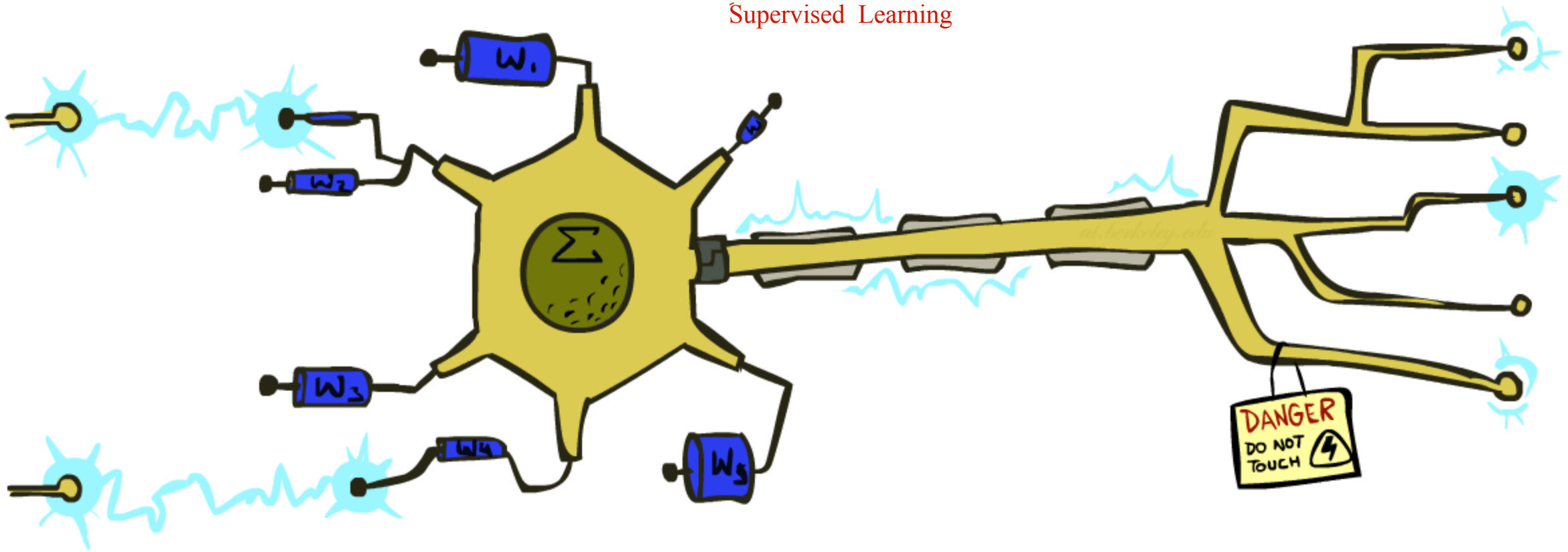
CS 188: Artificial Intelligence

Perceptrons

Supervised Learning



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

Error-Driven Classification

معمولا روش های cclassificationبا Error یاد می گیرند
یعنی یک مدلی را در نظر می گیرند و شروع می کنند به بررسی داده های تمرینی
اگر درست بودکه 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 . . .

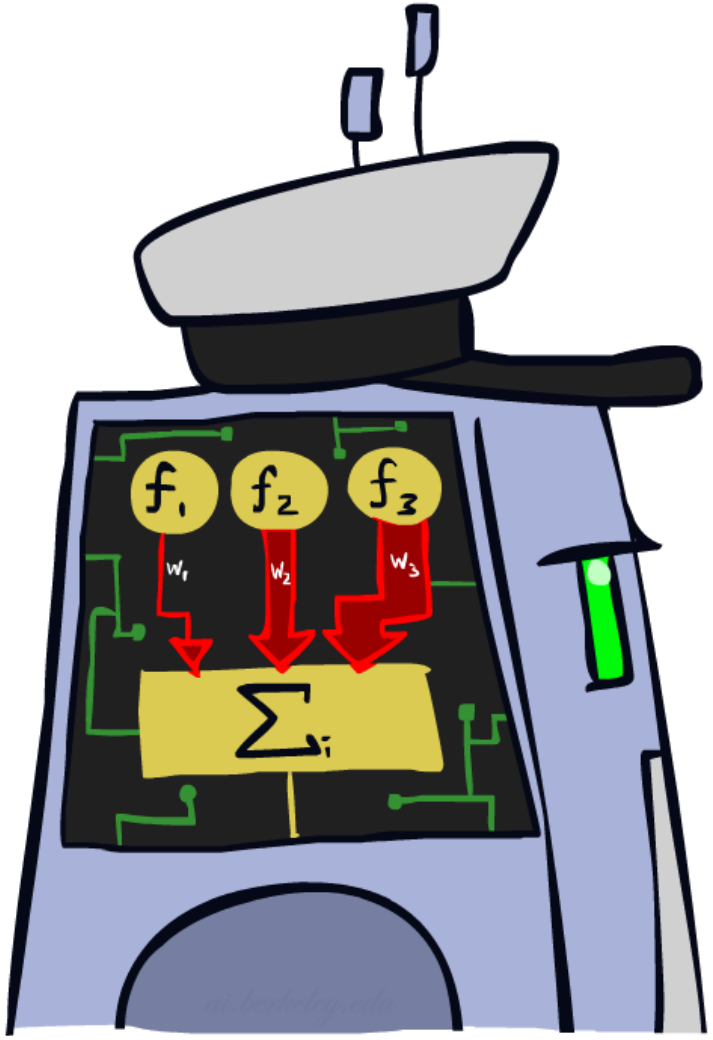
What to Do About Errors

- 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 $f(x)$ y

```
Hello,

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

```
# free       : 2
YOUR_NAME    : 0
MISSPELLED   : 2
FROM_FRIEND  : 0
...
```

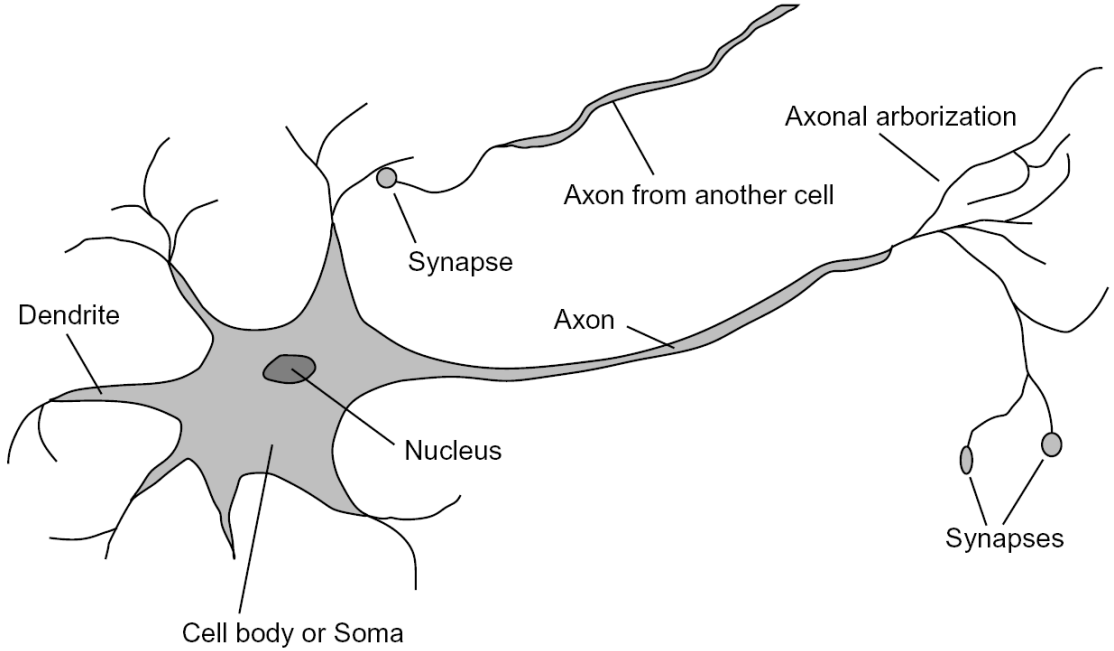
```
PIXEL-7,12   : 1
PIXEL-7,13   : 0
...
NUM_LOOPS    : 1
...
```

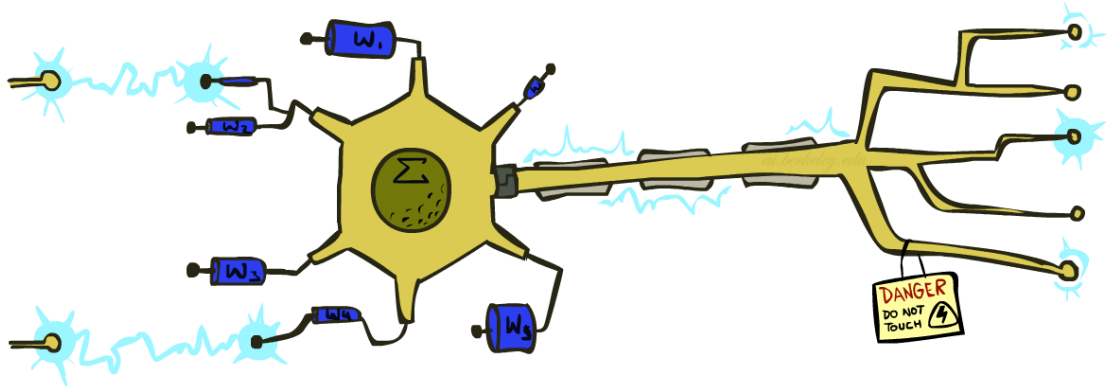
SPAM
or
+

“2”

Some (Simplified) Biology

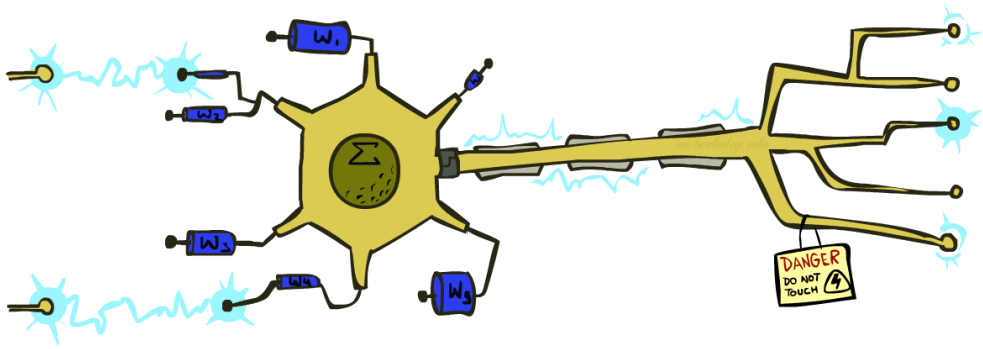
- Very loose inspiration: human neurons





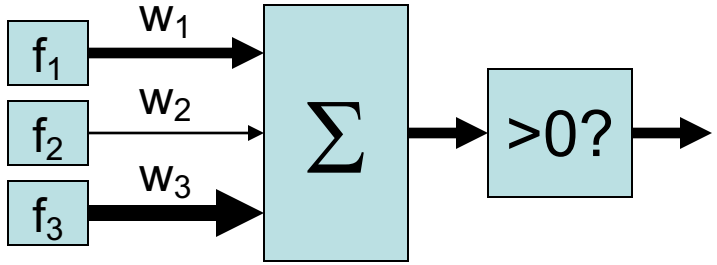
Linear Classifiers

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



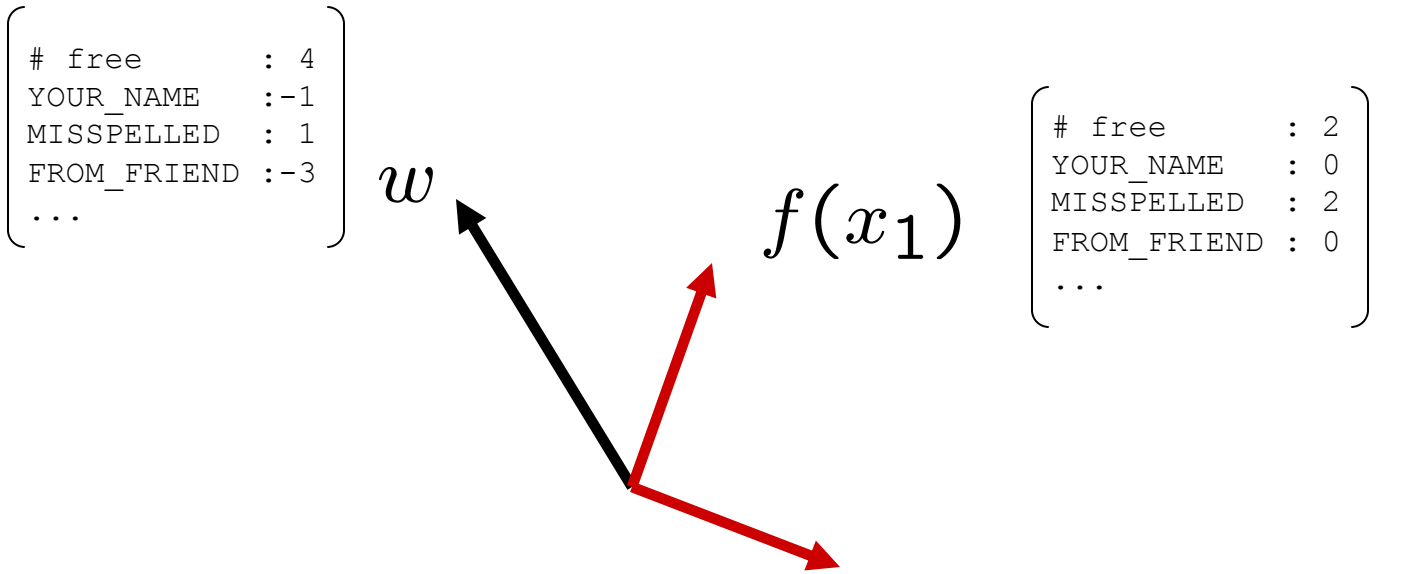
$$\text{activation}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$$

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



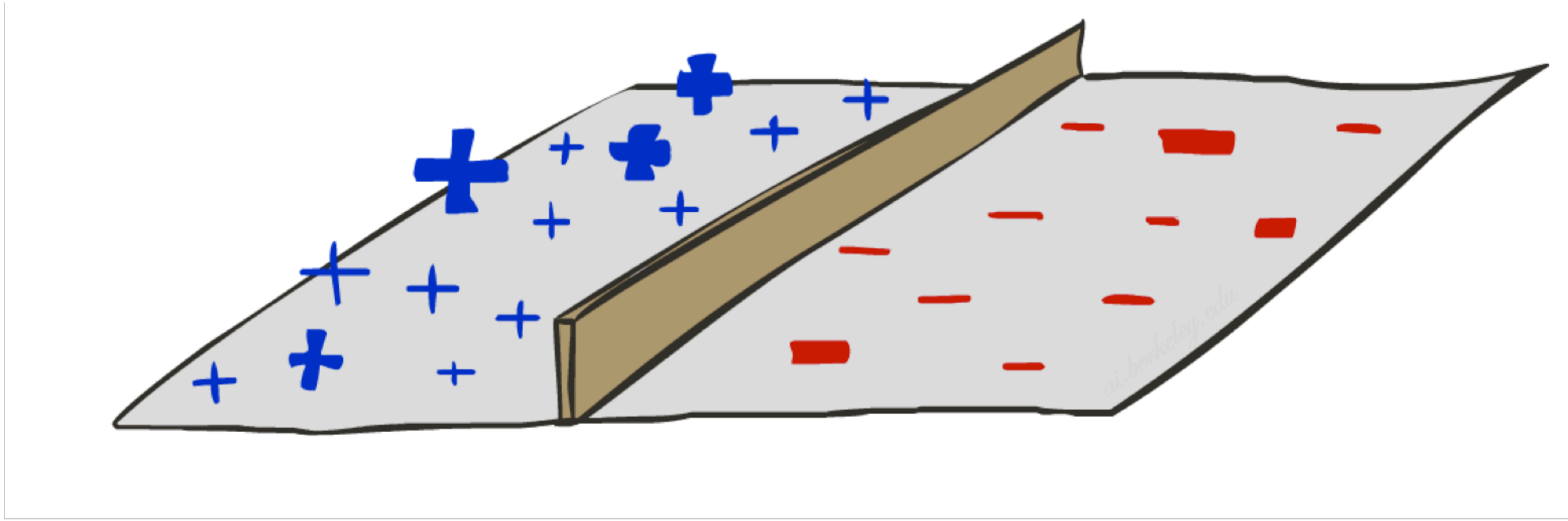
Weights

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



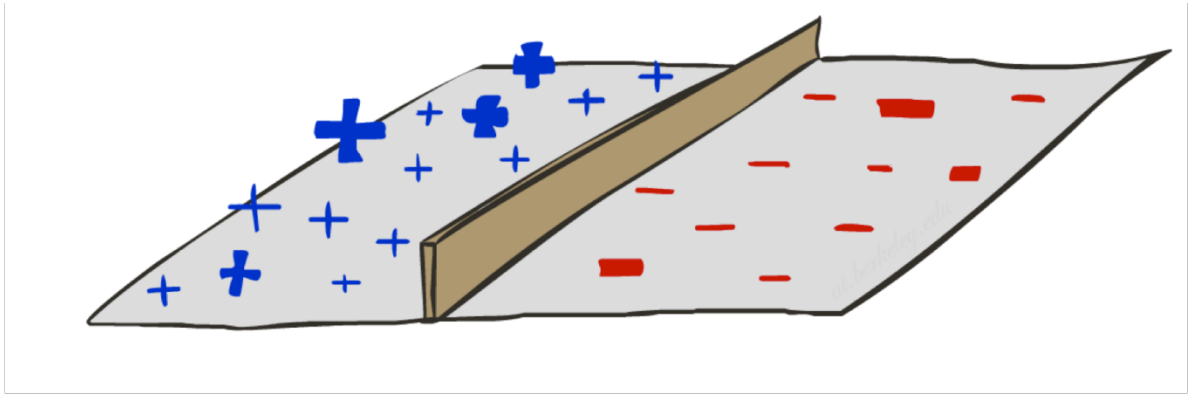
Dot product $w \cdot f$ positive means the positive class

Decision Rules



Binary Decision Rule

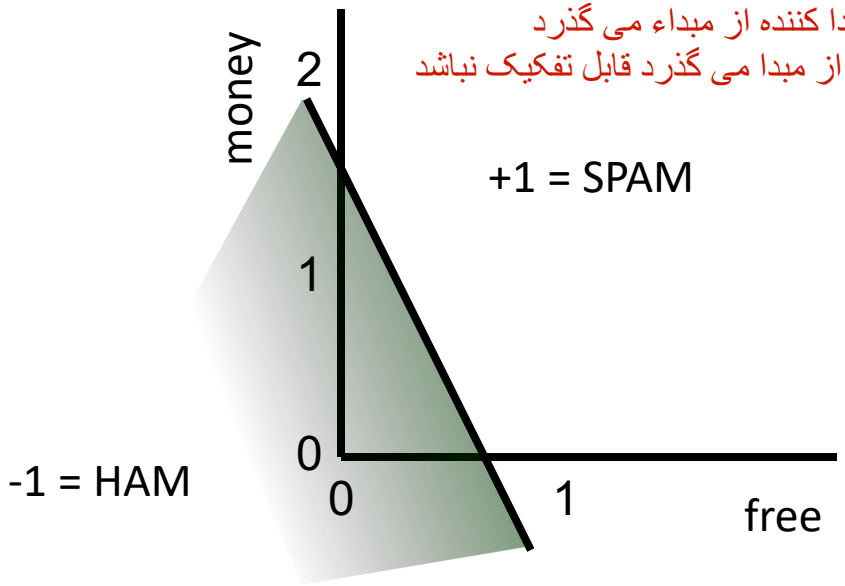
- In the space of feature vectors
  - Examples are points
  - Any weight vector is a hyperplane
  - One side corresponds to Y=+1
  - Other corresponds to Y=-1



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

$$w$$

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

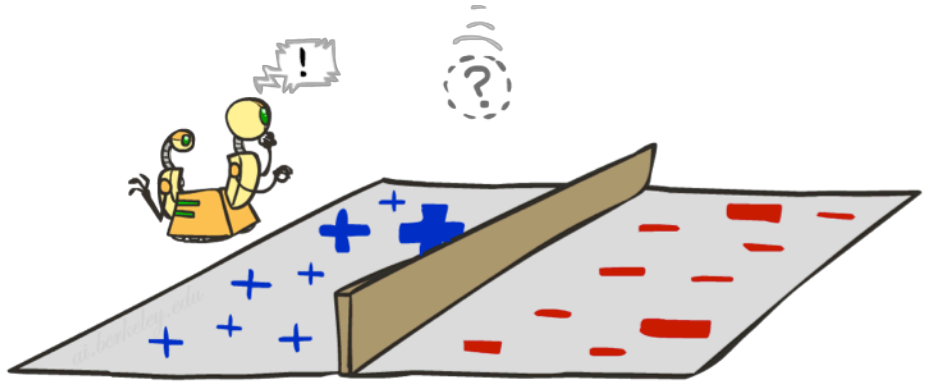


$$f \cdot w = 0$$

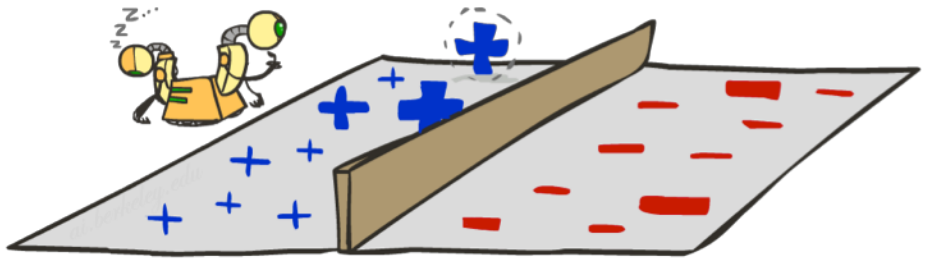
Weight Updates

Learning: Binary Perceptron

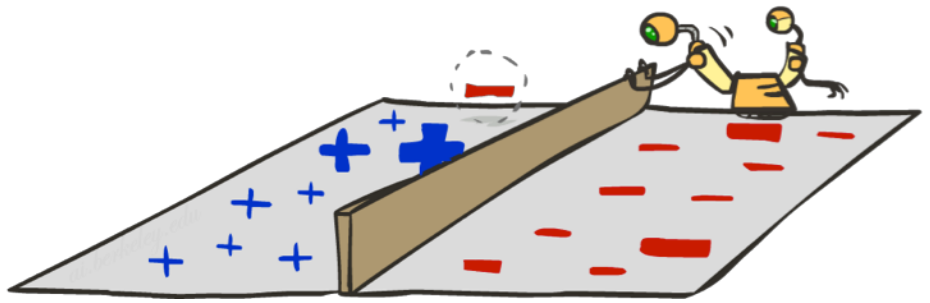
- Start with weights = 0
- For each training instance:
  - Classify with current weights



  - If correct (i.e., y=y*), no change!



  - If wrong: adjust the weight vector



Learning: Binary Perceptron

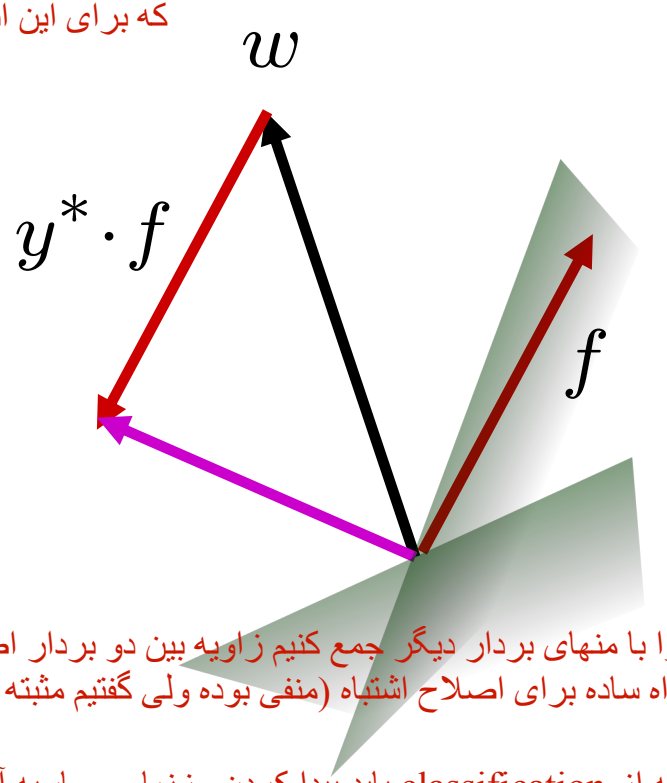
در هر روش classification باید در هر وزن برای هر داده یک مقدار ثابت به نام bias در نظر بگیریم .
که برای این است که تمام خط های جدا کننده از مبدا نگذرد .

فرض می کنیم وزن اولیه برای همه برابر با صفر باشد

- 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)



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

  - 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.

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

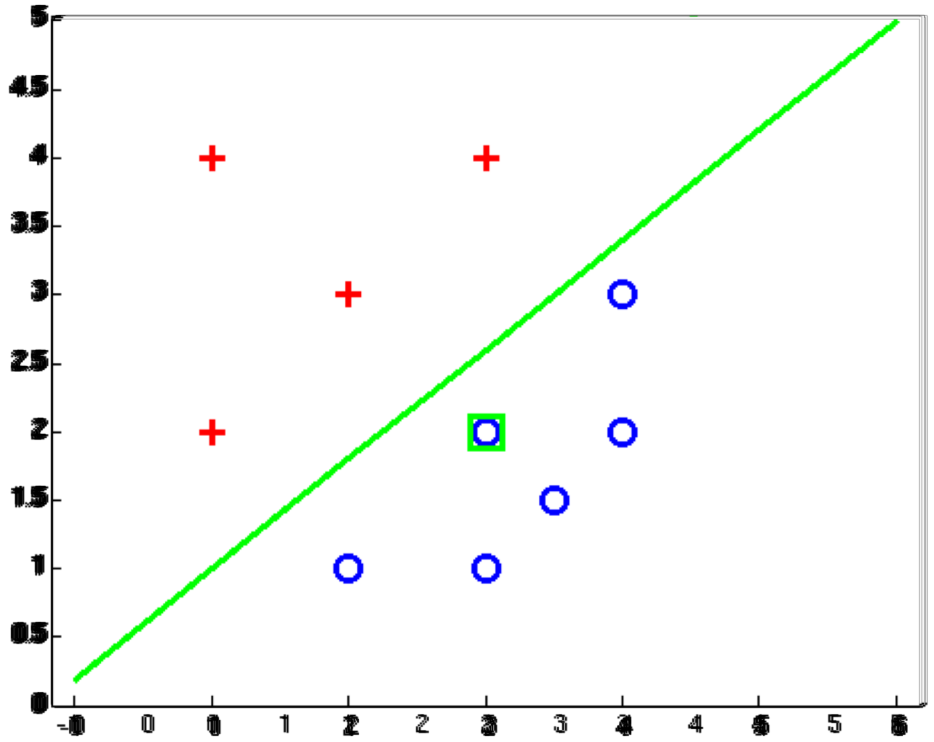
یک مرحله از classification باید پیدا کردن وزنها و محاسبه آن ها
باشد که باید از روی داده های تمرینی باید تعیین شود .

به جایی می رسیم 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 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 تغییری نمی کند
و همگرا می شود



Multiclass Decision Rule

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

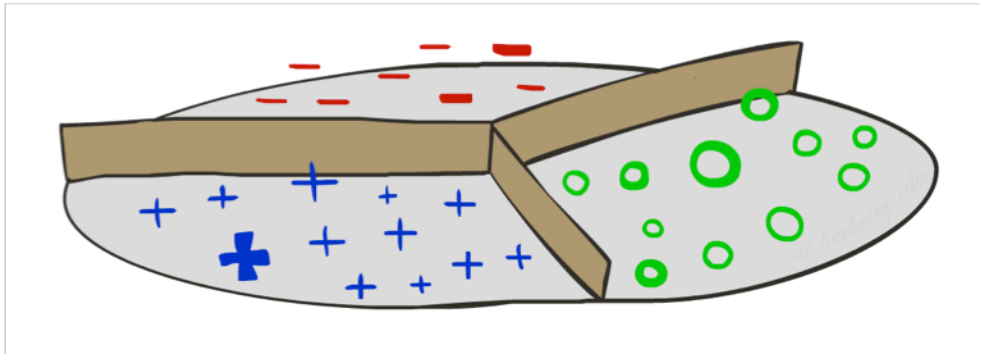
$$w_y$$

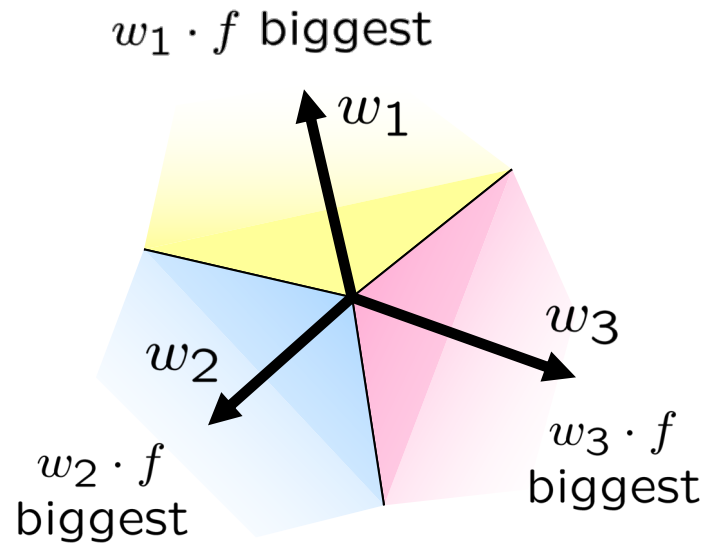
  - Score (activation) of a class y:

$$w_y \cdot f(x)$$

  - Prediction highest score wins

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





Binary = multiclass where the negative class has weight zero

Learning: Multiclass Perceptron

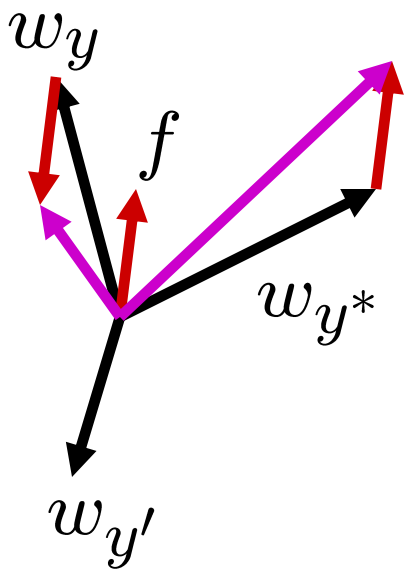
- Start with all weights = 0
- Pick up training examples one by one
- Predict with current weights

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

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

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

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



Example: Multiclass Perceptron

"win the vote"

"win the election"

"win the game"

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}

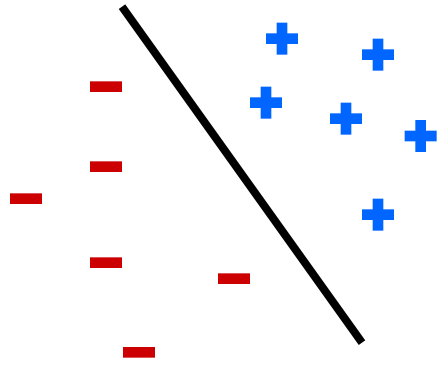
```
BIAS  : 0
win   : 0
game  : 0
vote  : 0
the   : 0
...
```

Properties of Perceptrons

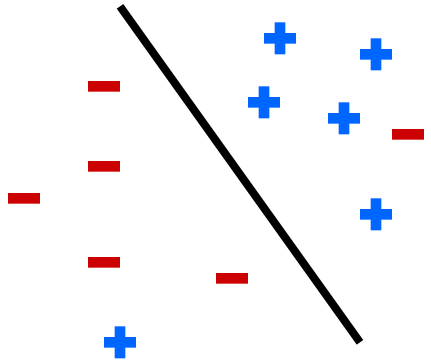
- Separability: true if some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

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

Separable

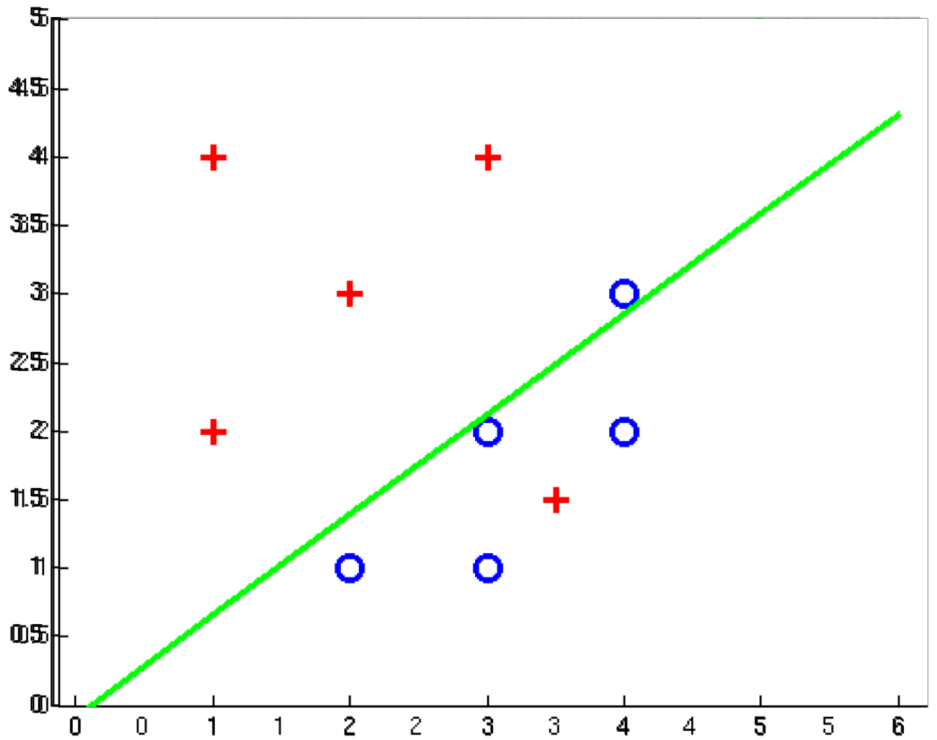


Non-Separable

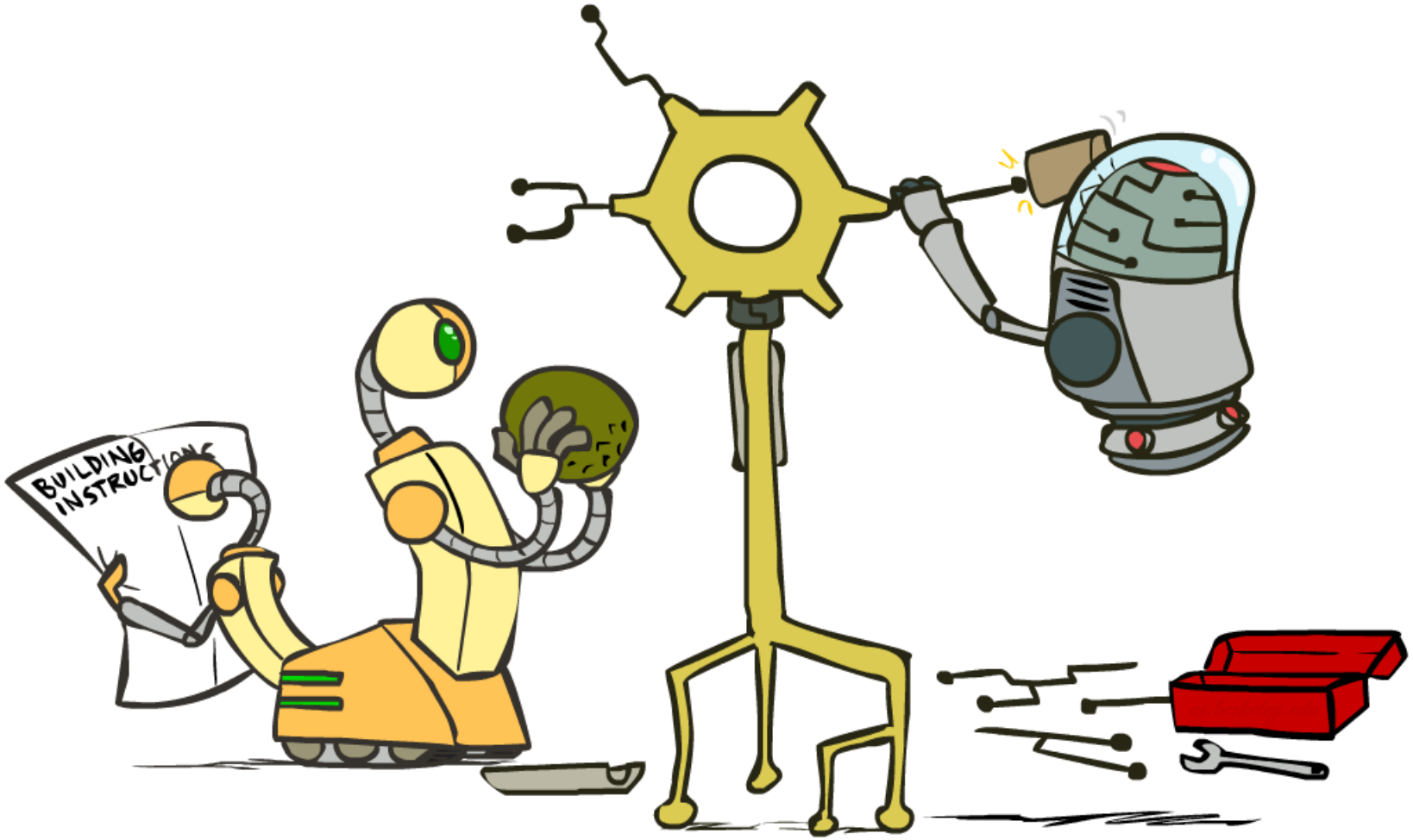


Examples: Perceptron

- Non-Separable Case

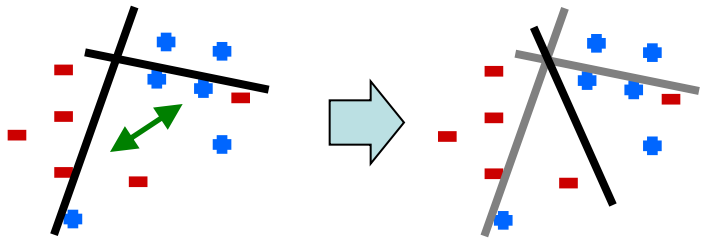


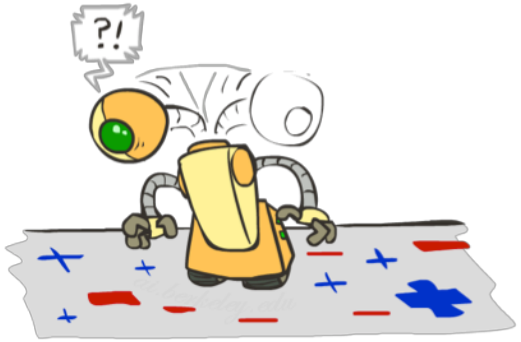
Improving the Perceptron



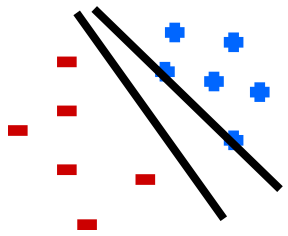
Problems with the Perceptron

- Noise: if the data isn't separable, weights might thrash
  - Averaging weight vectors over time can help (averaged perceptron)

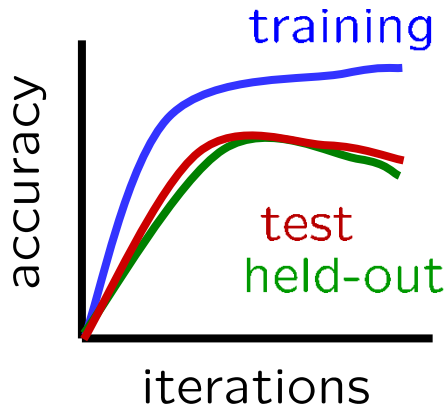


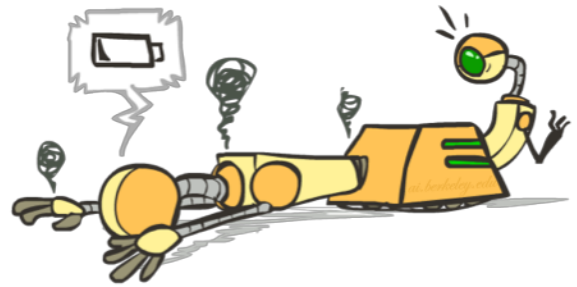


- Mediocre generalization: finds a "barely" separating solution



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





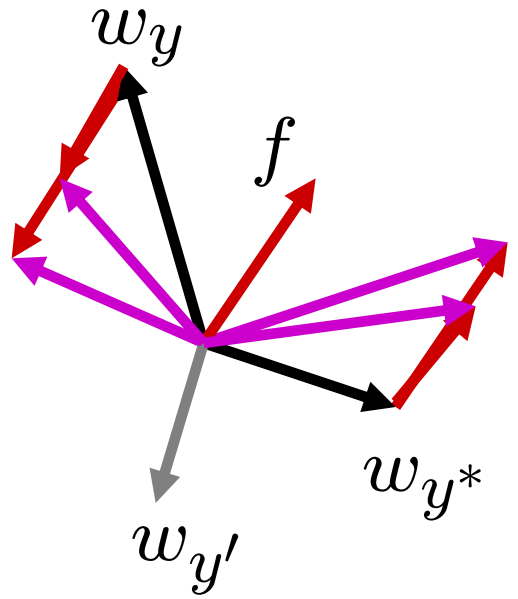
Fixing the Perceptron

- 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

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

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

- The +1 helps to generalize



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

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

* Margin Infused Relaxed Algorithm

Minimum Correcting Update

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

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

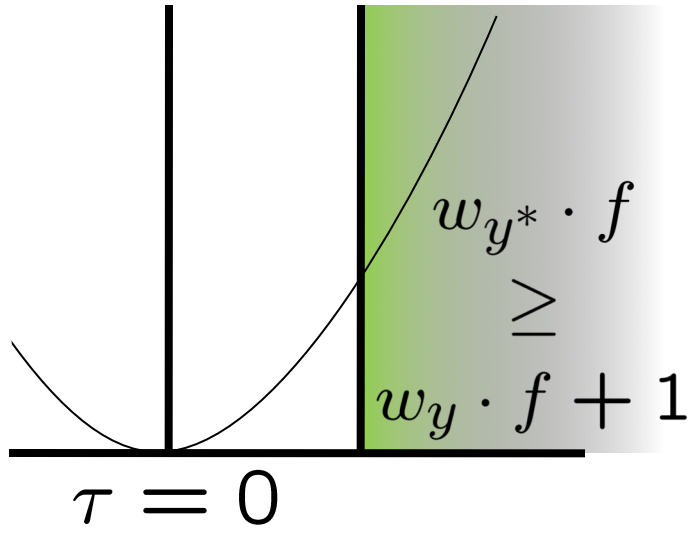
$$\min_\tau ||\tau f||^2$$

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

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

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

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

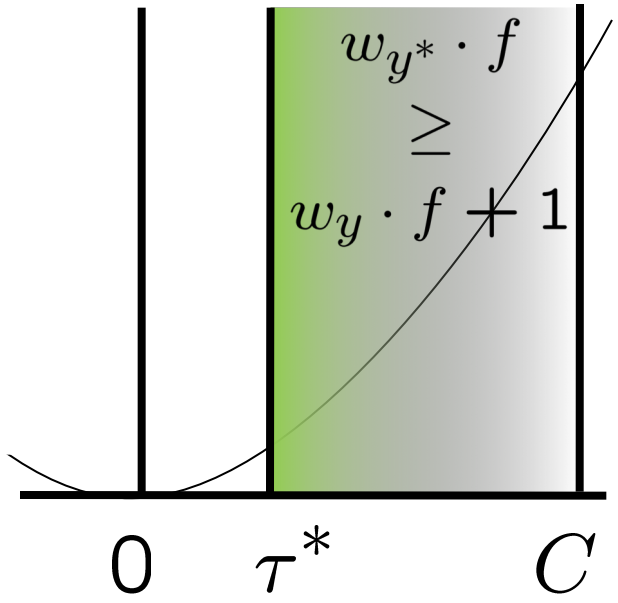


min not τ=0, or would not have made an error, so min will be where equality holds

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 τ with some constant C

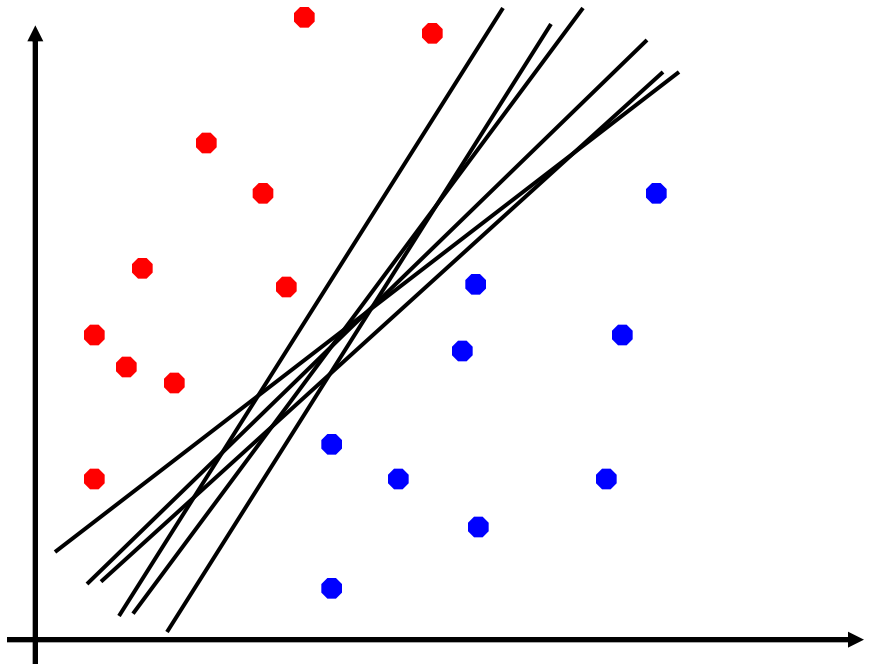
$$\tau^* = \min\left(\frac{(w'_y - w'_{y^*}) \cdot f + 1}{2f \cdot f}, C\right)$$



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

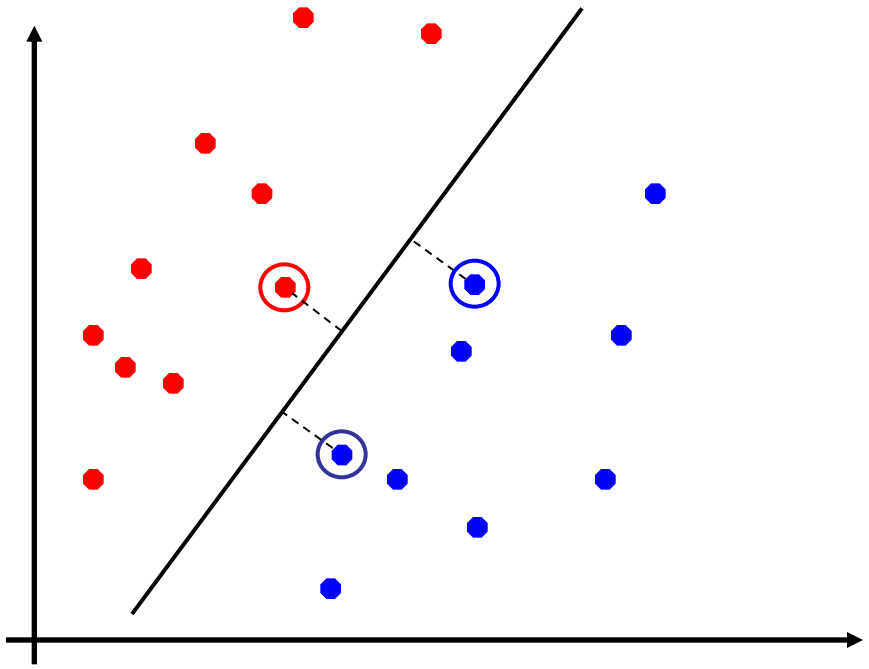
Linear Separators

- Which of these linear separators is optimal?



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



MIRA

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

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

SVM

$$\min_w \frac{1}{2}||w||^2$$

$$\forall i, y \; w_{y^*} \cdot f(x_i) \geq w_y \cdot f(x_i) + 1$$