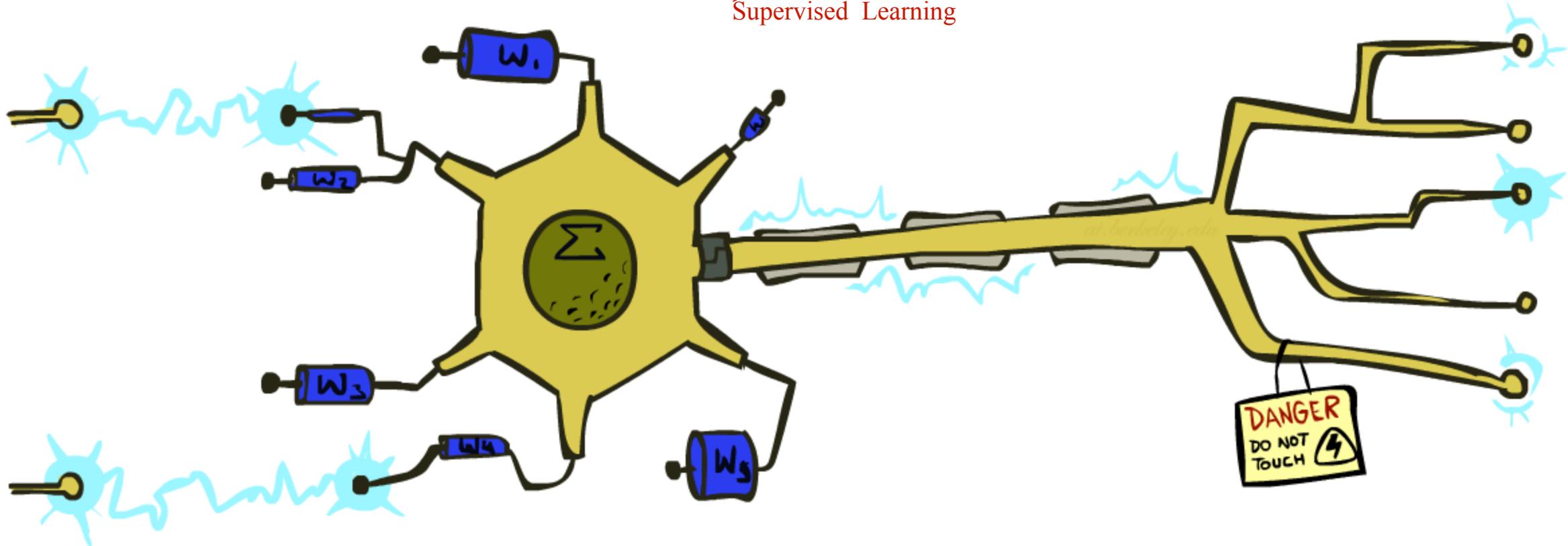


# CS 188: Artificial Intelligence

## Perceptrons

Supervised Learning



[These slides were created by Dan Klein and Pieter Abbeel.]

# Error-Driven Classification

معمولًا روش های Error classification یاد می گیرند

یعنی یک مدلی را در نظر می گیرند و شروع می کنند به بررسی داده های تمرینی  
اگر درست بود که هیچی

اگر اشتباه بود باید مدل را تغییر داد تا بتوان داده ای اشتباه تشخیص داده شده را درست تشخیص دهم .



# Errors, and What to Do

---

## ■ Examples of errors

Dear GlobalSCAPE Customer,

GlobalSCAPE has partnered with ScanSoft to offer you the latest version of OmniPage Pro, for just \$99.99\* - the regular list price is \$499! The most common question we've received about this offer is - Is this genuine? We would like to assure you that this offer is authorized by ScanSoft, is genuine and valid. You can get the . . .

. . . To receive your \$30 Amazon.com promotional certificate, click through to

<http://www.amazon.com/apparel>

and see the prominent link for the \$30 offer. All details are there. We hope you enjoyed receiving this message. However, if you'd rather not receive future e-mails announcing new store launches, please click . . .

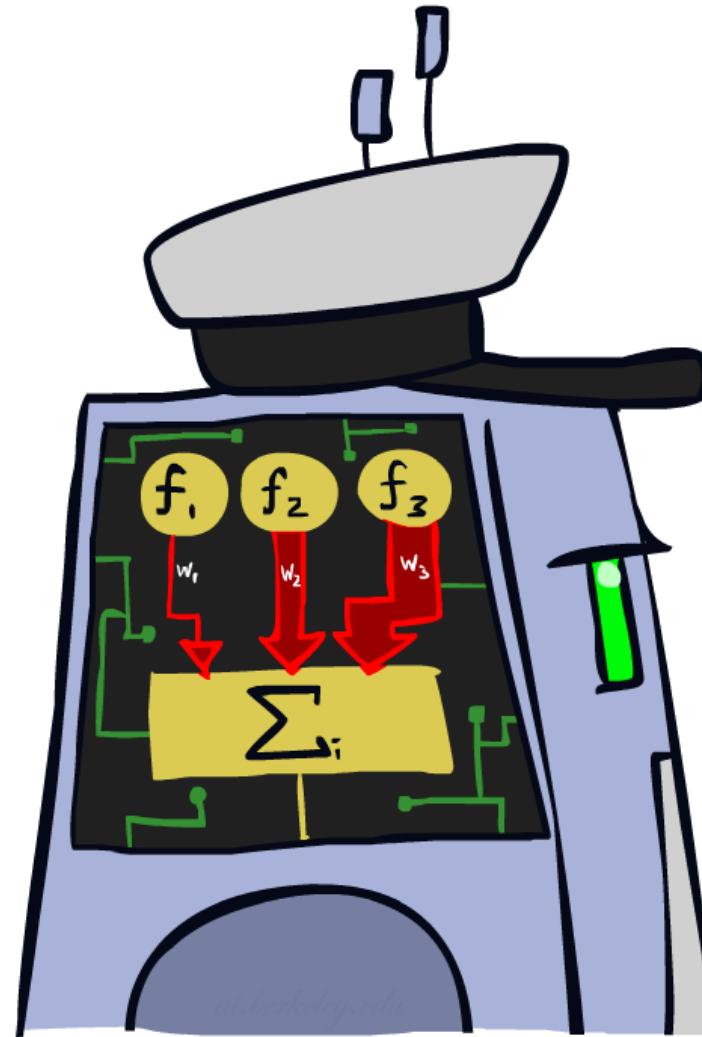
# What to Do About Errors

---

- Problem: there's still spam in your inbox
- Need more **features** – words aren't enough!
  - Have you emailed the sender before?
  - Have 1M other people just gotten the same email?
  - Is the sending information consistent?
  - Is the email in ALL CAPS?
  - Do inline URLs point where they say they point?
  - Does the email address you by (your) name?

# Linear Classifiers

Perceptron



# Feature Vectors

$x$

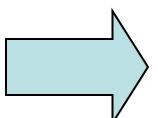
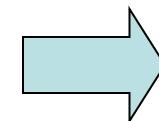
```
Hello,  
  
Do you want free printr  
cartridges? Why pay more  
when you can get them  
ABSOLUTELY FREE! Just
```

$f(x)$

$$\begin{Bmatrix} \# \text{ free} & : 2 \\ \text{YOUR\_NAME} & : 0 \\ \text{MISSPELLED} & : 2 \\ \text{FROM\_FRIEND} & : 0 \\ \dots \end{Bmatrix}$$

$y$

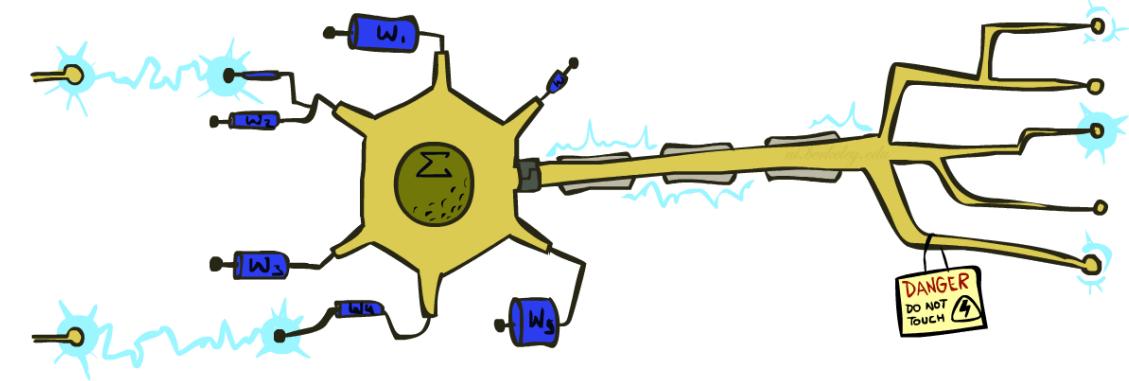
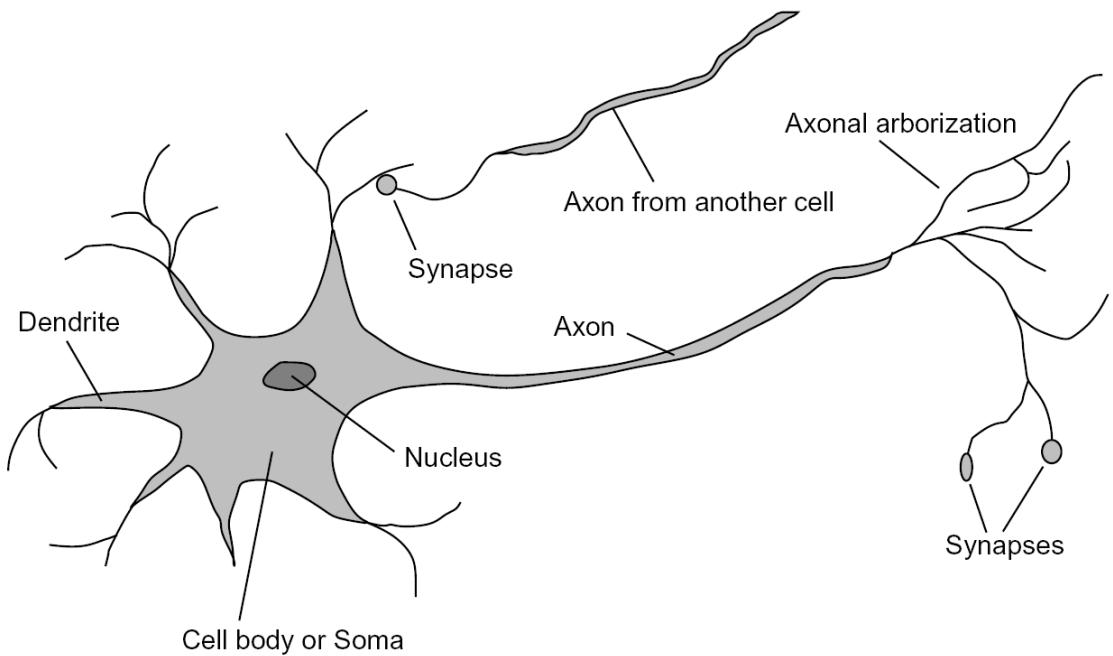
SPAM  
or  
+


$$\begin{Bmatrix} \text{PIXEL-7,12} & : 1 \\ \text{PIXEL-7,13} & : 0 \\ \dots \\ \text{NUM\_LOOPS} & : 1 \\ \dots \end{Bmatrix}$$


“2”

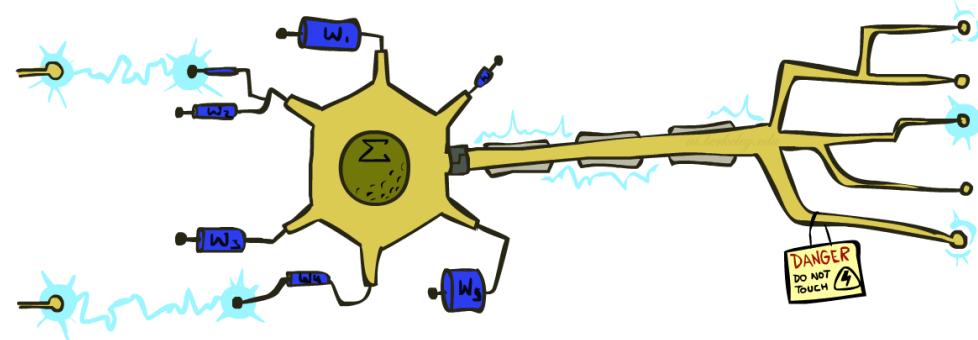
# Some (Simplified) Biology

- Very loose inspiration: human neurons



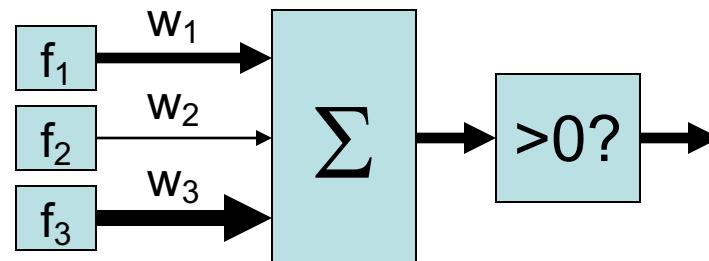
# Linear Classifiers

- Inputs are **feature values**
- Each feature has a **weight**
- Sum is the **activation**



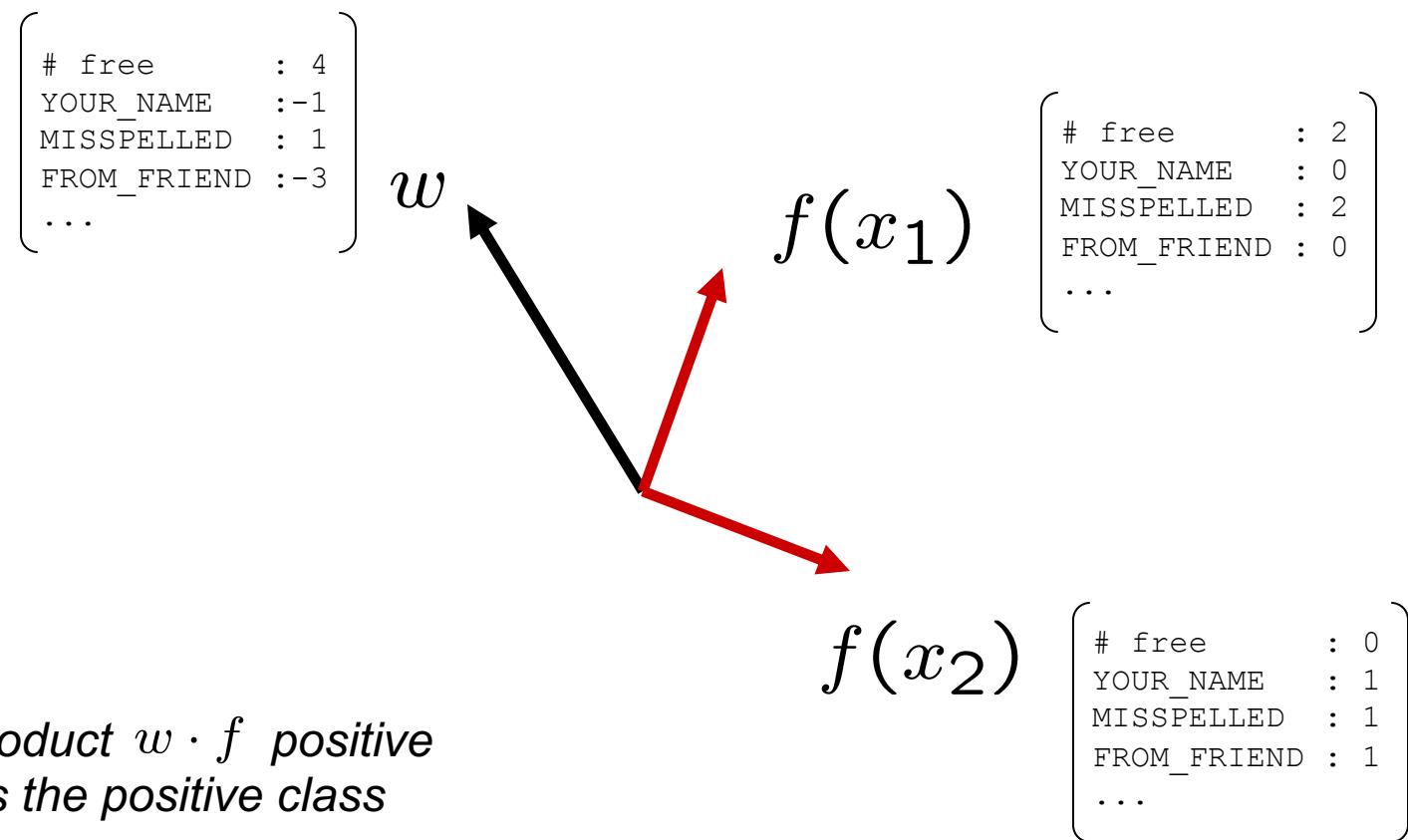
$$\text{activation}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$$

- If the activation is:
  - Positive, output +1
  - Negative, output -1

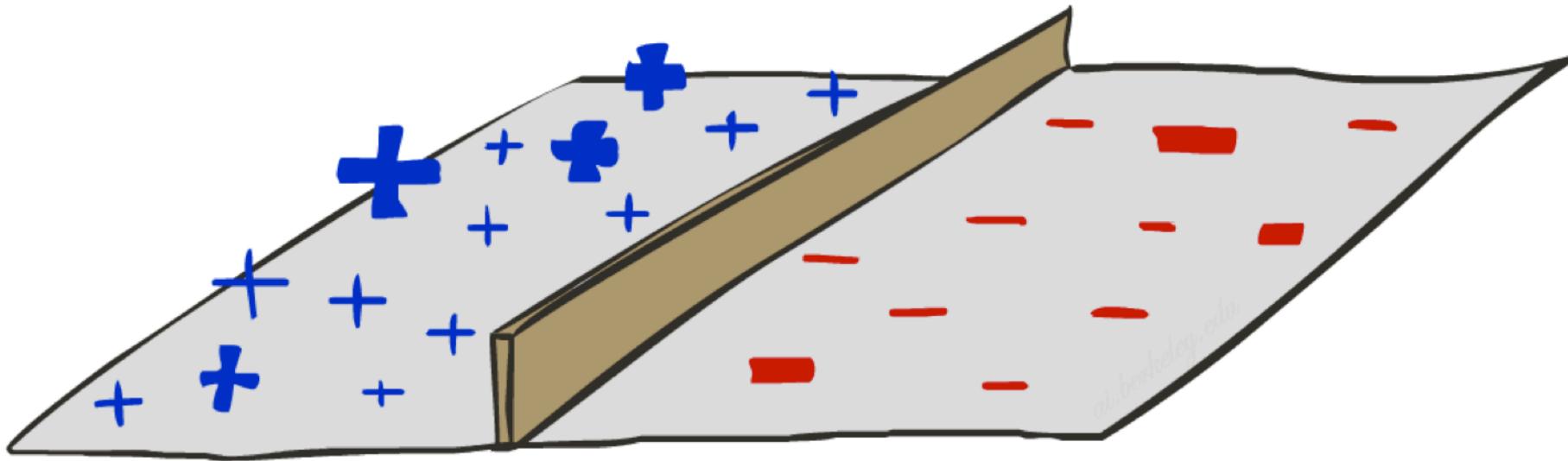


# Weights

- Binary case: compare features to a weight vector
- Learning: figure out the weight vector from examples



# Decision Rules

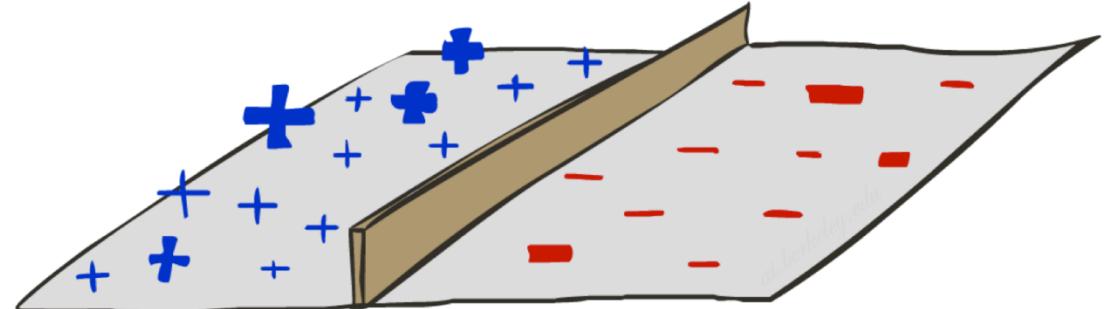


# Binary Decision Rule

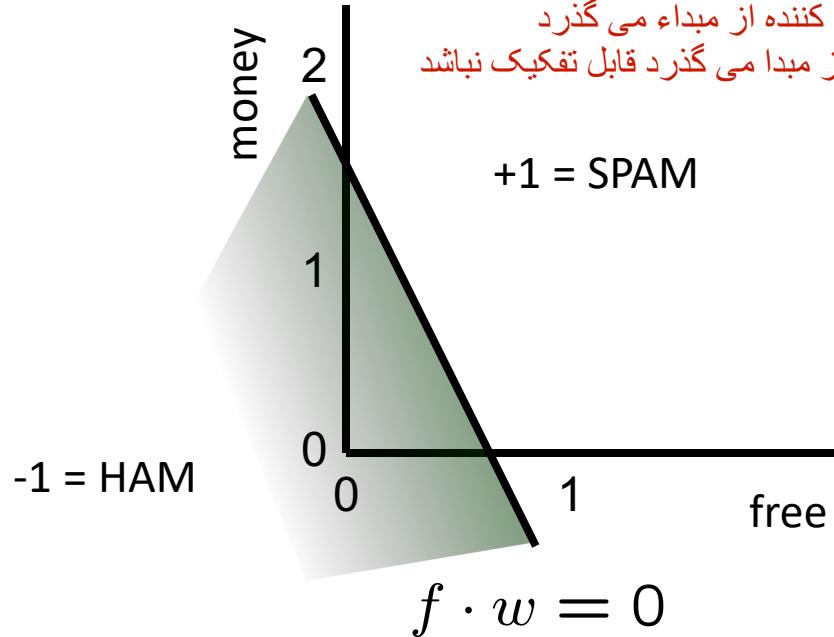
- In the space of feature vectors
  - Examples are points
  - Any weight vector is a hyperplane
  - One side corresponds to  $Y=+1$
  - Other corresponds to  $Y=-1$

$w$

BIAS	:	-3
free	:	4
money	:	2
...		

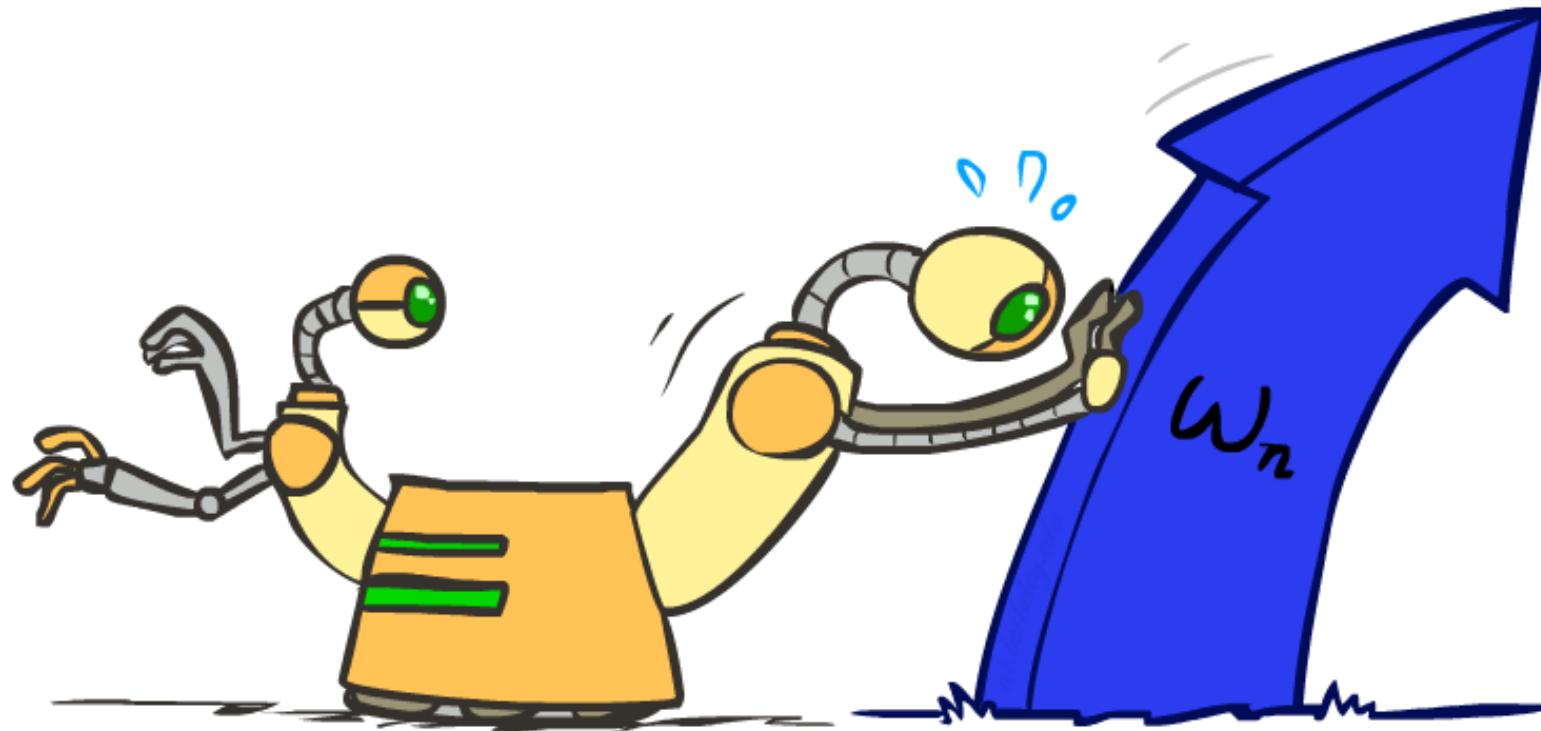


اگر bias را نگذاریم همیشه خط جدا کننده از مبداء می گزند  
ولی ممکن است داده های ما توسط خطی که از مبداء می گزد قابل تفکیک نباشد



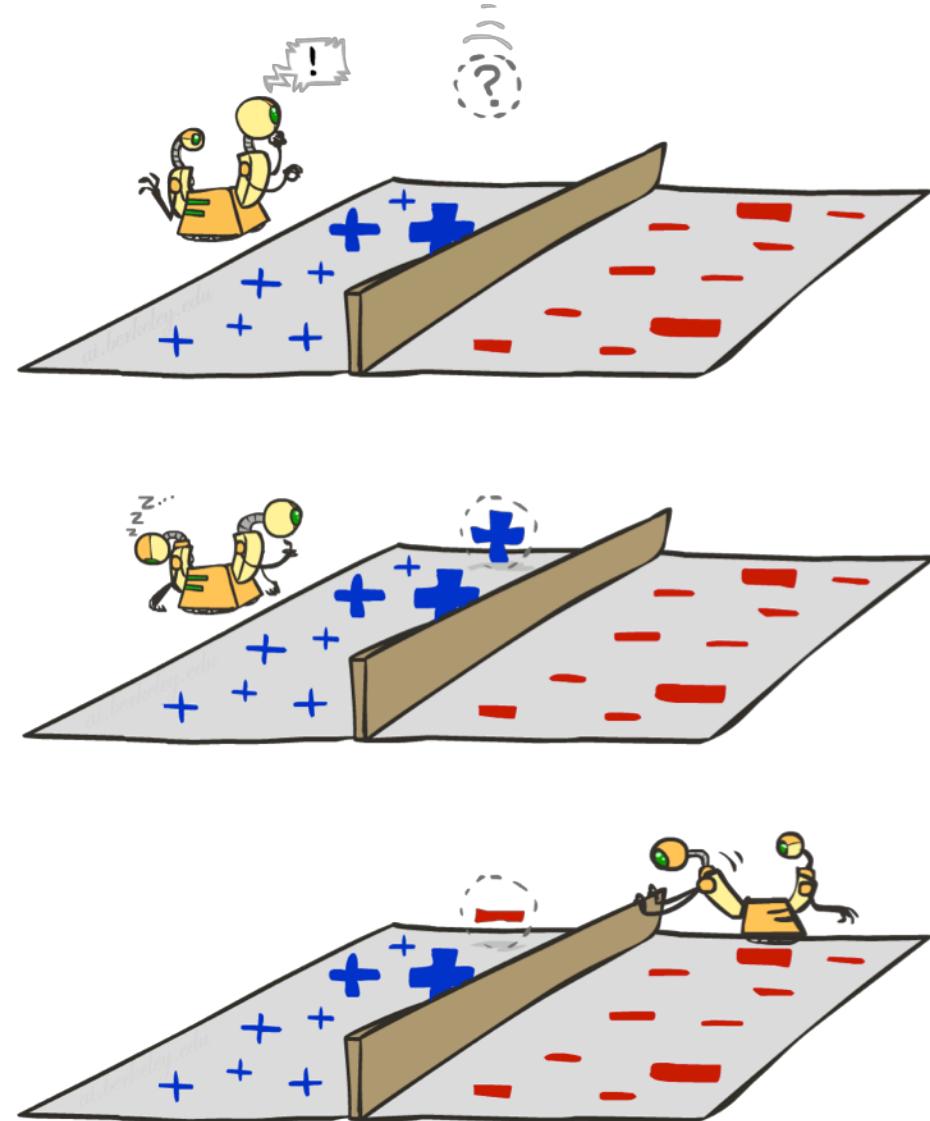
# Weight Updates

---



# Learning: Binary Perceptron

- Start with weights = 0
- For each training instance:
  - Classify with current weights
  - If correct (i.e.,  $y=y^*$ ), no change!
  - If wrong: adjust the weight vector



# Learning: Binary Perceptron

- Start with weights = 0
- For each training instance:
  - Classify with current weights

$$y = \begin{cases} +1 & \text{if } w \cdot f(x) \geq 0 \\ -1 & \text{if } w \cdot f(x) < 0 \end{cases}$$

برای کم کردن زاویه با آن بردار جمع می زنیم برای اصلاح اشتباه (مثبت بوده ولی منفی تشخیص دادیم)

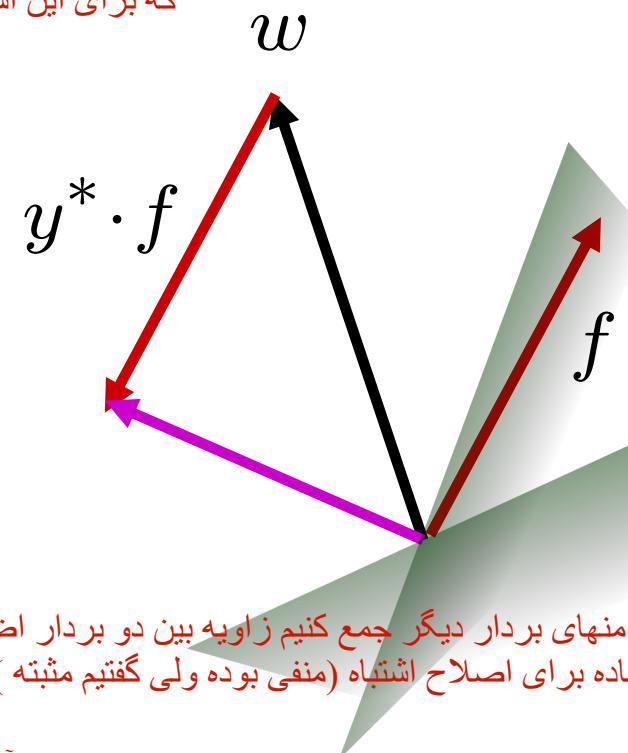
- If correct (i.e.,  $y=y^*$ ), no change!
- If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if  $y^*$  is -1.

$$w = w + y^* \cdot f$$

در هر روش classification

باید در هر وزن برای هر داده یک مقدار ثابت به نام bias در نظر بگیریم .  
که برای این است که تمام خط های جدا کننده از مبدا نگزند .

فرض می کنیم وزن اولیه برای همه برابر با صفر باشد



اگر یک بردار را با منهای بردار دیگر جمع کنیم زاویه بین دو بردار اضافه می شود  
یک راه ساده برای اصلاح اشتباه (منفی بوده ولی گفتیم مثبته )

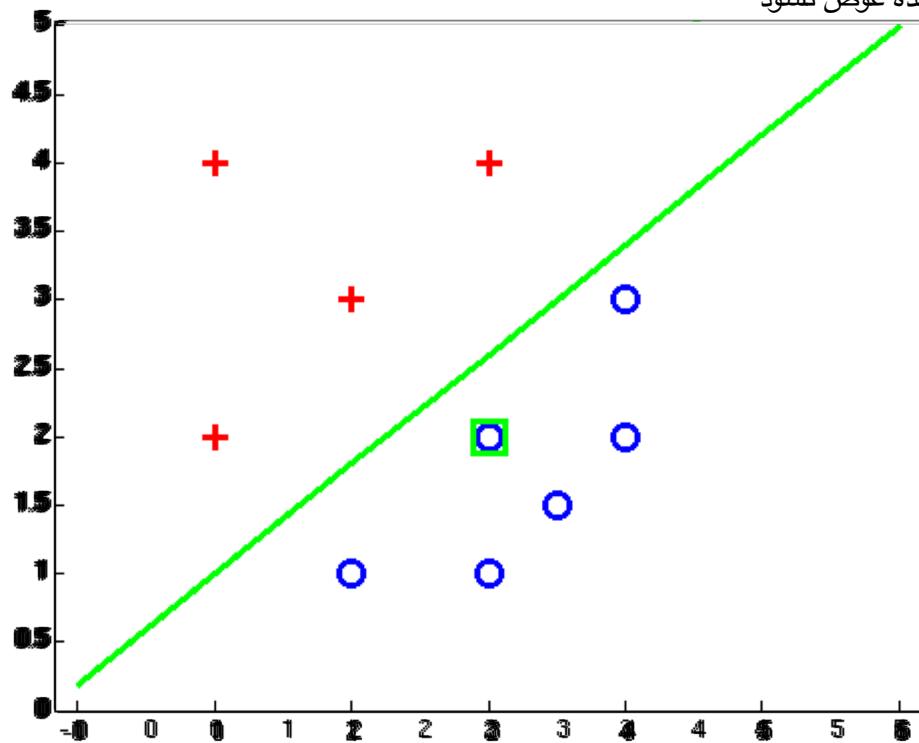
یک مرحله از classification باید پیدا کردن وزنها و محاسبه آن ها  
باشد که باید از روی داده های تمرینی باید تعیین شود .

به جایی می رسمیم که هز چه قدر زیاد و کم کنیم دیگر بردار وزن ها تغییری نمی کند  
در آن جا دیگر همگرا شده است و کار classifier تمام است .

# Examples: Perceptron

## ■ Separable Case

در این اسلام چند تا عکس روی هم افتد در اصل ۹ مرحله است 😊



در این نوع دیتا ها می توان واقعا داده ها را با یک خط از یک دیگر جدا کرد  
در این صورت اگر بردار وزن وسط بیافتد و دیگر مکانش تغییری نمی کند  
و همگرا می شود

در هر بار که داده وارد می کنیم این عملیات تکرار می شود تا به درستی بتوانیم  
دسته بندی کنیم تا جایی که دیگر جای خط جدا کننده عوض نشود

اگر ضرب داخلی صفر شود مثبت در نظر می گیریم

اگر یک داده خیلی منفی (اشتباه تشخیص داده شده) داشته باشیم  
می تواند نویز ایجاد کند  
بنابراین این روش خیلی نسبت به نویز مقاوم نیست

# Multiclass Decision Rule

- If we have multiple classes:
  - A weight vector for each class:

$$w_y$$

به جای این که یک بردار وزن داشته باشیم به ازای هر کلاس یک بردار وزن در نظر بگیریم

- Score (activation) of a class  $y$ :

هر داده که وارد می شود ضرب نقطه ای آن را با بردار وزن هر کلاس محاسبه می کنیم

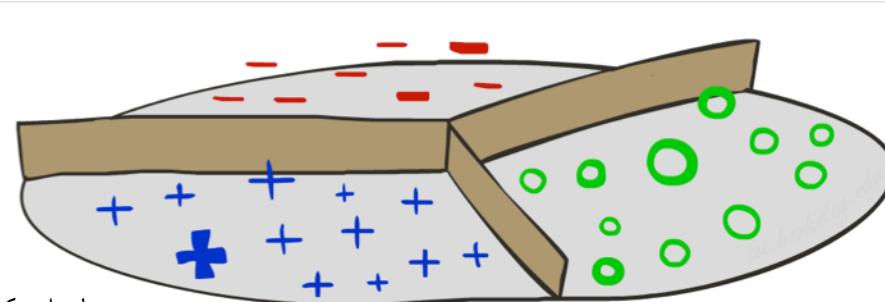
$$w_y \cdot f(x)$$

اگر اندازه  $w_1$  نسبت به دو تای دیگر خیلی بیشتر شود  
مرز ها رو بزرگ تر میکنه به نفع خودش

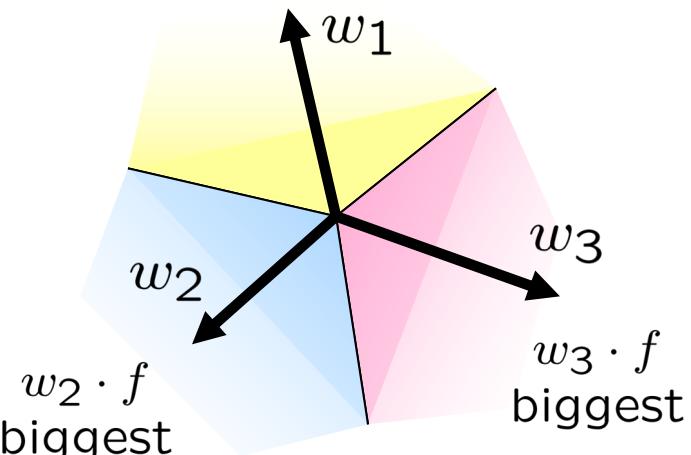
- Prediction highest score wins

$$y = \arg \max_y w_y \cdot f(x)$$

برداری که مаксیمم مقدار را داشته باشد به عنوان کلاس خروجی در نظر گرفته می شود (یعنی داده ورودی جزو این کلاس میباشد)



$w_1 \cdot f$  biggest



*Binary = multiclass where the negative class has weight zero*

# Learning: Multiclass Perceptron

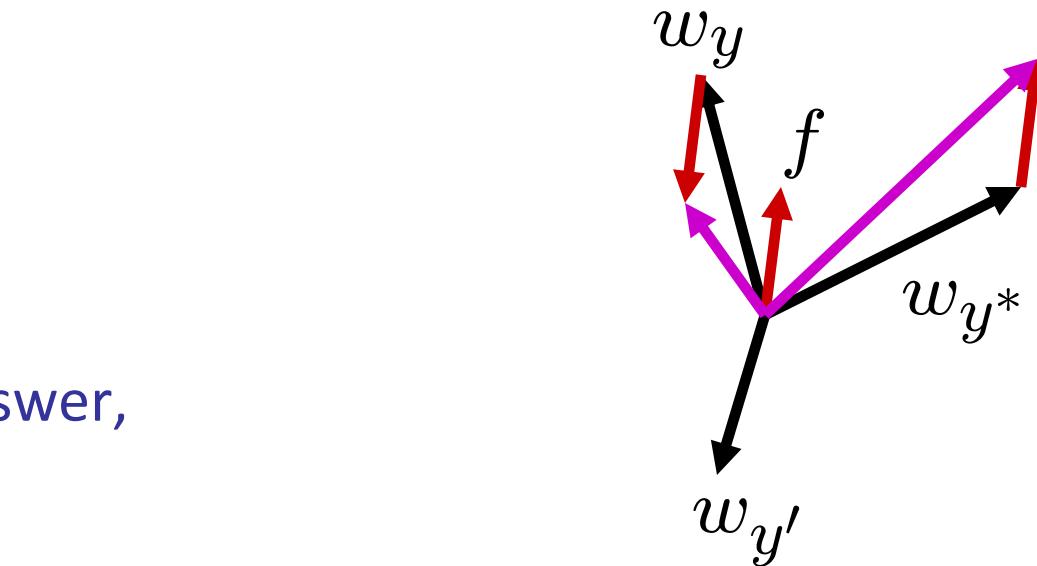
- Start with all weights = 0
- Pick up training examples one by one
- Predict with current weights

$$y = \arg \max_y w_y \cdot f(x)$$

- If correct, no change!
- If wrong: lower score of wrong answer,  
raise score of right answer

$$w_y = w_y - f(x)$$

$$w_{y^*} = w_{y^*} + f(x)$$



اونی که اشتباه کرده با منفی این بردار جمع کنیم  
اونی که باید درست می بوده با مثبت اون بردار جمع می شود  
در این صورت اونی که باید درست می بوده فاصله آن از بردار وزن ها  
نزدیک تر می شود و داده که اشتباه تشخیص داده شده فاصله اش از بردار وزن ها  
بیشتر می شود

# Example: Multiclass Perceptron

“win the vote”

“win the election”

“win the game”

در مرحله اول متن سیاسی به اشتباه ورزشی تشخیص داده می شود .  
برای اصلاح اشتباه خود بردار وزن ورزش Wsport را از بردار وزن خط اول کم می کنیم

با این برای تمام داده ها یک می باشد (نه لزوما یک ولی باید یک  $k$  ثابتی باشد )  
ثابت می شه که چه ۲ کلاس داشته باشیم یا  $k$  تا کلاس داشته باشیم این روش perceptron درصورتی که  
داده ها به صورت خطی جدا پذیر باشند (separable) باشند به آن خط همگرا می شود .

$w_{SPORTS}$

BIAS	:	1
win	:	0
game	:	0
vote	:	0
the	:	0
...		

$w_{POLITICS}$

BIAS	:	0
win	:	0
game	:	0
vote	:	0
the	:	0
...		

$w_{TECH}$

BIAS	:	0
win	:	0
game	:	0
vote	:	0
the	:	0
...		

# Properties of Perceptrons

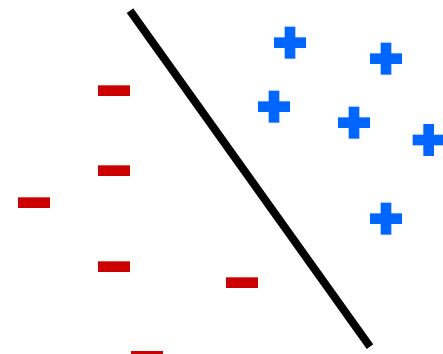
- Separability: true if some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

$$\text{mistakes} < \frac{k}{\delta^2}$$

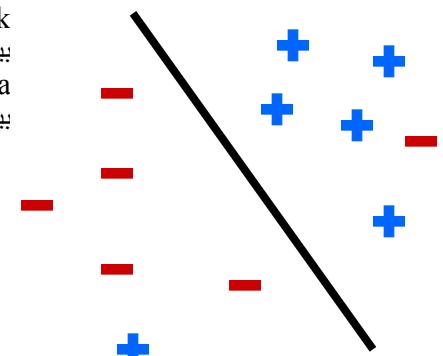
چرا اگر به داده های منفی بچسید ، بد است ؟  
چیز های خیلی نزدیک به منفی را مثبت تشخیص داده خواهد شد  
آن خطی که margin را مаксیمم می کند بهتر است (تعداد خطاهایی که انجام می دهیم تا به خط درست بررسیم )

برای یک خط ثابت margin برابر مینیمم فاصله خط از داده هاست  
اگر خط های مختلف را در نظر بگیریم باید بین margin های مختلف مаксیمم بگیریم

Separable



Non-Separable



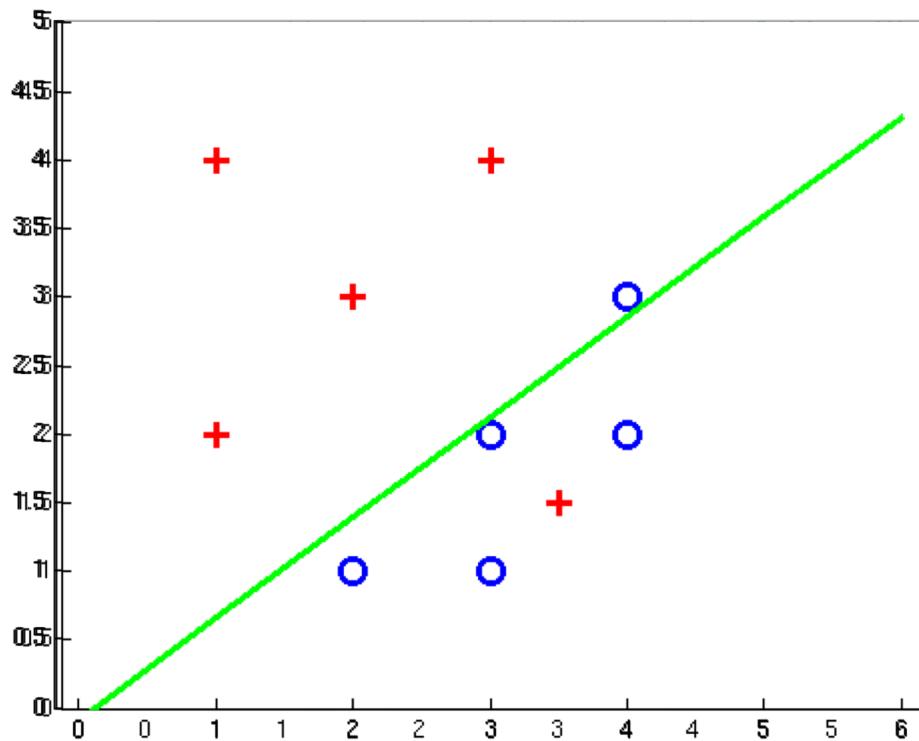
# Examples: Perceptron

## ■ Non-Separable Case

در صورتی که داده ها خطی تفکیک پذیر نباشد دو تا اشکال به وجود میاد :

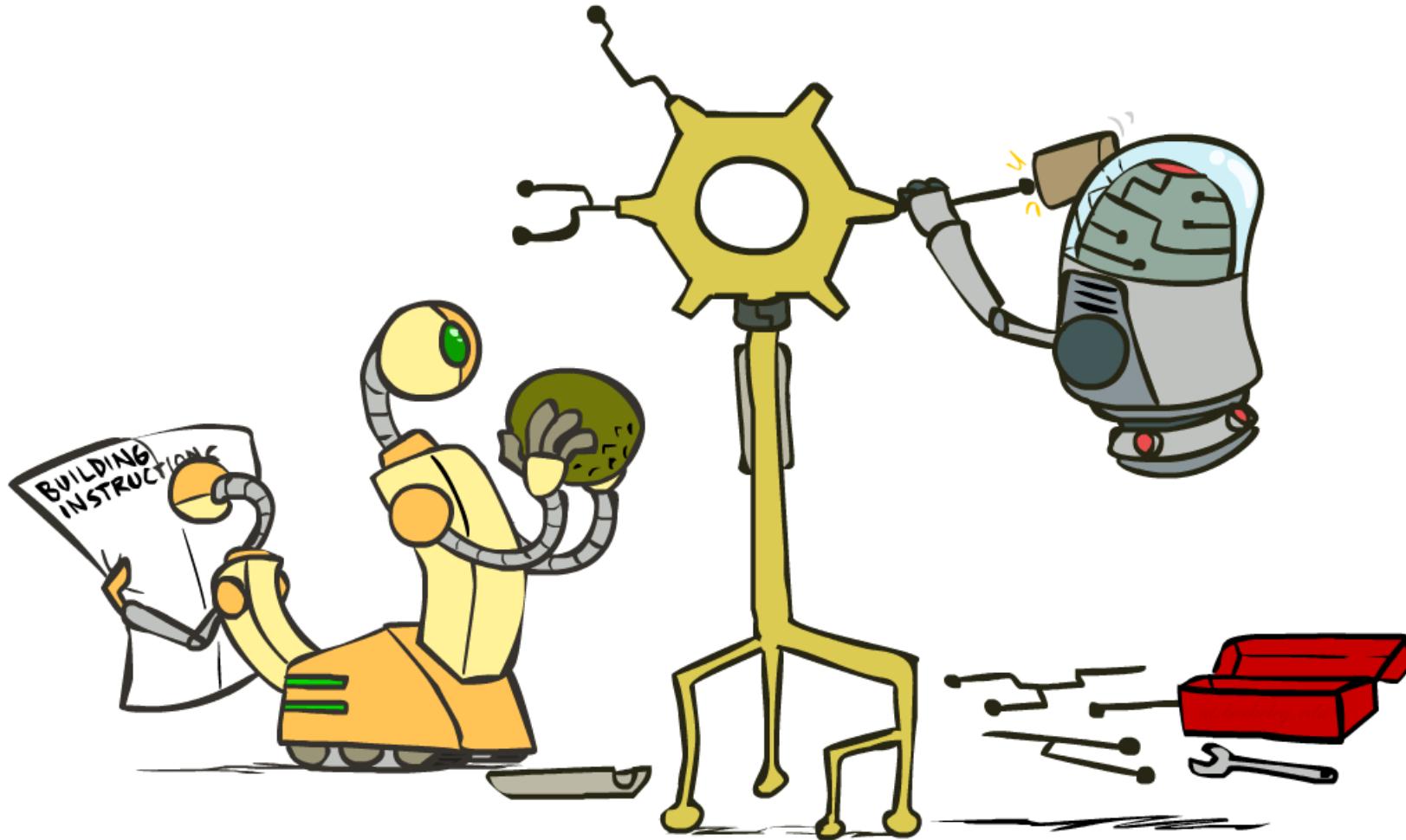
1- خیلی طول می کشه تا کار الگوریتم train به اتمام برسه

2- نتیجه هم ممکنه خیلی بد باشه



# Improving the Perceptron

---

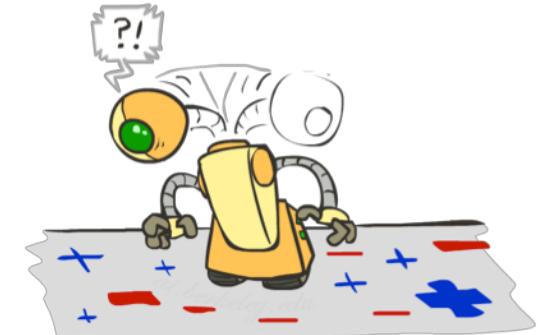
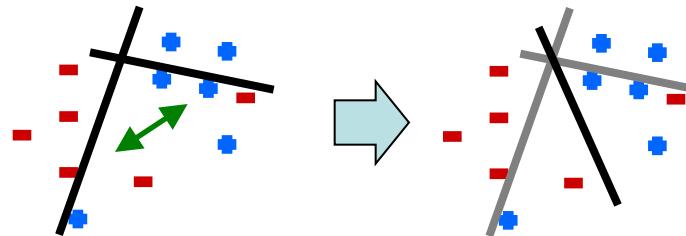


# Problems with the Perceptron

مشکل زیر با MIRA حل می شود

- Noise: if the data isn't separable, weights might thrash

- Averaging weight vectors over time can help (averaged perceptron)



ما مدل را تا اینجا train کردم بعد از این نگاه می کنیم  
دقت روی داده هایی که بیرون نگه داشتیم چه قدر است

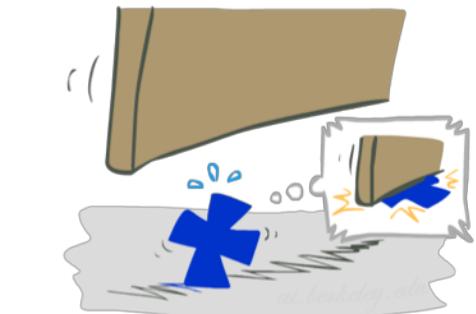
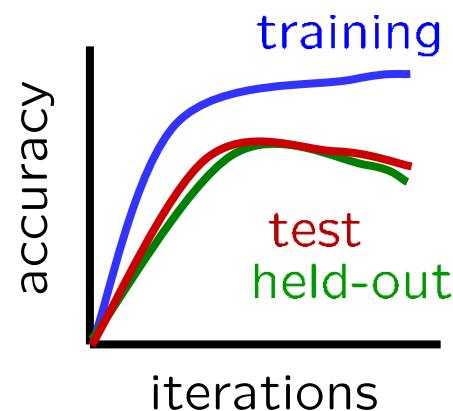
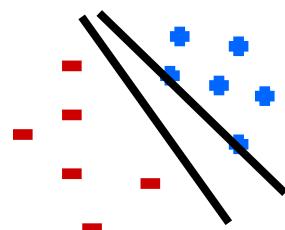
و زمانی که می بینیم دقیق دارد بر روی این داده ها کم می شود باید training را متوقف کنیم



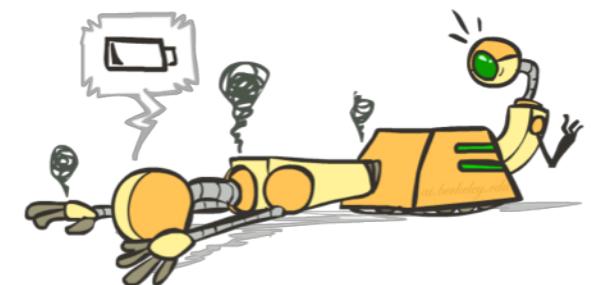
- Mediocre generalization: finds a “barely” separating solution

برای جلوگیری از overfit شدن داده های train را به دو قسمت تقسیم می کنند:  
1- یک قسمت داده هایی هستند که روشون train انجام می دهیم **مشکل بالا با MIRA حل می شود**

2- یک دسته دیگر را نگه می دارند و train را بر روی آنها انجام نمی دهد و برای تنظیم پارامتر های training مورد استفاده قرار می کنند (مثل این که چند بار training انجام بدیم یا تا کجا train را انجام بدیم و ...)



- Overtraining: test / held-out accuracy usually rises, then falls
- Overtraining is a kind of overfitting



# Fixing the Perceptron

- Idea: adjust the weight update to mitigate these effects
- MIRA\*: choose an update size that fixes the current mistake...
- ... but, minimizes the change to w

$$\min_w \frac{1}{2} \sum_y ||w_y - w'_y||^2$$

Condition :

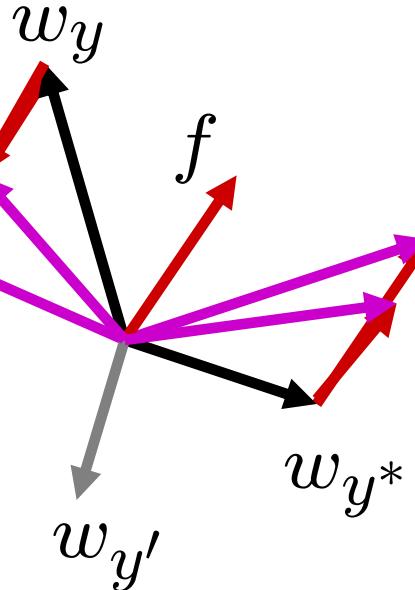
$$w_{y^*} \cdot f(x) \geq w_y \cdot f(x) + 1$$

- The +1 helps to generalize

\* Margin Infused Relaxed Algorithm

در این روش هر جا که می خواهیم وزن هارا تغییر دهیم  
به جای این که از روش قبلی استفاده کنیم  
در این روش یک ضریب برای اضافه و کم کردن در نظر می گیریم  
و به ما کنترل می دهد که چه قدر بردار وزن را تغییر دهیم  
و سعی می کنیم تغییرات کمی داشته باشیم ولی از یک حدی باید بیشتر باشد  
باید یک کمیت را  $\min$  کنیم

تغییر می دهیم تا مقدار کلاسی که باید به دست می آمد  
با کلاسی که به صورت اشتباه به دست آمده از یک عدد ثابتی بیشتر باشد  
بنابراین عبارت زیر سیگما را برابر  $\min$  کنیم باشرط گفته شده در پایین



Guessed  $y$  instead of  $y^*$  on example  $x$  with features  $f(x)$

می خواهیم کلاس درستمان به یک اندازه مشخص اختلاف  
 $w^* f(x), Wy^* f(x), Wy f(x)$   
می خواهیم اختلاف این دو تا مقدار از یک کمتر باشد  
ولی می خواهیم یک مقدار مشخصی اختلاف داشته باشیم

$$w_y = w'_y - \tau f(x)$$

$$w_{y^*} = w'_{y^*} + \tau f(x)$$

# Minimum Correcting Update

$$\min_w \frac{1}{2} \sum_y \|w_y - w'_y\|^2$$

$$w_{y^*} \cdot f \geq w_y \cdot f + 1$$



$$\min_{\tau} \|\tau f\|^2$$

$$w_{y^*} \cdot f \geq w_y \cdot f + 1$$



$$(w'_{y^*} + \tau f) \cdot f = (w'_y - \tau f) \cdot f + 1$$

$$\tau = \frac{(w'_y - w'_{y^*}) \cdot f + 1}{2f \cdot f}$$

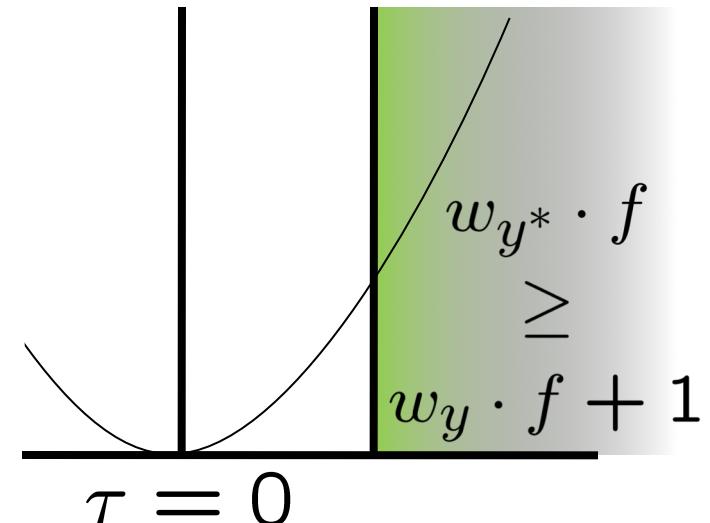
$$w_y = w'_y - \tau f(x)$$

$$w_{y^*} = w'_{y^*} + \tau f(x)$$

بنابراین یک تاو پیدا کردیم که بردار وزن را تغییر میدهد.  
ولی از یک حدی بیشتر این تغییرات را انجام نمی دهد.  
حداقل به اندازه یک واحد از کلاس اشتباه بیشتر شود مقدارش

مقدار  $f$  همان داده فعلی است و تغییری نمی کند

در نقطه صفر شرط نامساوی برقرار نیست  
ولی به جایی می رسد که این شرط برقرار است



min not  $\tau=0$ , or would not have made an error, so min will be where equality holds

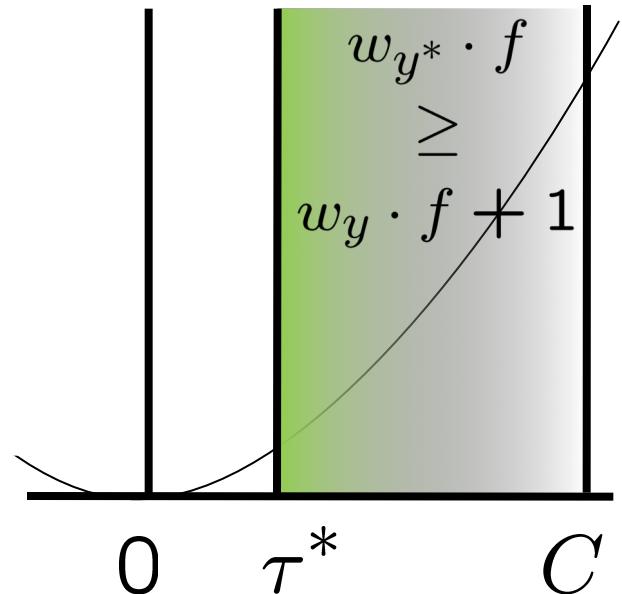
# Maximum Step Size

- In practice, it's also bad to make updates that are too large

- Example may be labeled incorrectly
- You may not have enough features
- Solution: cap the maximum possible value of  $\tau$  with some constant  $C$

$$\tau^* = \min \left( \frac{(w'_y - w'_{y^*}) \cdot f + 1}{2f \cdot f}, C \right)$$

مقدار ثابت  $C$  را در عمل برای بالا بردن دقت در نظر می‌گیرند  
مقدار update را با  $C$  مینیمم می‌گیرند  
این ثابت اجازه نمی‌دهد بیشتر از حدی بردار تغییر کند  
در بسیاری از موارد دیده شده که این ثابت از تأثیر بیش از حد outlayer ها را کم می‌کند و نمی‌گذارد بردار وزن بیش از حد تغییر پیدا کند



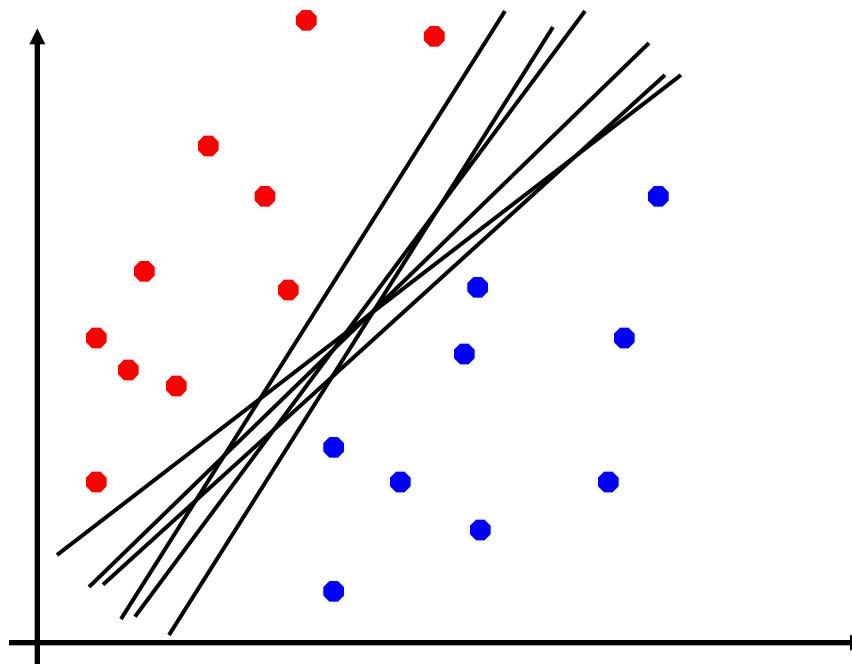
- Corresponds to an optimization that assumes non-separable data
- Usually converges faster than perceptron
- Usually better, especially on noisy data

# Linear Separators

- Which of these linear separators is optimal?

کیفیت این خط ها با یک دیگر متفاوت است  
به خاطر این که ما الان این داده رو می بینیم و  
داده اصلی (test) را مانداریم

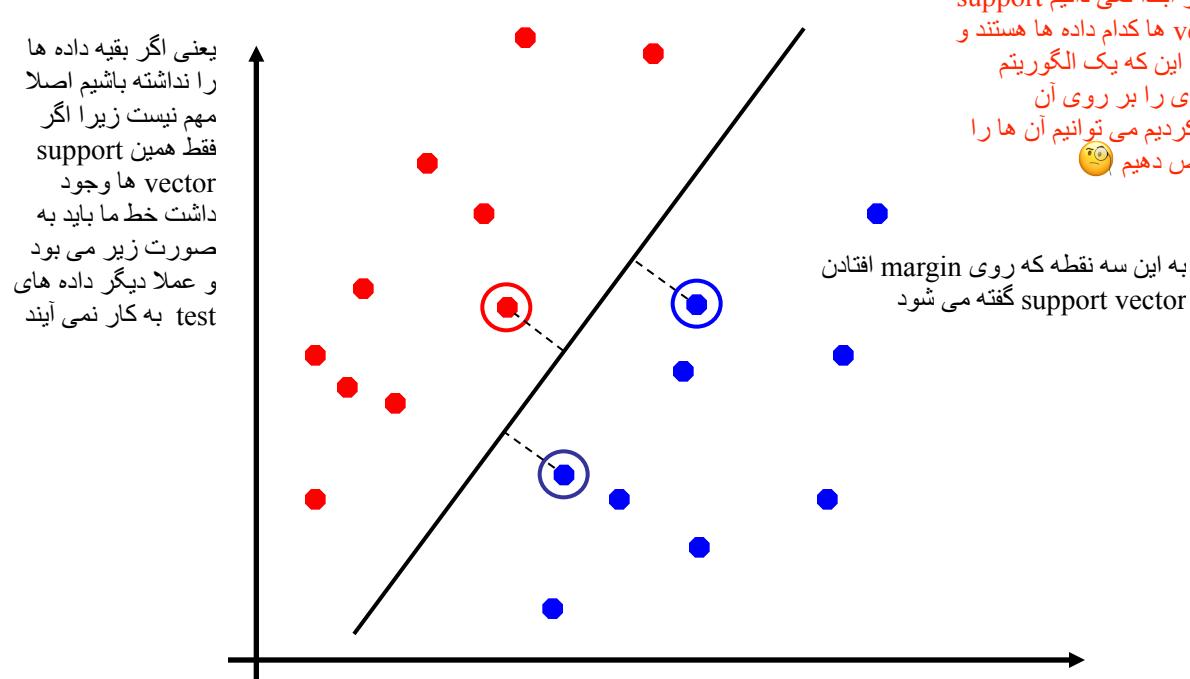
اگر داده خطی تلقیک پذیر باشد یک خط unique وجود  
دارد که مینیمم فاصله هایی که با این نقاط دارد را بین همه خط ها مаксیمم می کند



# Support Vector Machines

- Maximizing the margin: good according to intuition, theory, practice
- Only support vectors matter; other training examples are ignorable
- Support vector machines (SVMs) find the separator with max margin
- Basically, SVMs are MIRA where you optimize over all examples at once

وقتی خط را رسم کنیم از هر کلاس حداقل یک داده بر روی margin می‌افتد  
یک خط موازی با خط اصلی و نزدیک به داده‌های مثبت رسم می‌کنیم تا حداقل یک داده بر روی این خط بیافتد  
همین کار را برای داده‌های منفی هم انجام می‌دهیم



MIRA

$$\min_w \frac{1}{2} \|w - w'\|^2$$
$$w_y^* \cdot f(x_i) \geq w_y \cdot f(x_i) + 1$$

SVM

$$\min_w \frac{1}{2} \|w\|^2$$
$$\forall i, y \quad w_y^* \cdot f(x_i) \geq w_y \cdot f(x_i) + 1$$