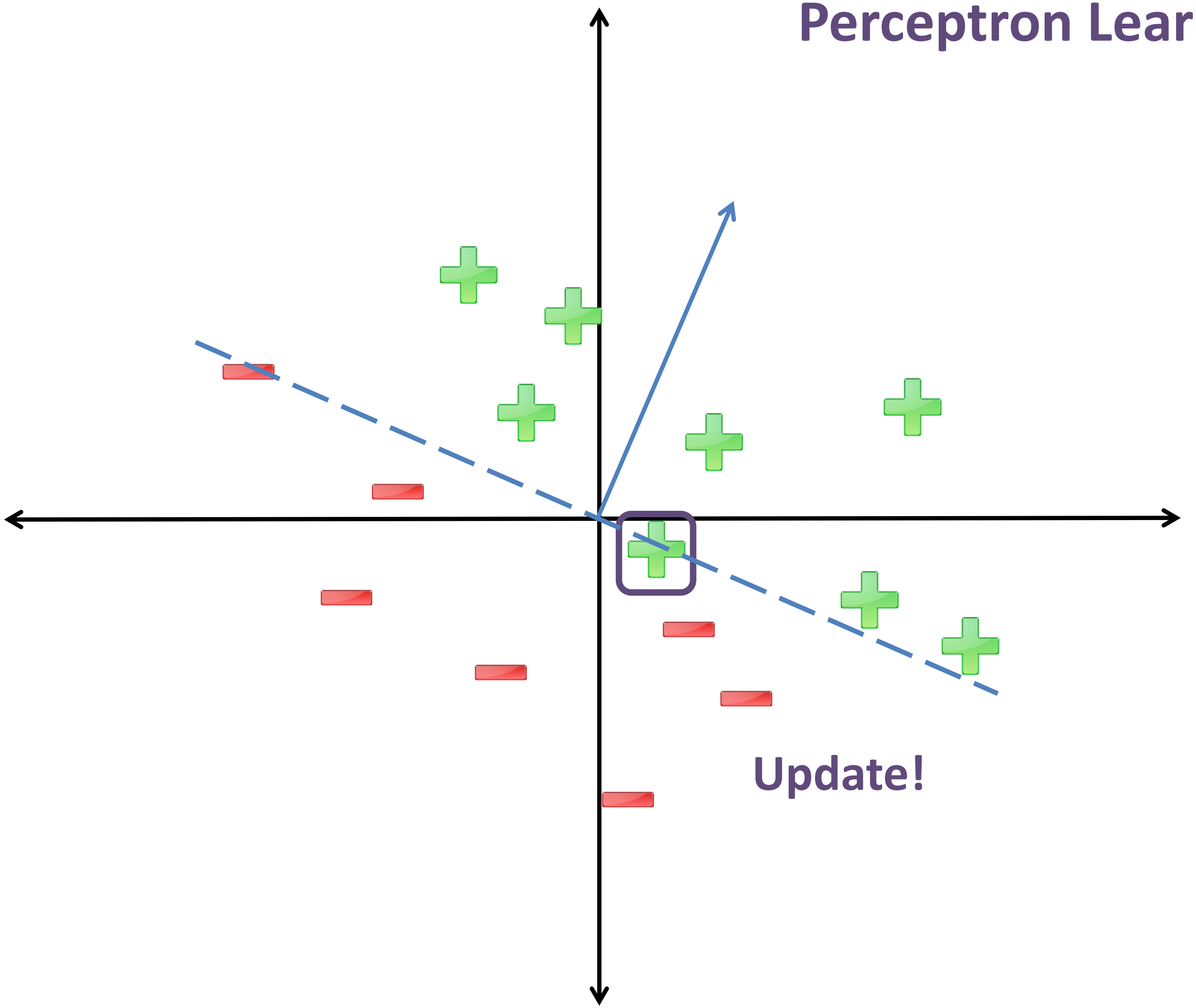


Misclassified!

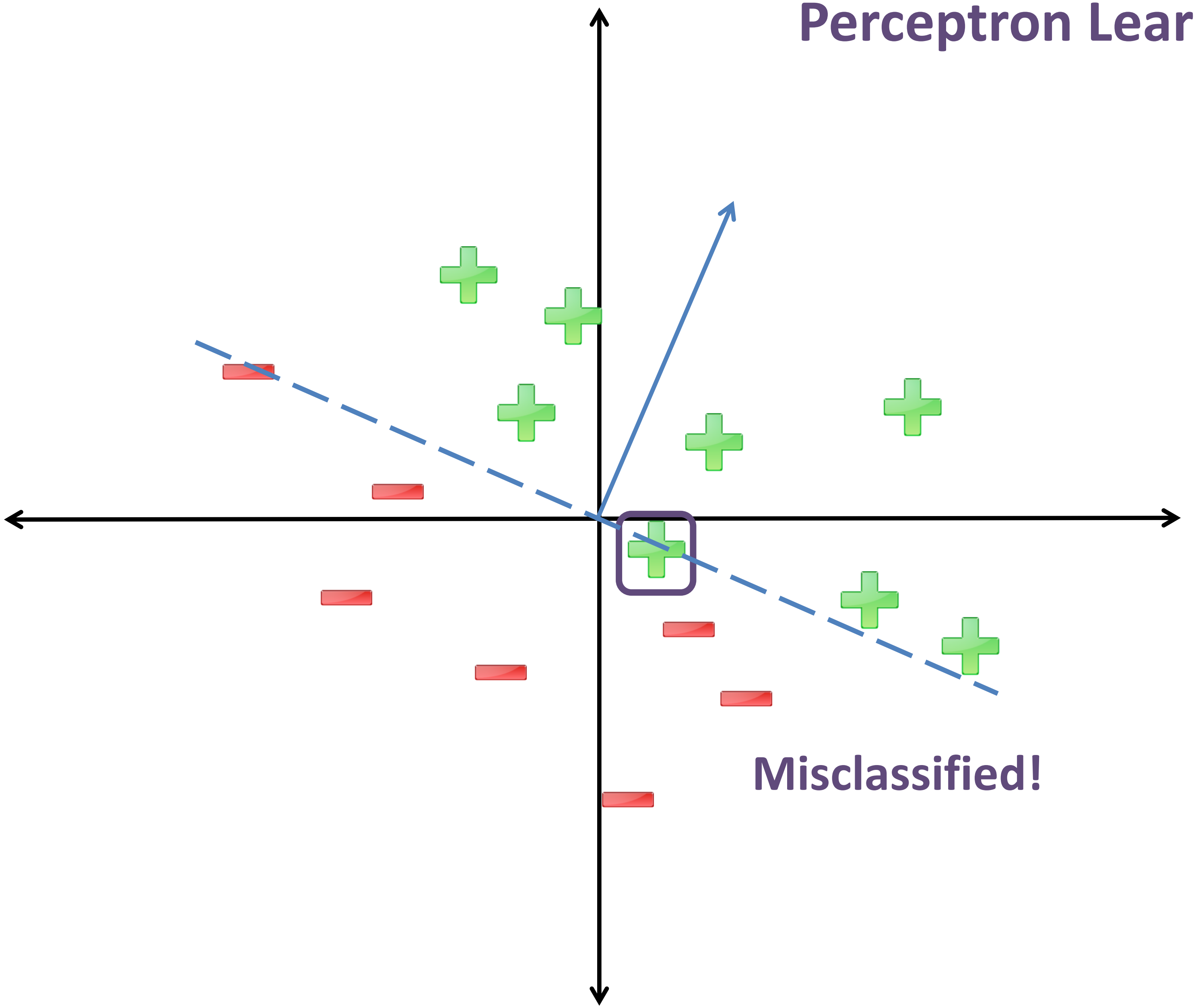
Perceptron Learning



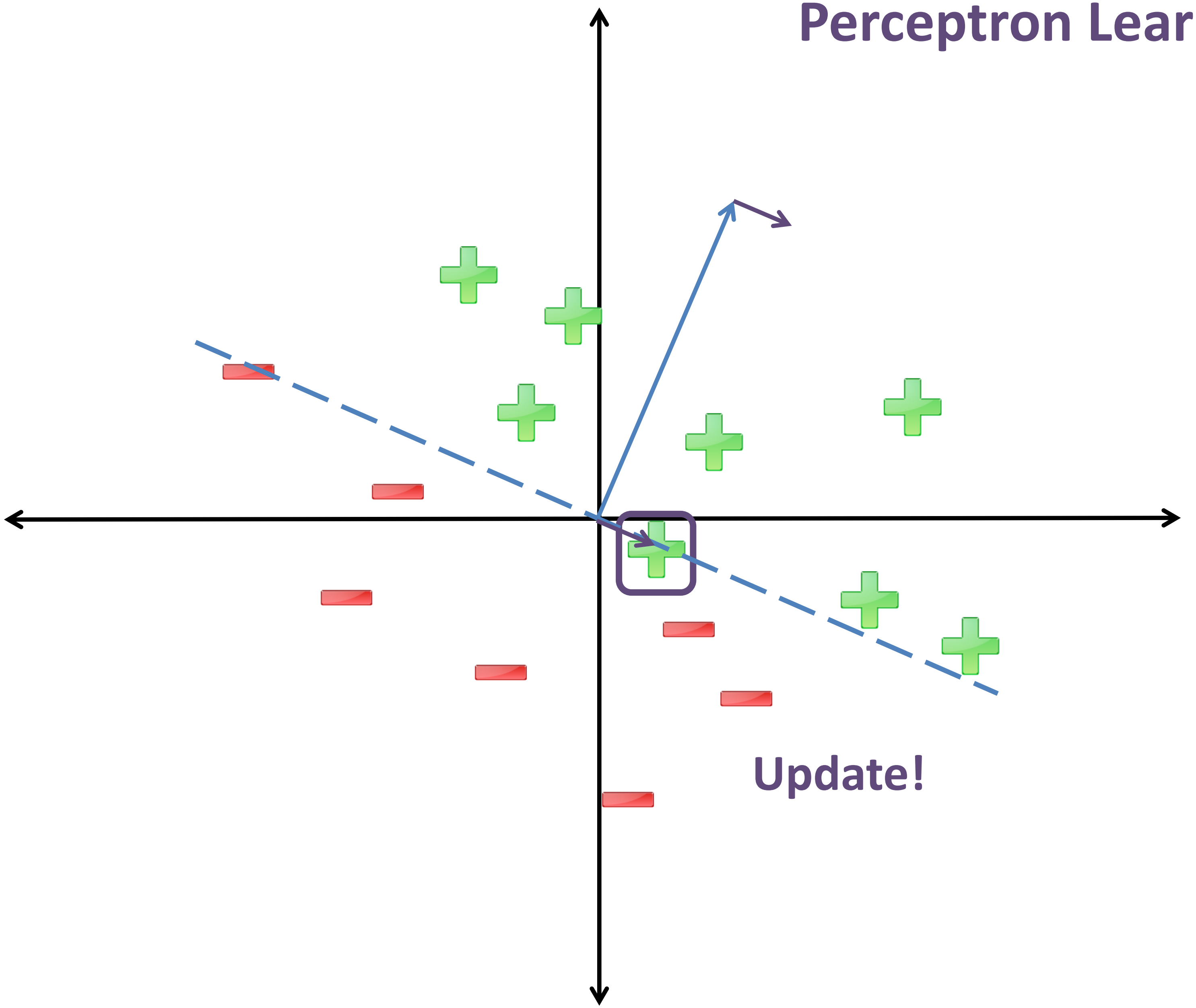
Perceptron Learning



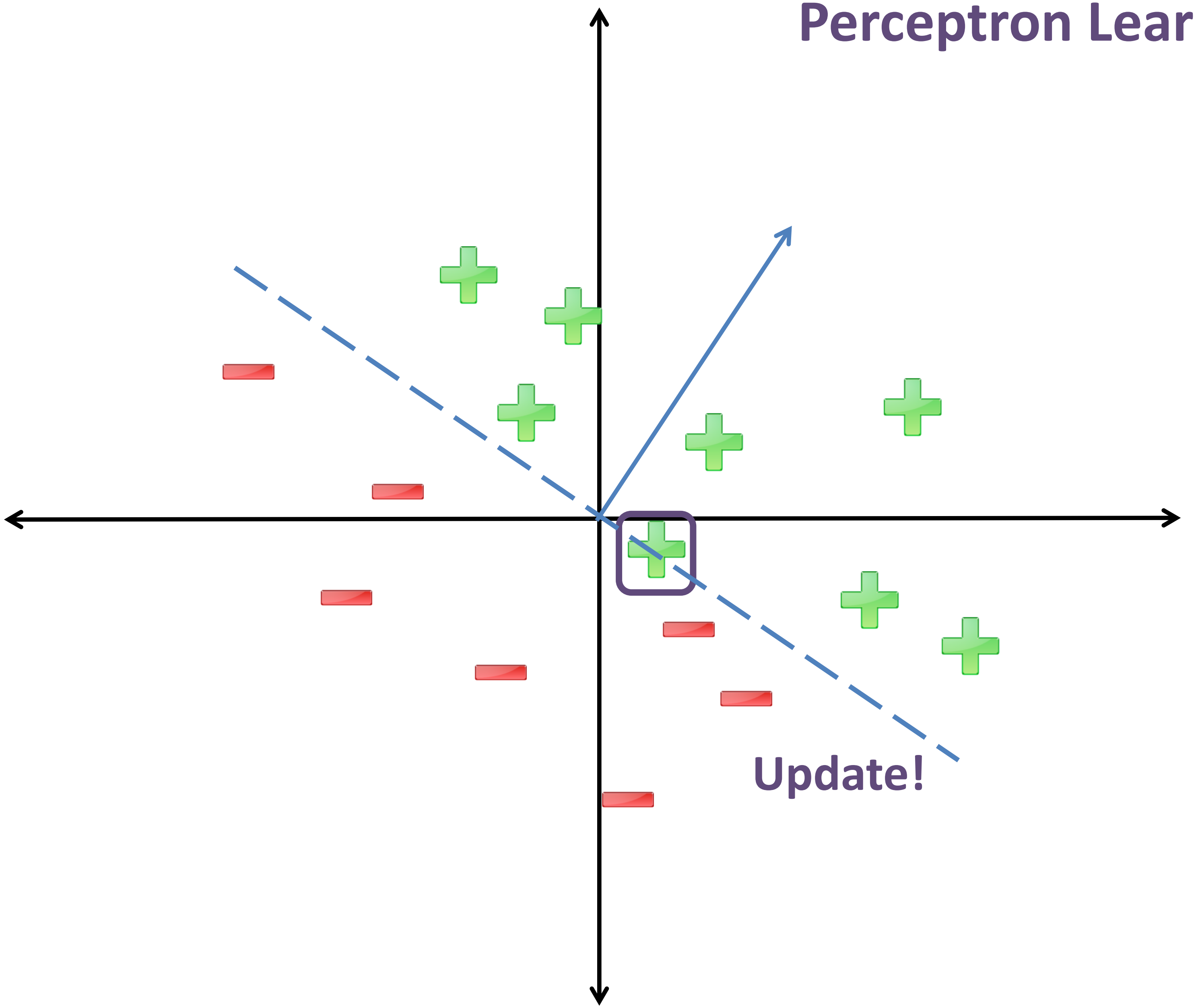
Perceptron Learning



Perceptron Learning

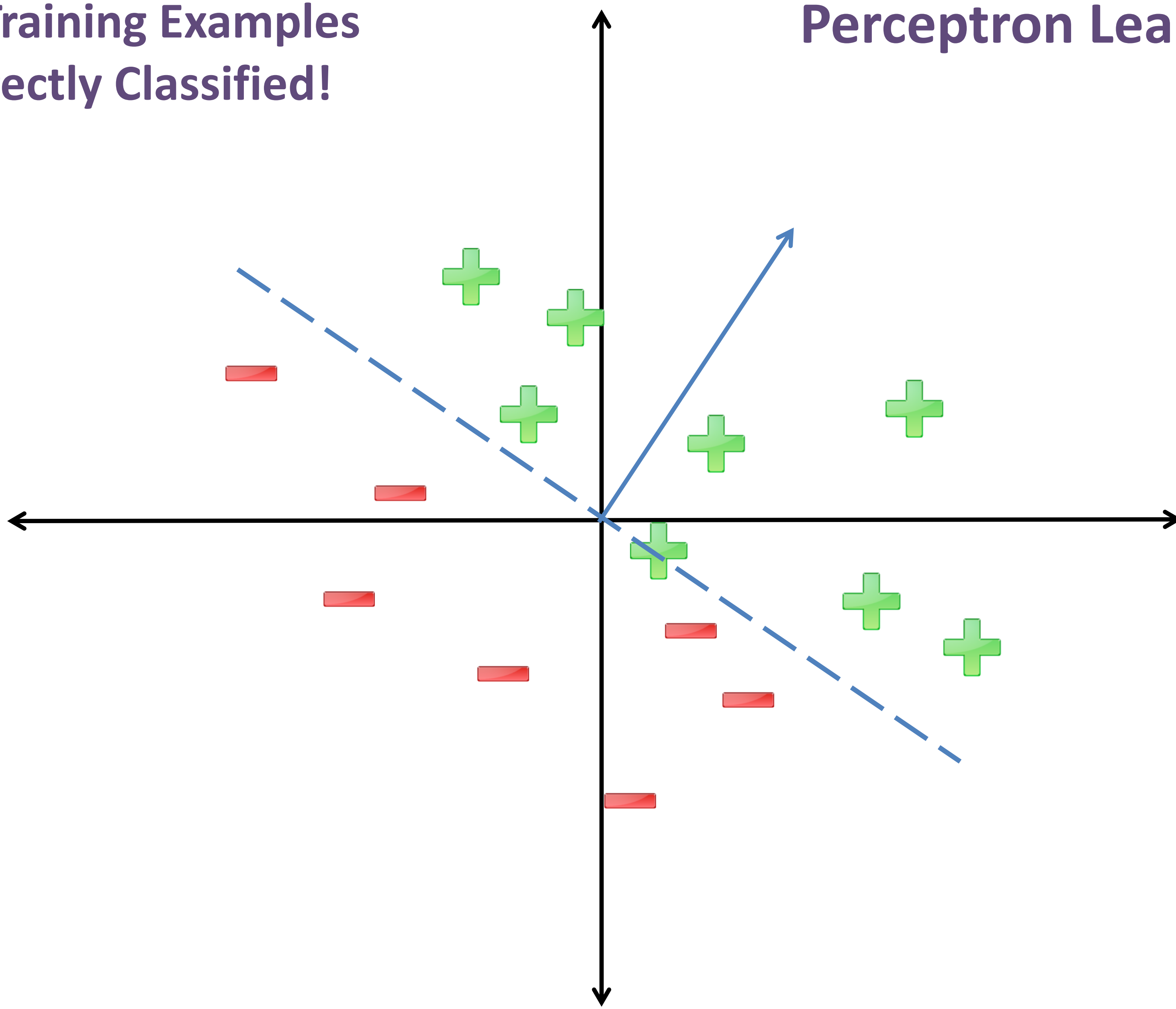


Perceptron Learning



Perceptron Learning

All Training Examples Correctly Classified!



Recap: Perceptron Learning Algorithm
(Linear Classification Model)

- $w^1 = 0$, $b^1 = 0$
- For t = 1 ….
  - Receive example (x,y)
  - If $f(x|w^t,b^t) = y$
    - $[w^{t+1}, b^{t+1}] = [w^t, b^t]$
  - Else
    - $w^{t+1} = w^t + yx$
    - $b^{t+1} = b^t - y$

$$f(x \mid w) = sign(w^T x - b)$$

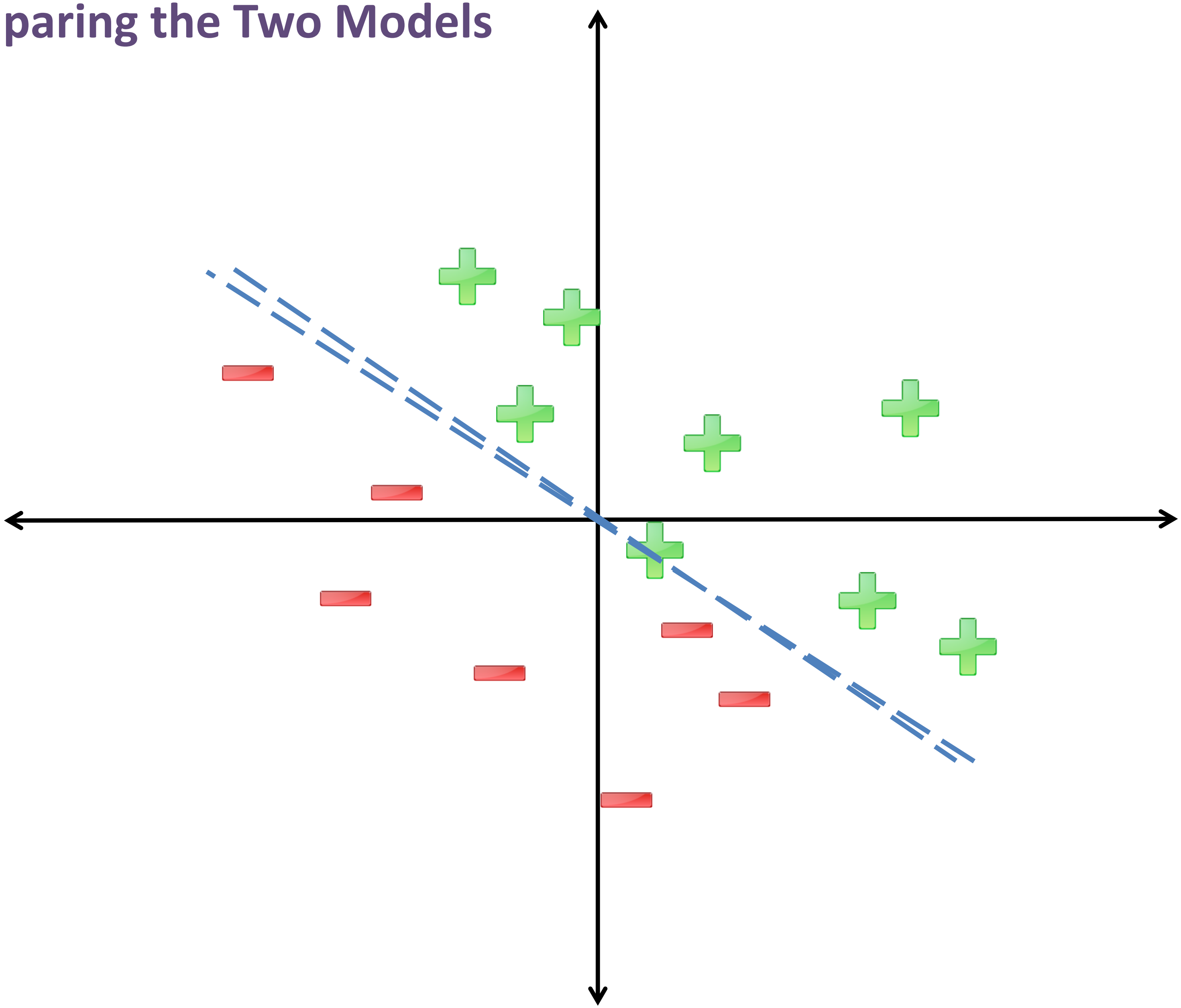
Training Set:

$$S = \{(x_i, y_i)\}_{i=1}^{N}$$

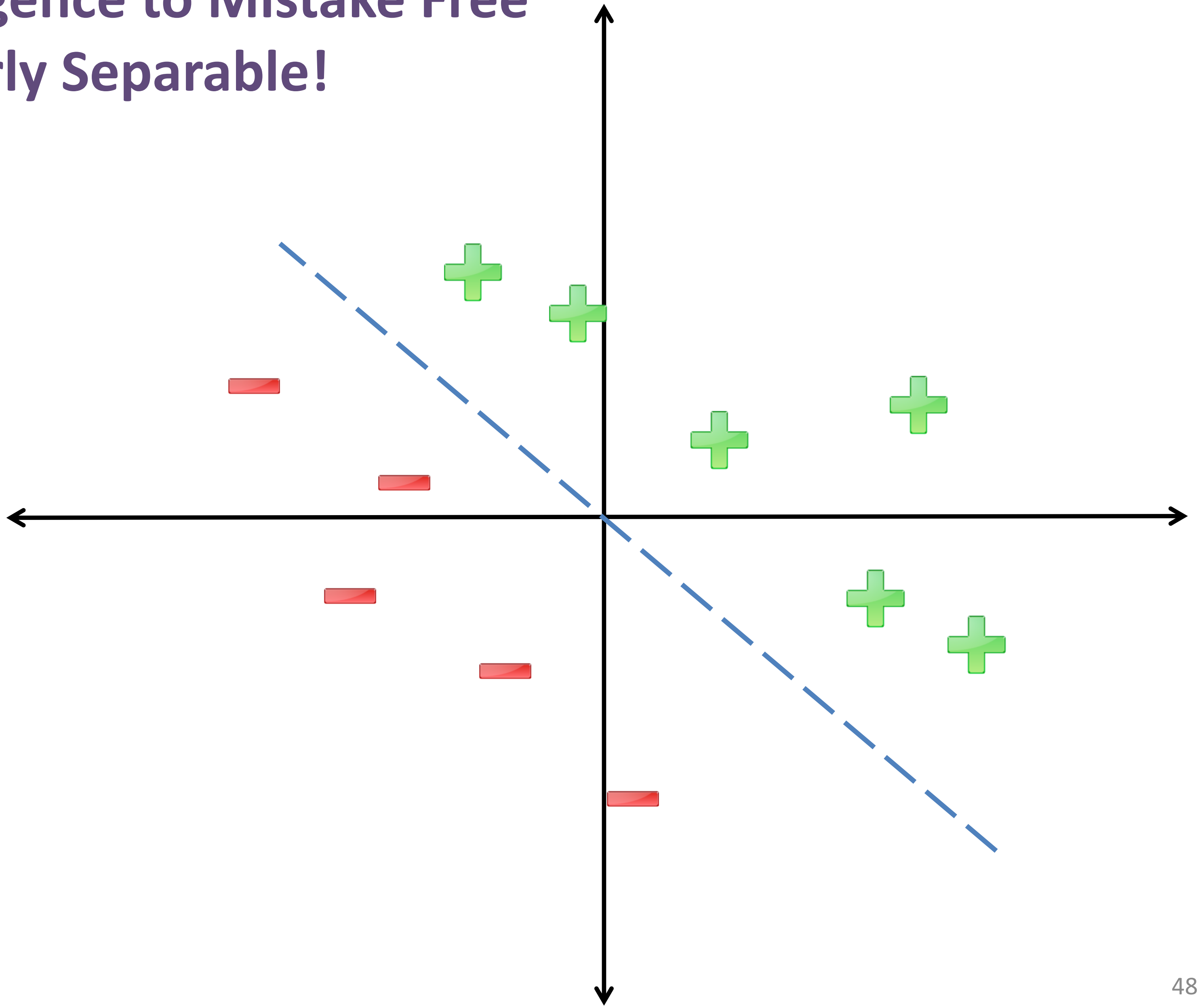
$$y \in \{+1, -1\}$$

Go through training set in arbitrary order (e.g., randomly)

Comparing the Two Models

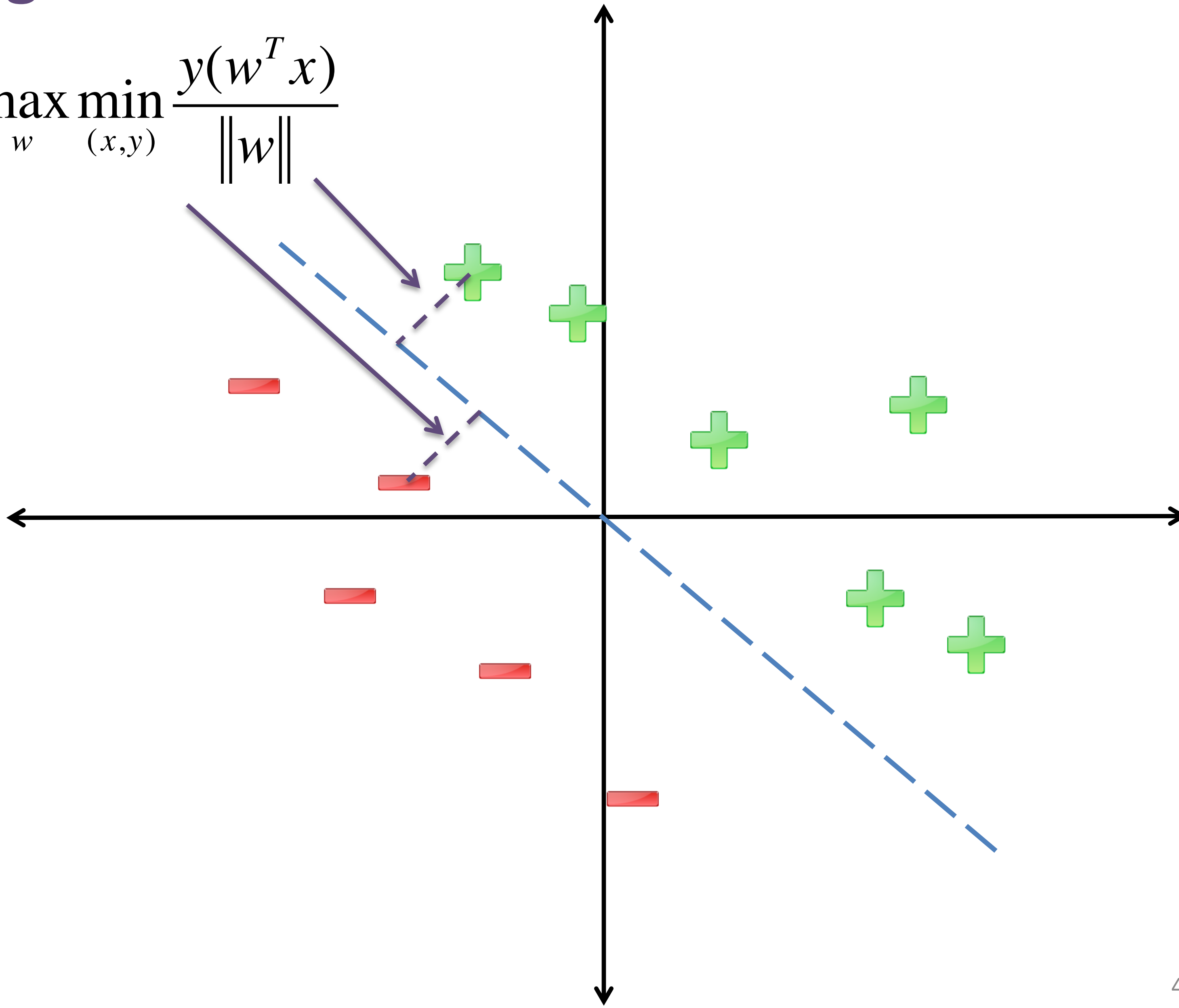


Convergence to Mistake Free = Linearly Separable!



Margin

$$\gamma = \max_{w} \min_{(x,y)} \frac{y(w^T x)}{\|w\|}$$



Linear Separability

- A classification problem is Linearly Separable:
  - Exists w with perfect classification accuracy

- Separable with Margin γ:

$$\gamma = \max_{w} \min_{(x,y)} \frac{y(w^T x)}{\|w\|}$$

- Linearly Separable: γ > 0

Perceptron Mistake Bound

Holds for any ordering of training examples!

"Radius" of Feature Space

$$R = \max_x \|x\|$$

#Mistakes Bounded By: $\dfrac{R^2}{\gamma^2}$

Margin

**If Linearly Separable

More Details: http://www.cs.nyu.edu/~mohri/pub/pmb.pdf

In the Real World…

- Most problems are NOT linearly separable!
- May never converge…
- So what to do?
- **Use validation set!**

Early Stopping via Validation

- Run Perceptron Learning on Training Set
- Evaluate current model on Validation Set
- Terminate when validation accuracy stops improving

https://en.wikipedia.org/wiki/Early_stopping

Online Learning vs Batch Learning

- Online Learning:
  - Receive a stream of data (x,y)
  - Make incremental updates (typically)
  - Perceptron Learning is an instance of Online Learning
- Batch Learning
  - Given all the data up front
  - Can use online learning algorithms for batch learning
  - E.g., stream the data to the learning algorithm

https://en.wikipedia.org/wiki/Online_machine_learning

Recap: Perceptron

- One of the first machine learning algorithms
- **Benefits:**
  - Simple and fast
  - Clean analysis
- **Drawbacks:**
  - Might not converge to a very good model
  - What is the objective function?

(Stochastic) Gradient Descent

Back to Optimizing Objective Functions

- Training Data: $S = \{(x_i, y_i)\}_{i=1}^{N}$ $\quad x \in R^D$ $\quad y \in \{-1,+1\}$
- Model Class: $f(x \mid w,b) = w^T x - b$ **Linear Models**
- Loss Function: $L(a,b) = (a-b)^2$ **Squared Loss**
- Learning Objective: $\underset{w,b}{\operatorname{argmin}} \sum_{i=1}^{N} L\left(y_i, f(x_i \mid w,b)\right)$

Optimization Problem

Back to Optimizing Objective Functions

$$\underset{w,b}{\operatorname{argmin}} L(w,b) \equiv \sum_{i=1}^{N} L\left(y_i, f(x_i \mid w,b)\right)$$

- Typically, requires optimization algorithm.

- Simplest: **Gradient Descent**

- This Lecture: stick with squared loss
  - Talk about various loss functions next lecture

Gradient Review for Squared Loss

$$\partial_w L(w,b) = \partial_w \sum_{i=1}^{N} L\left(y_i, f(x_i \mid w,b)\right)$$

$$= \sum_{i=1}^{N} \partial_w L\left(y_i, f(x_i \mid w,b)\right)$$

Linearity of Differentiation

$$= \sum_{i=1}^{N} -2(y_i - f(x_i \mid w,b))\partial_w f(x_i \mid w,b)$$

$$L(a,b) = (a-b)^2$$

Chain Rule

$$= \sum_{i=1}^{N} -2(y_i - f(x_i \mid w,b))x_i$$

$$f(x \mid w,b) = w^T x - b$$

Gradient Descent

- Initialize: $w^1 = 0$, $b^1 = 0$
- For t = 1…

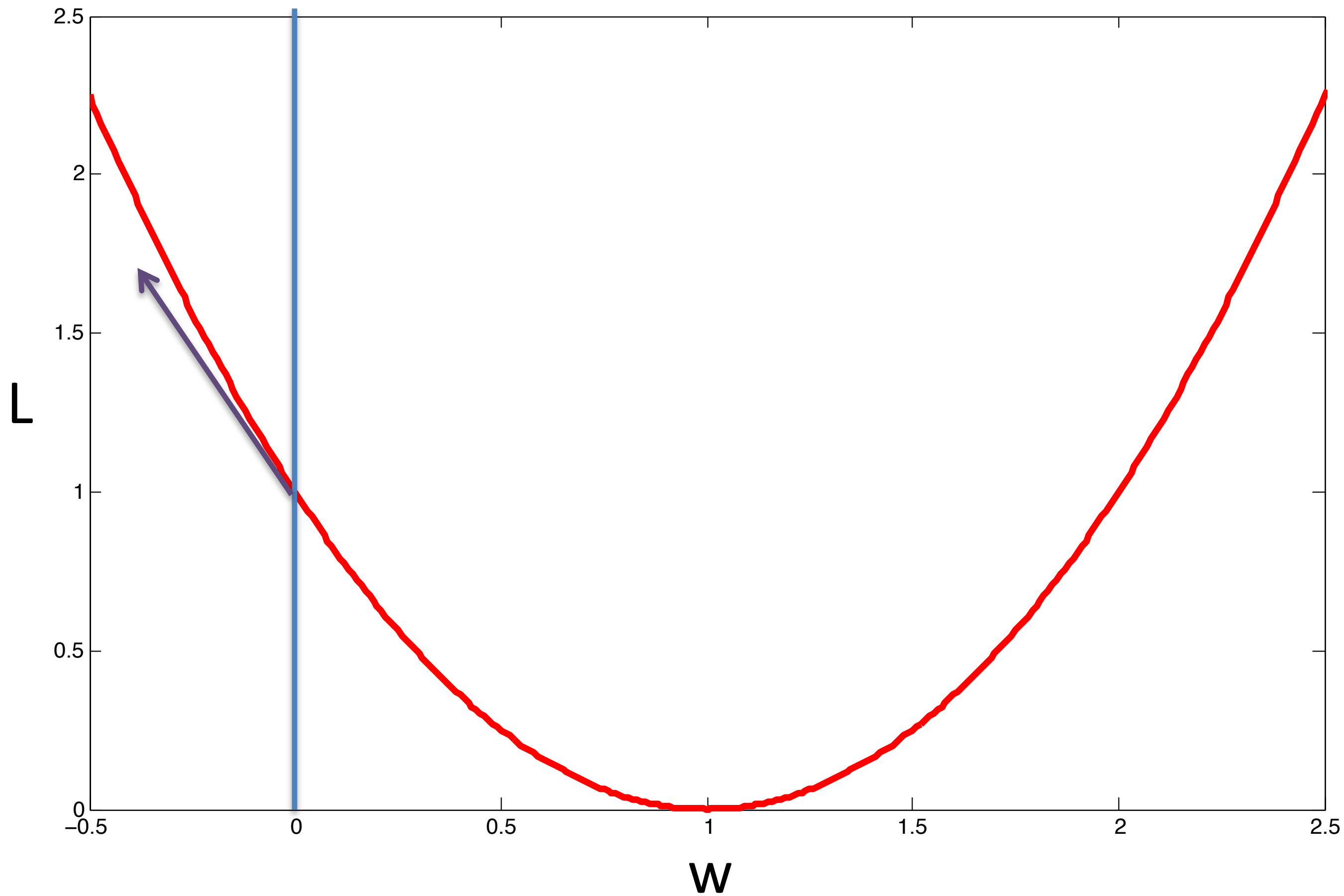
$$w^{t+1} = w^t - \eta^{t+1} \partial_w L(w^t, b^t)$$

$$b^{t+1} = b^t - \eta^{t+1} \partial_b L(w^t, b^t)$$

"Step Size"

How to Choose Step Size?

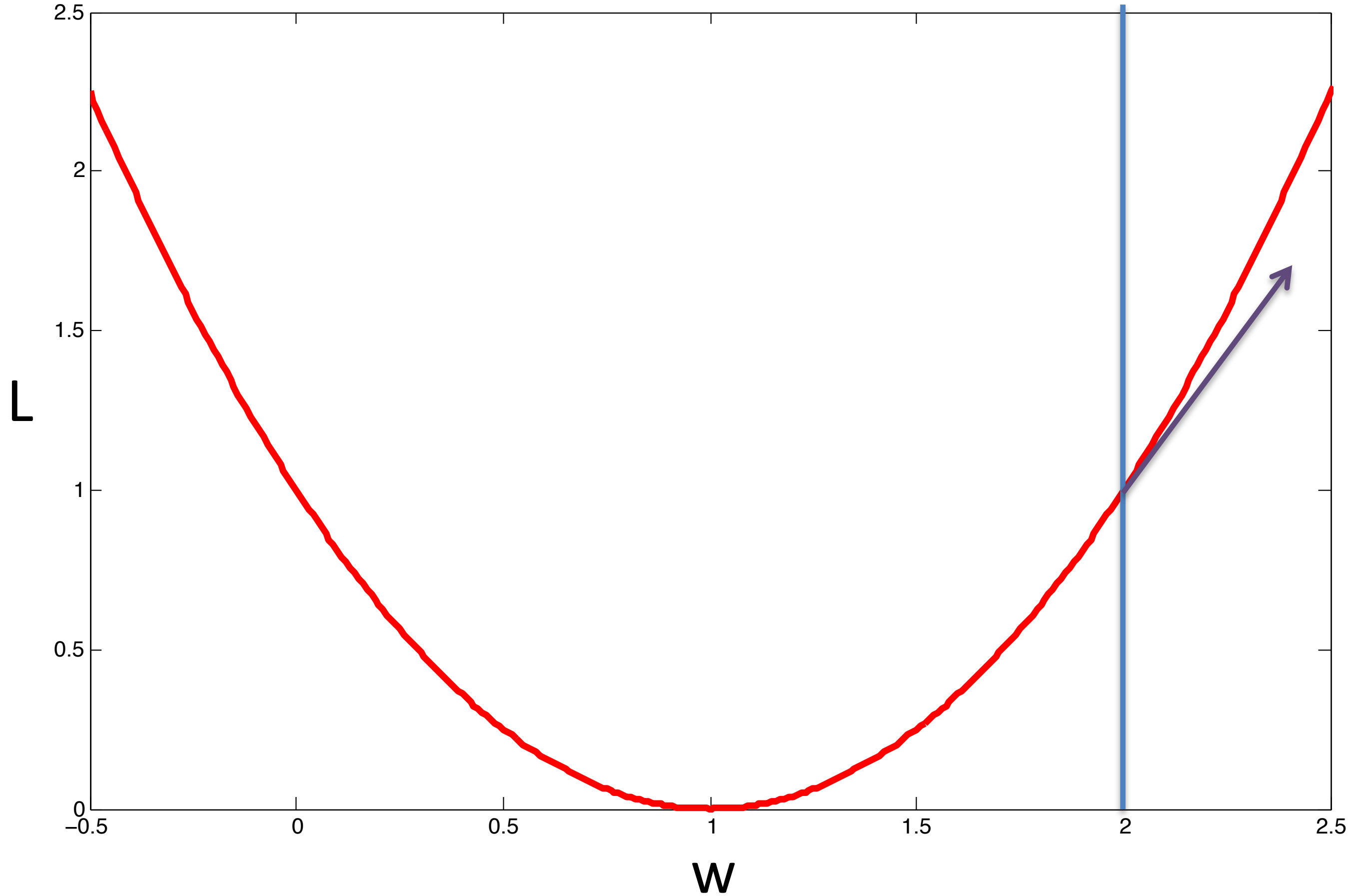
$$\eta = 1 \qquad\qquad \partial_w L(w) = -2(1-w)$$



How to Choose Step Size?

$\eta = 1$

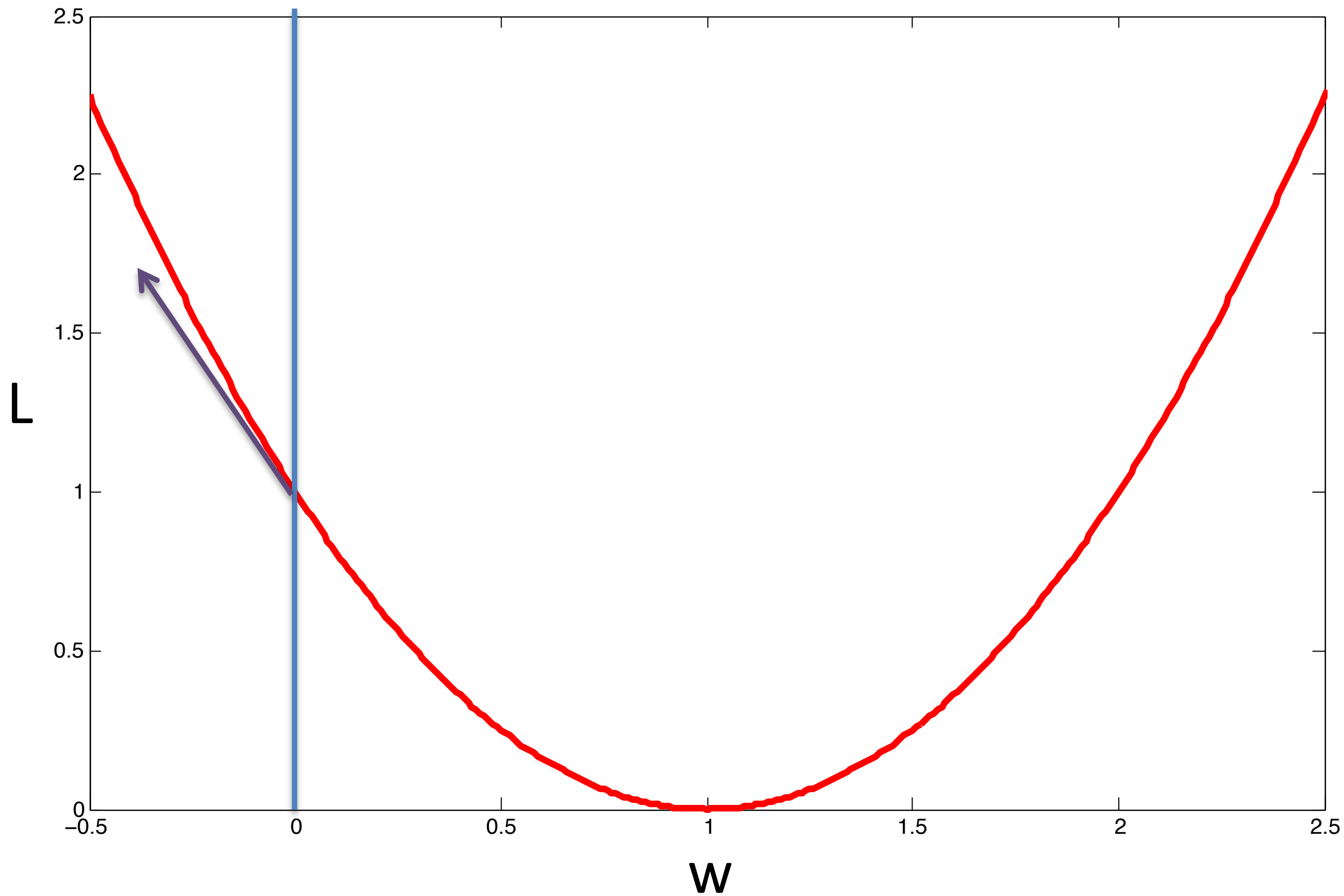
$$\partial_w L(w) = -2(1 - w)$$



How to Choose Step Size?

$\eta = 1$

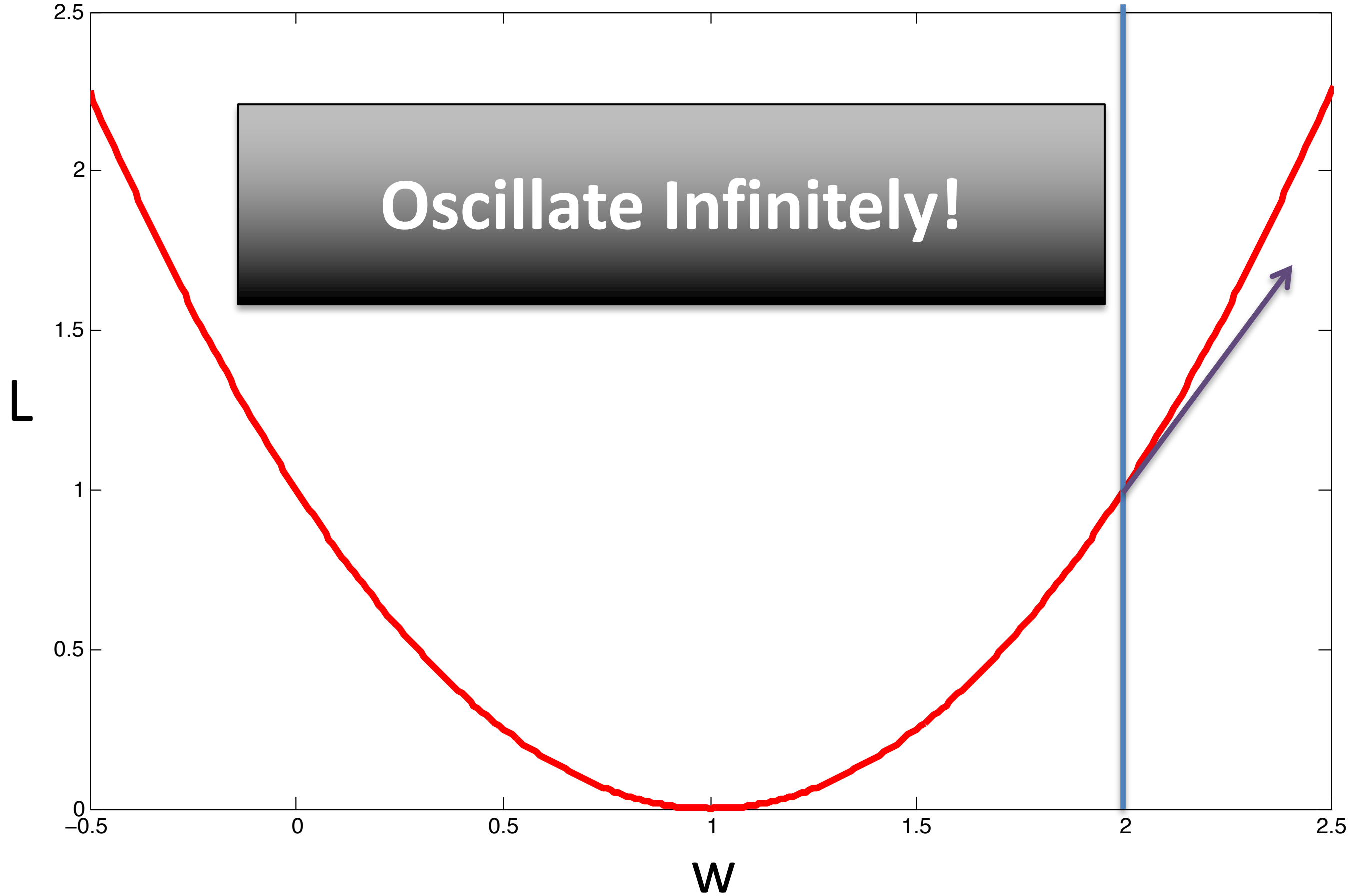
$$\partial_w L(w) = -2(1 - w)$$



How to Choose Step Size?

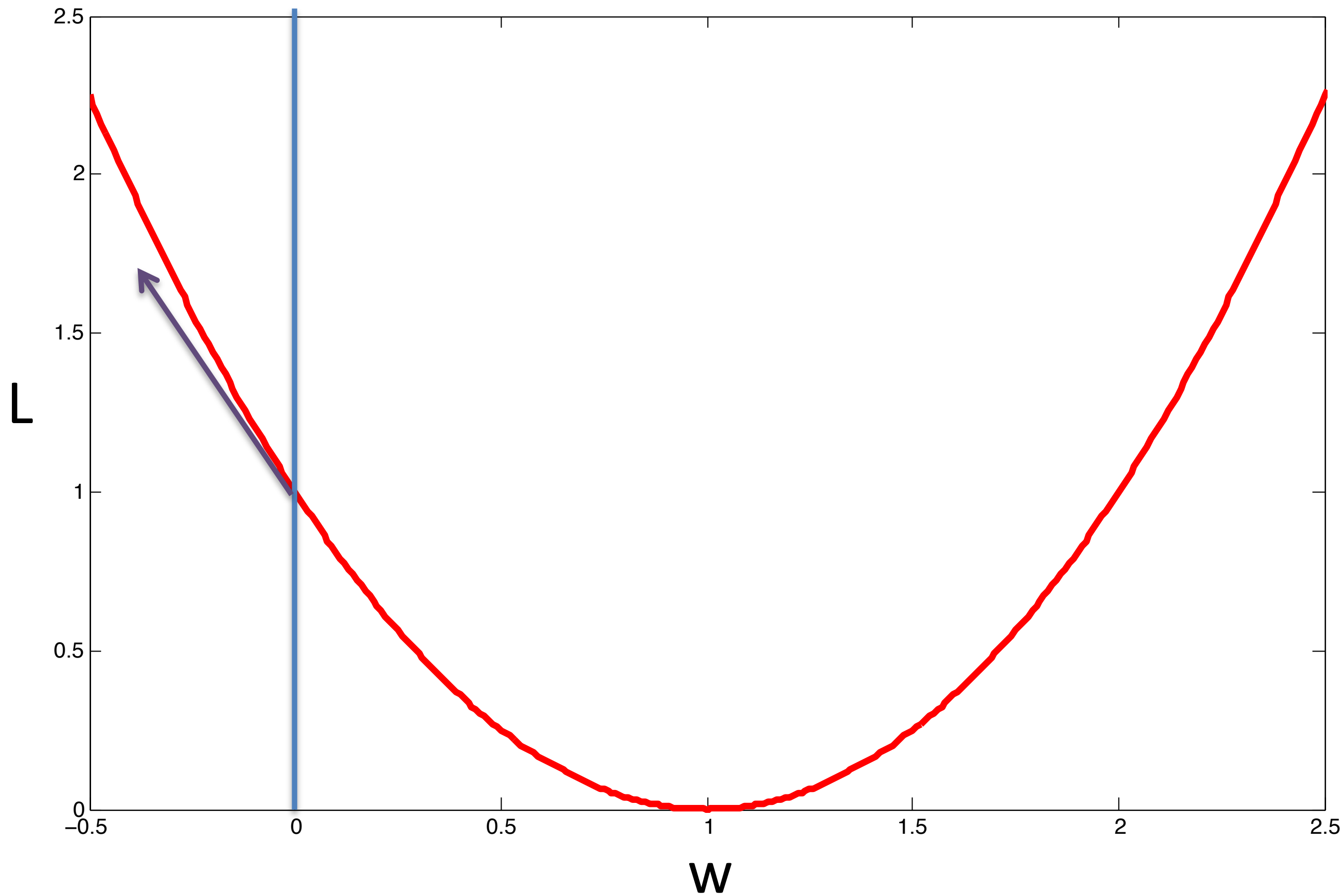
$\eta = 1$

$$\partial_w L(w) = -2(1 - w)$$



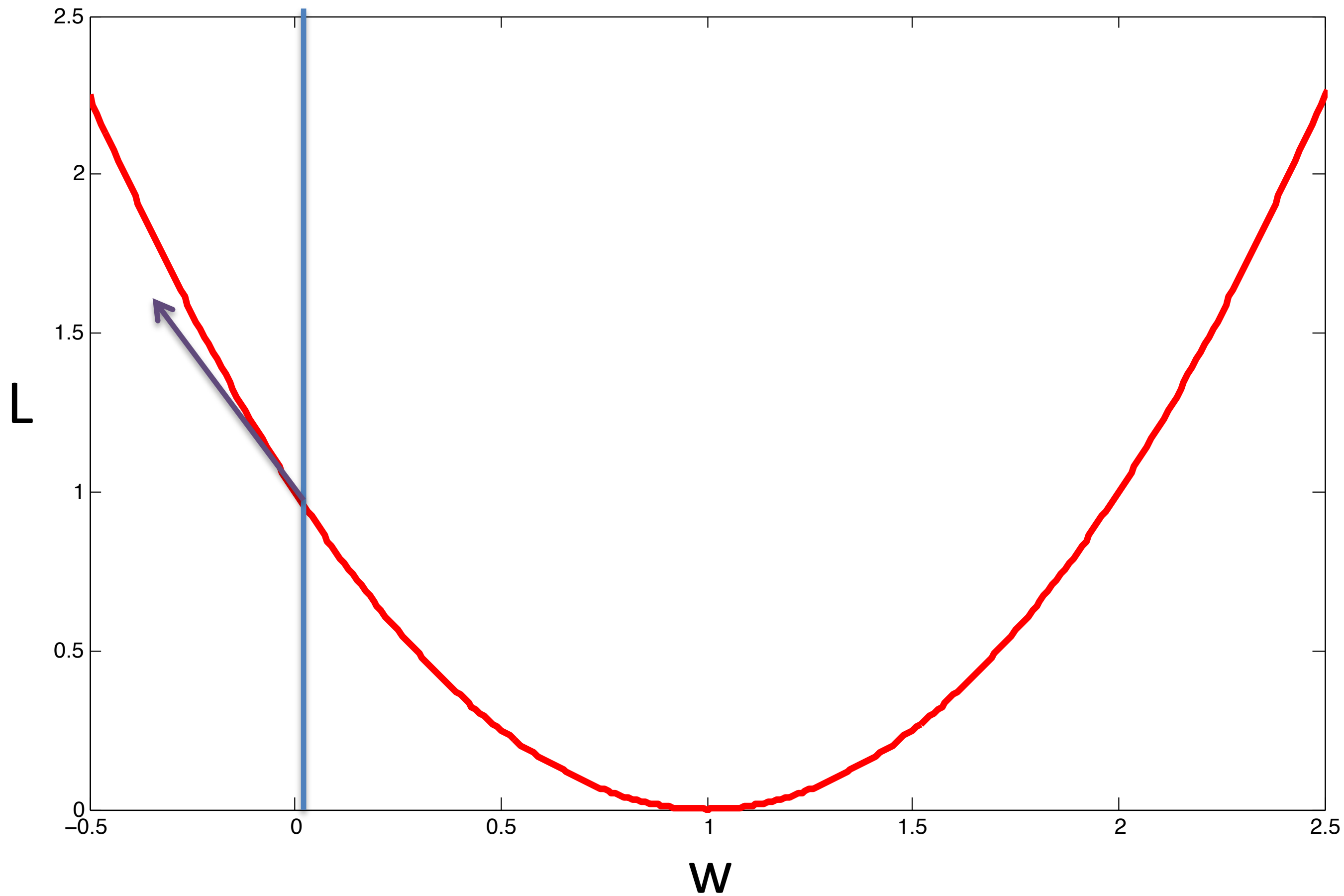
How to Choose Step Size?

$$\eta = 0.0001 \qquad \partial_w L(w) = -2(1 - w)$$



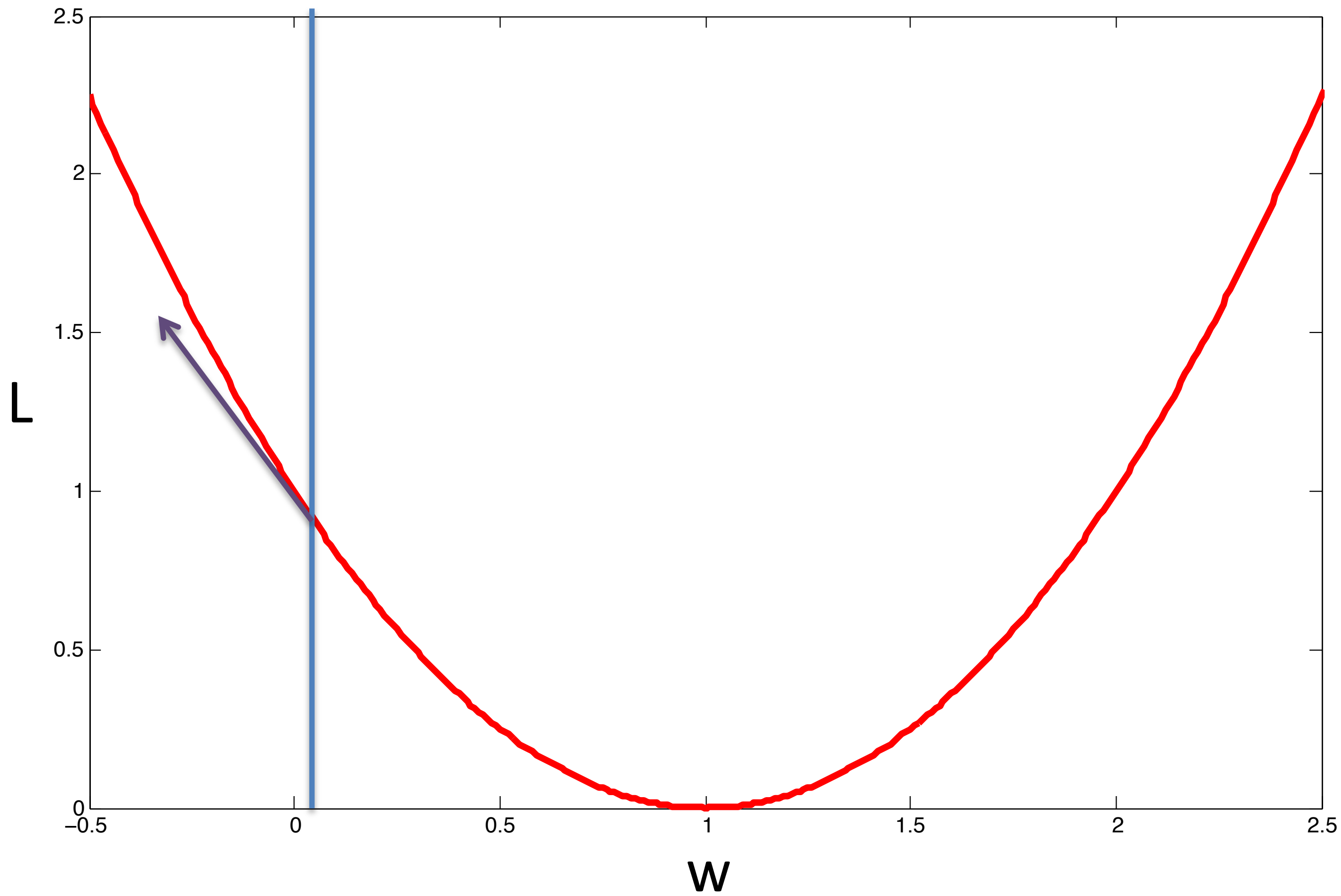
How to Choose Step Size?

$\eta = 0.0001$ $\qquad \partial_w L(w) = -2(1 - w)$



How to Choose Step Size?

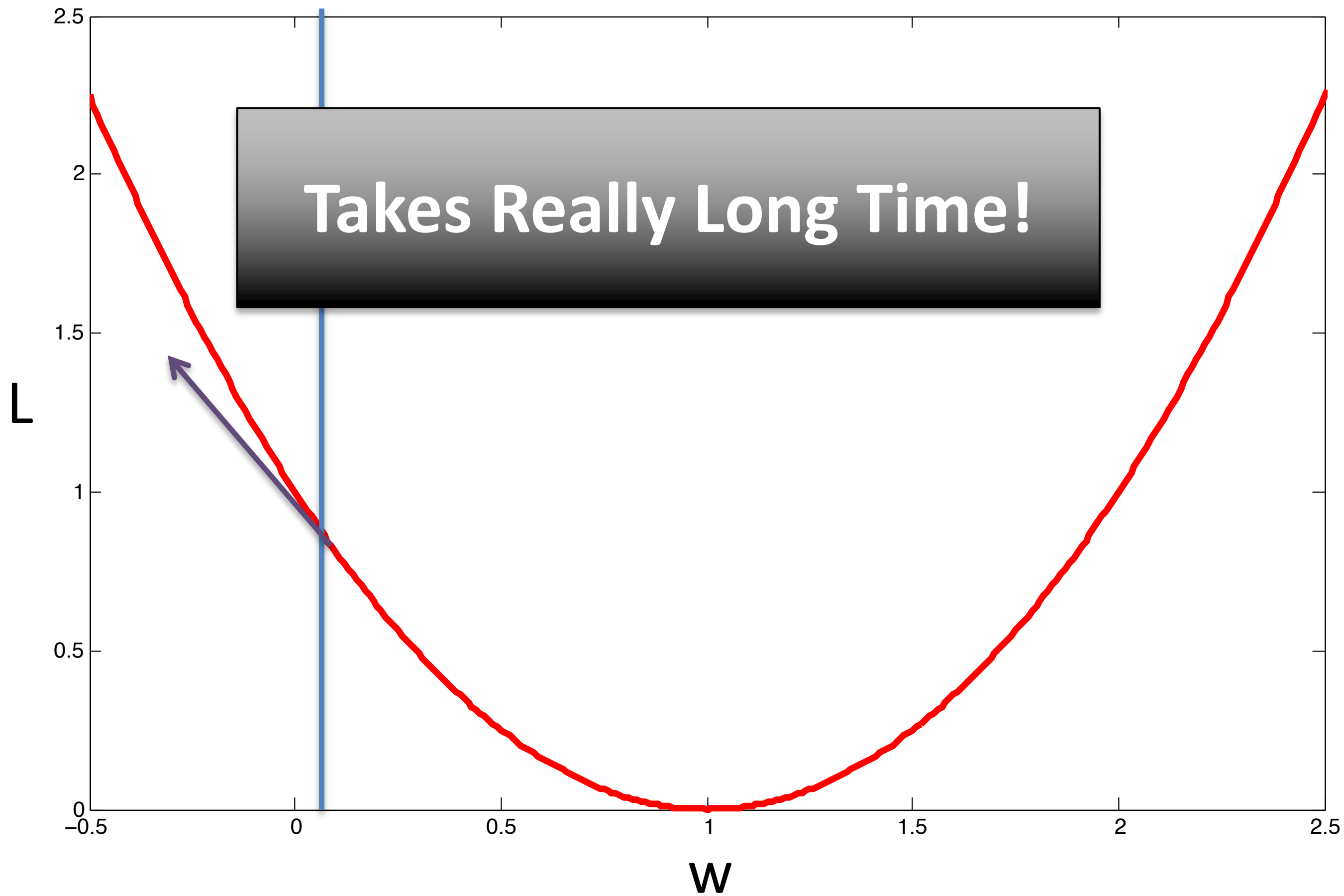
$$\eta = 0.0001 \qquad \partial_w L(w) = -2(1-w)$$



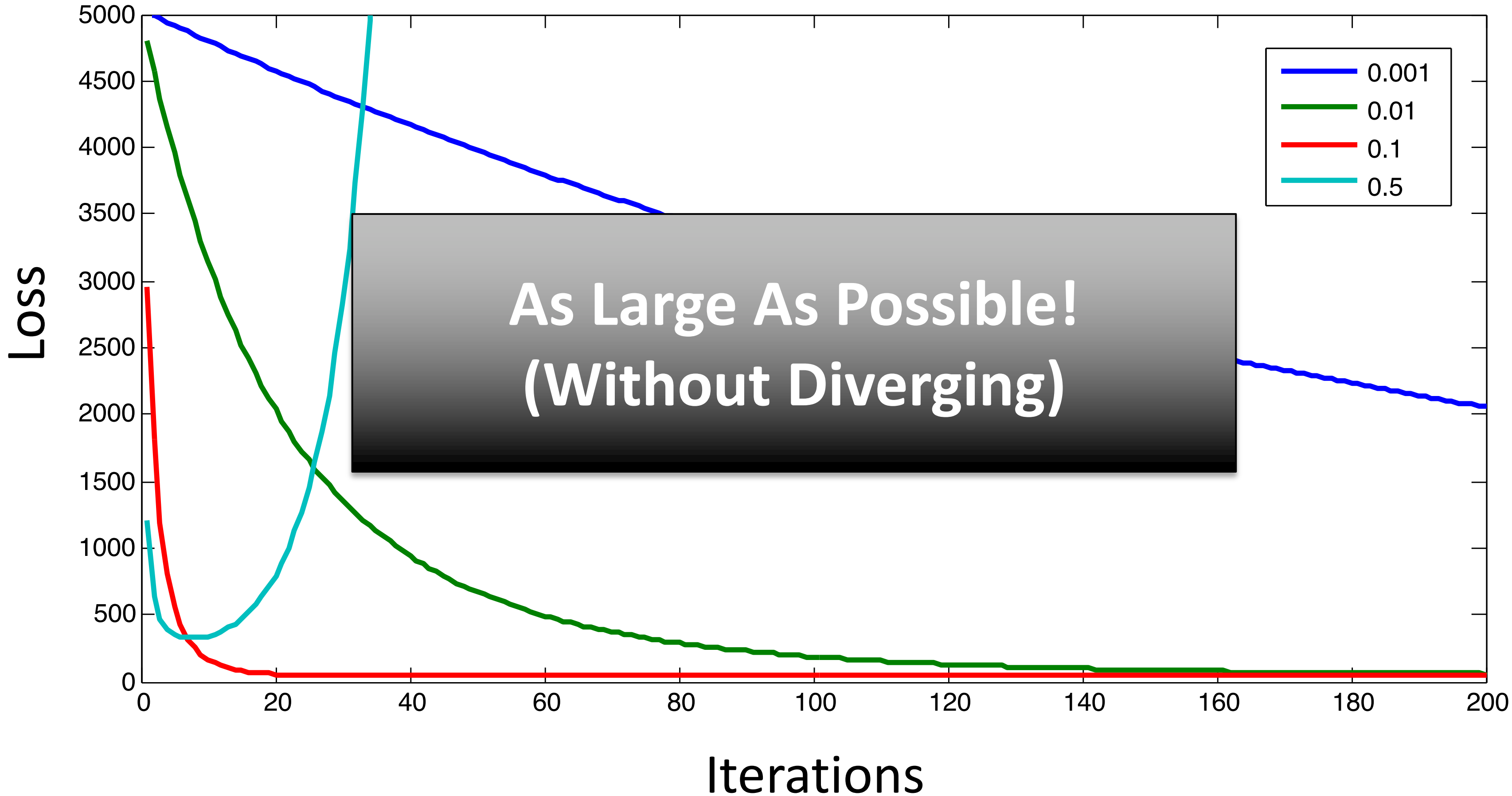
How to Choose Step Size?

$\eta = 0.0001$

$\partial_w L(w) = -2(1-w)$



How to Choose Step Size?



Note that the absolute scale is not meaningful
Focus on the relative magnitude differences

Being Scale Invariant

- Consider the following two gradient updates:

$$w^{t+1} = w^t - \eta^{t+1} \partial_w L(w^t, b^t)$$

$$w^{t+1} = w^t - \hat{\eta}^{t+1} \partial_w \hat{L}(w^t, b^t)$$

- Suppose: $\hat{L} = 1000L$
  - **How are the two step sizes related?**

$$\hat{\eta}^{t+1} = \eta / 1000$$

Practical Rules of Thumb

- Divide Loss Function by Number of Examples:

$$w^{t+1} = w^t - \left(\frac{\eta^{t+1}}{N}\right)\partial_w L(w^t, b^t)$$

- Start with large step size
  - If loss plateaus, divide step size by 2
  - (Can also use advanced optimization methods)
  - (Step size must decrease over time to guarantee convergence to global optimum)

Aside: Convexity

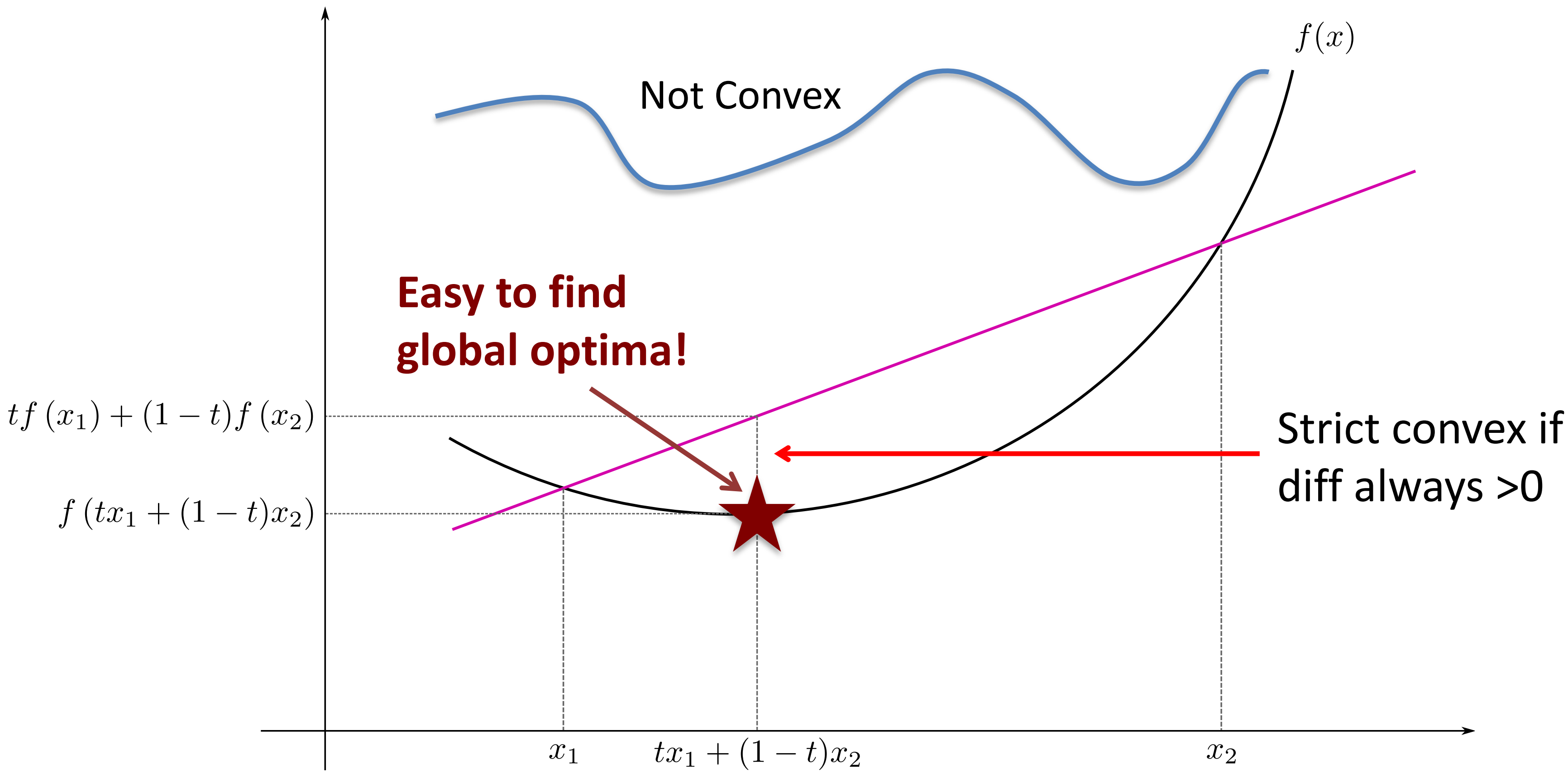
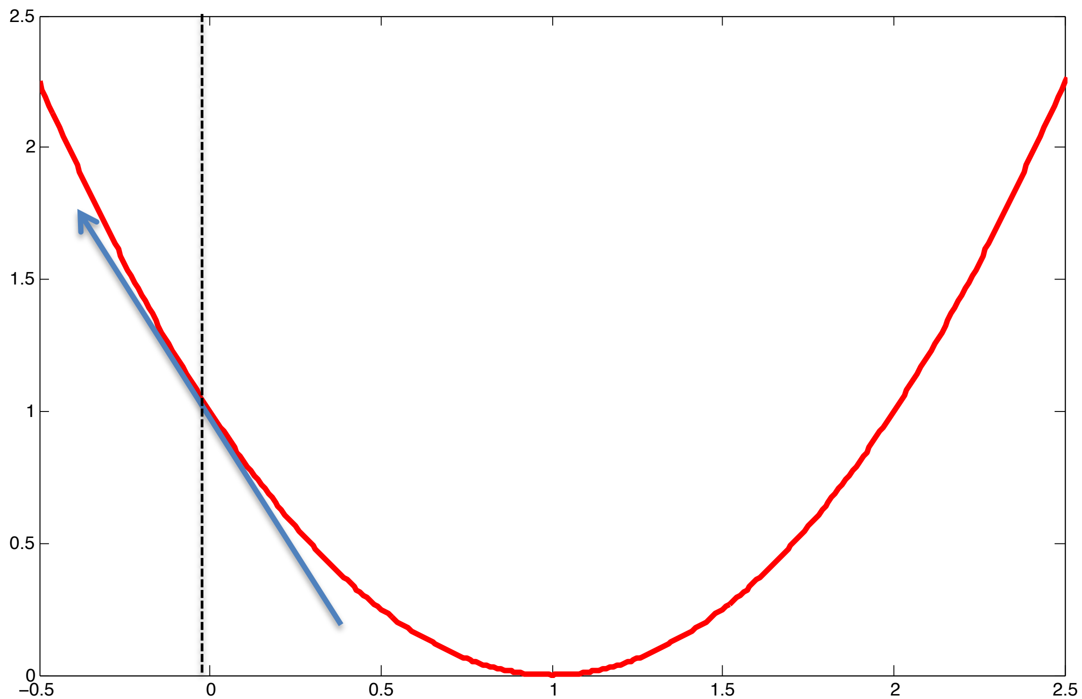


Image Source: http://en.wikipedia.org/wiki/Convex_function

Aside: Convexity

$$L(x_2) \geq L(x_1) + \nabla L(x_1)^T (x_2 - x_1)$$

Function is always above the locally linear extrapolation



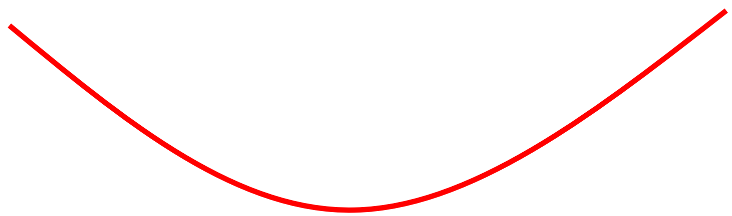
Aside: Convexity

- All local optima are global optima:

Gradient Descent will find optimum

Assuming step size chosen safely

- Strictly convex: unique global optimum:



- Almost all standard objectives are (strictly) convex:
  - Squared Loss, SVMs, LR, Ridge, Lasso
  - We will see non-convex objectives later (e.g., deep learning)

Convergence

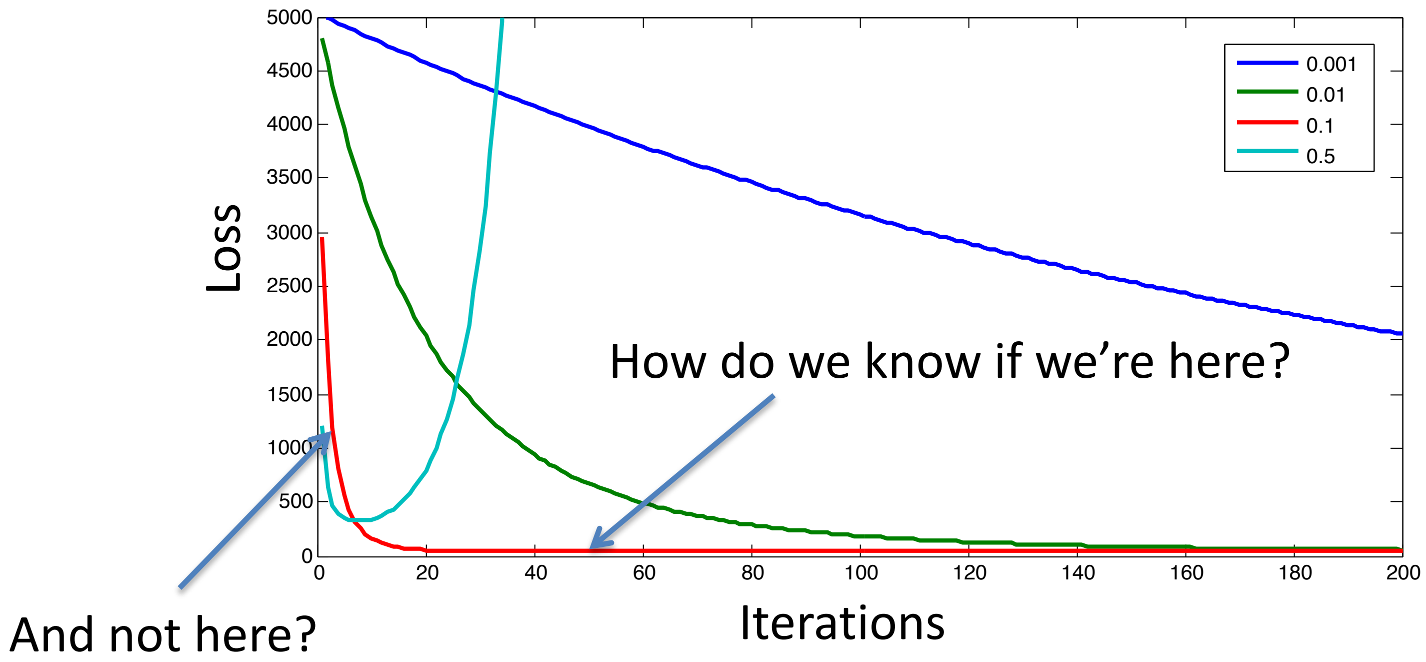
- Assume L is convex
- How many iterations to achieve: $L(w) - L(w^*) \leq \varepsilon$
- If: $|L(a) - L(b)| \leq \rho \|a - b\|$ L is "ρ-Lipschitz"
  - Then $O(1/\varepsilon^2)$ iterations
- If: $|\nabla L(a) - \nabla L(b)| \leq \rho \|a - b\|$ L is "ρ-smooth"
  - Then $O(1/\varepsilon)$ iterations
- If: $L(a) \geq L(b) + \nabla L(b)^T (a - b) + \frac{\rho}{2} \|a - b\|^2$
  - Then $O(\log(1/\varepsilon))$ iterations

L is "ρ-strongly convex"

More Details: Bubeck Textbook Chapter 3

Convergence

- In general, takes infinite time to reach global optimum.
- But in general, we don't care!
  - As long as we're close enough to the global optimum



When to Stop?

- Convergence analyses = worst-case upper bounds
  - **What to do in practice?**
- Stop when progress is sufficiently small
  - E.g., relative reduction less than 0.001
- Stop after pre-specified #iterations
  - E.g., 100000
- Stop when validation error stops going down

Limitation of Gradient Descent

- Requires full pass over training set per iteration

$$\partial_w L(w,b \mid S) = \partial_w \sum_{i=1}^{N} L\left(y_i, f(x_i \mid w,b)\right)$$

- Very expensive if training set is huge

- **Do we need to do a full pass over the data?**

Stochastic Gradient Descent

- Suppose Loss Function Decomposes Additively

$$L(w,b) = \frac{1}{N}\sum_{i=1}^{N} L_i(w,b)$$

Each L_i corresponds to a single data point

- Gradient = expected gradient of sub-functions

$$\partial_w L(w,b) = \partial_w \mathrm{E}_i\left[L_i(w,b)\right] = \mathrm{E}_i\left[\partial_w L_i(w,b)\right]$$

$$L_i(w,b) \equiv \left(y_i - f(x_i \mid w,b\right)^2$$

Stochastic Gradient Descent

- Suffices to take random gradient update
  - So long as it matches the true gradient in expectation
- Each iteration t:
  - Choose i at random

Expected Value is: $\partial_w L(w,b)$

$$w^{t+1} = w^t - \eta^{t+1} \partial_w L_i(w,b)$$

$$b^{t+1} = b^t - \eta^{t+1} \partial_b L_i(w,b)$$

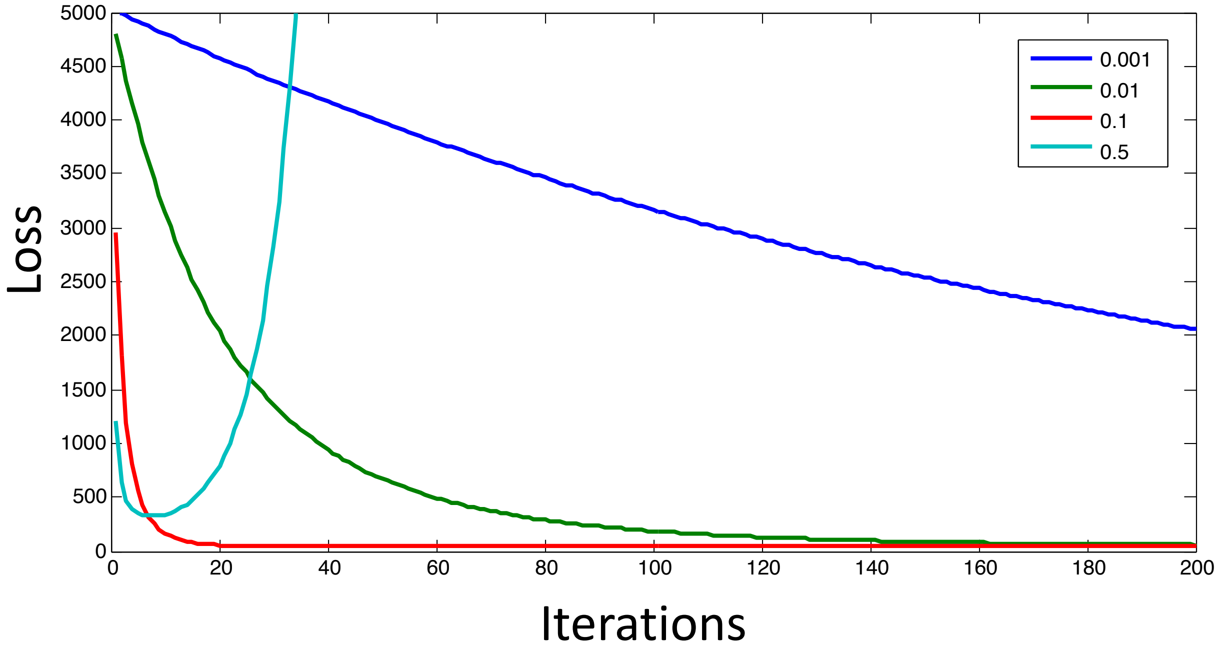
- **SGD is an online learning algorithm!**

Mini-Batch SGD

- Each L_i is a small batch of training examples
  - E.g,. 500-1000 examples
  - Can leverage vector operations
  - Decrease volatility of gradient updates
- Industry state-of-the-art
  - Everyone uses mini-batch SGD
  - Often parallelized
    - (e.g., different cores work on different mini-batches)

Checking for Convergence

- How to check for convergence?
  - Evaluating loss on entire training set seems expensive…



Checking for Convergence

- How to check for convergence?
  - Evaluating loss on entire training set seems expensive…
- Don’t check after every iteration
  - E.g., check every 1000 iterations
- Evaluate loss on a subset of training data
  - E.g., the previous 5000 examples.

Recap: Stochastic Gradient Descent

- Conceptually:
  - Decompose Loss Function Additively
  - Choose a Component Randomly
  - Gradient Update
- Benefits:
  - Avoid iterating entire dataset for every update
  - Gradient update is consistent (in expectation)
- Industry Standard

Perceptron Revisited
(What is the Objective Function?)

- $w^1 = 0, b^1 = 0$
- For t = 1 ….
  - Receive example (x,y)
  - If $f(x|w^t) = y$
    - $[w^{t+1}, b^{t+1}] = [w^t, b^t]$
  - Else
    - $w^{t+1} = w^t + yx$
    - $b^{t+1} = b^t - y$

$$f(x \mid w) = sign(w^T x - b)$$

Training Set:

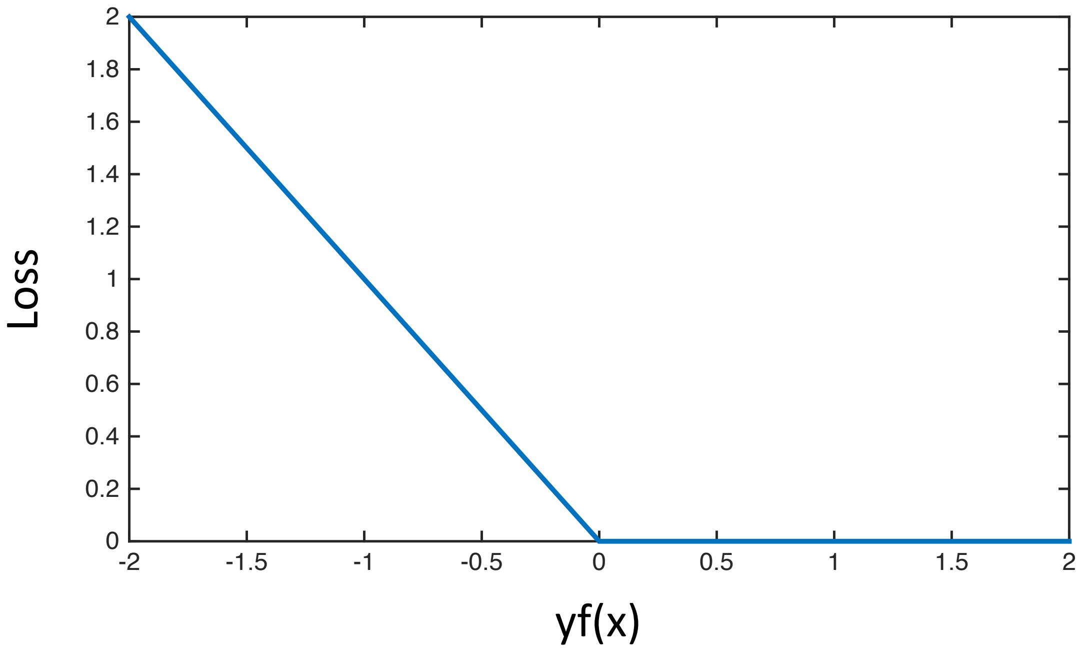
$$S = \{(x_i, y_i)\}_{i=1}^{N}$$

$$y \in \{+1, -1\}$$

Go through training set in arbitrary order (e.g., randomly)

Perceptron (Implicit) Objective

$$L_i(w,b) = \max\left\{0, -y_i f(x_i \mid w,b)\right\}$$



Recap: Complete Pipeline

$$S = \left\{(x_i, y_i)\right\}_{i=1}^{N}$$

Training Data

$$f(x \mid w, b) = w^T x - b$$

Model Class(es)

$$L(a,b) = (a-b)^2$$

Loss Function

$$\underset{w,b}{\operatorname{argmin}} \sum_{i=1}^{N} L\left(y_i, f(x_i \mid w, b)\right)$$

Use SGD!

Cross Validation & Model Selection

Profit!

Next Week

- Different Loss Functions
  - Hinge Loss (SVM)
  - Log Loss (Logistic Regression)
- Primer on Non-linear model classes
  - Neural Nets
- Regularization
- Tonight:
  - Intro to Python