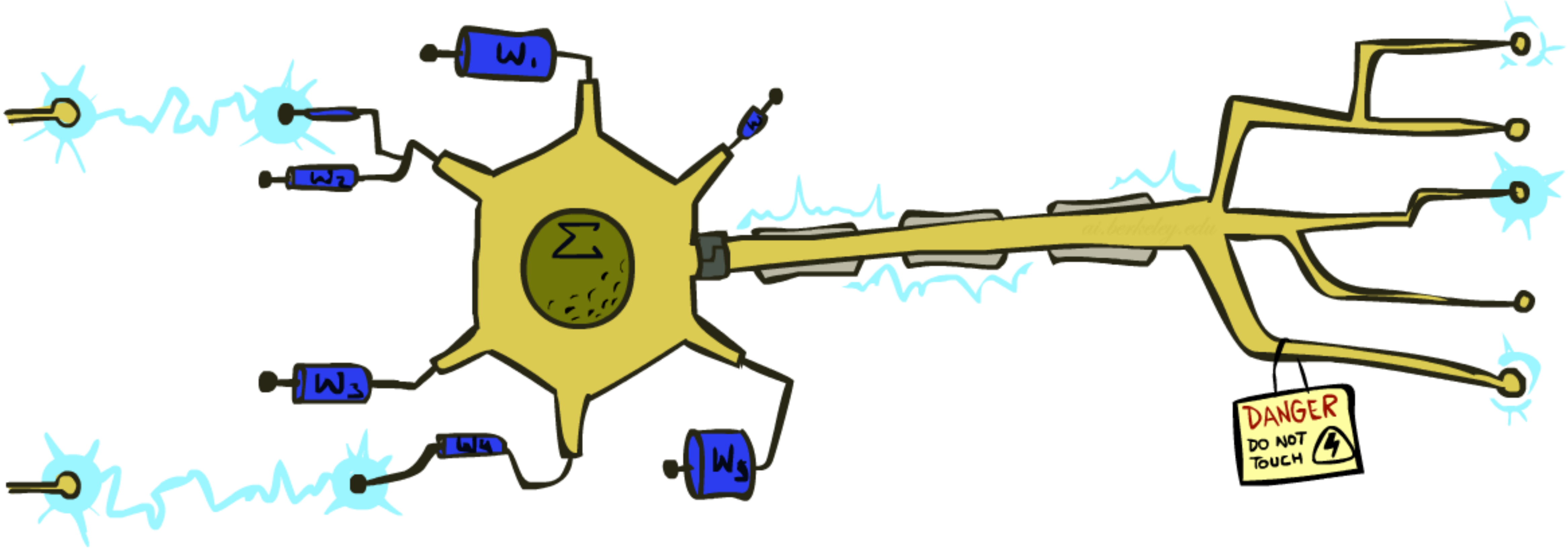
Announcements

- Midterm 2
  - Next Wednesday Nov 9 at 7:00 pm
  - Covers everything up to last Thursday (nothing this week)
  - OK to bring a two sided, handwritten cheat sheet.

- Adam's Office Hours moved (permanently).
  - Will be Thursday 11:30 – 12:30, 329 Soda

CS 188: Artificial Intelligence

Perceptrons

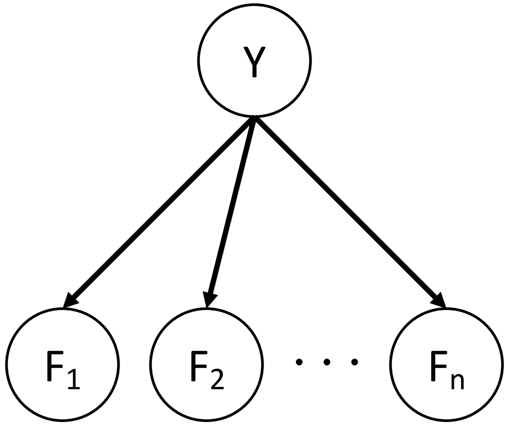


Instructors: Adam Janin, Josh Hug --- University of California, Berkeley

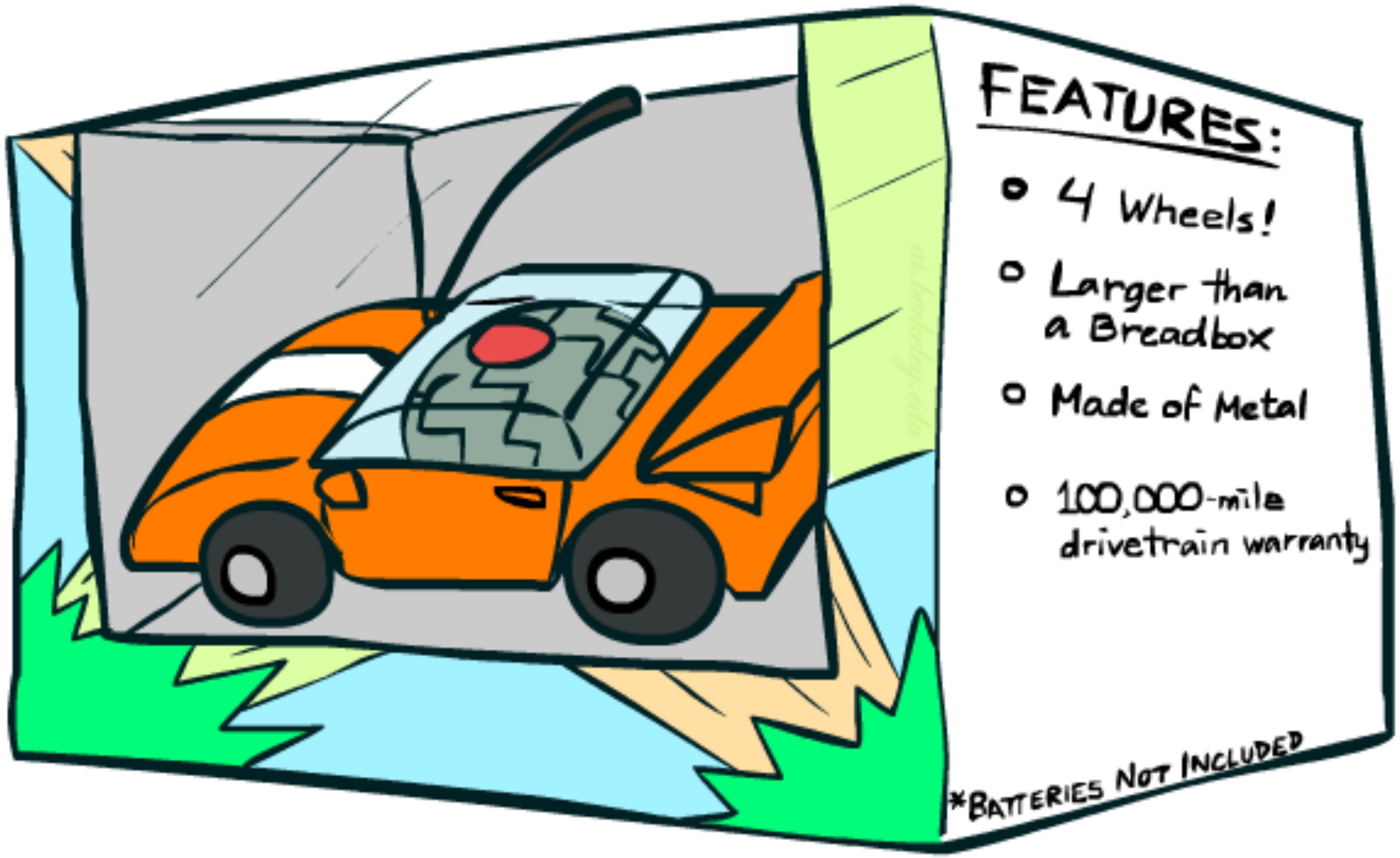
[These slides created by Dan Klein, Pieter Abbeel, Anca Dragan, Josh Hug for CS188 Intro to AI at UC Berkeley. All CS188 materials available at http://ai.berkeley.edu.]

Summary From Last Time

- Can perform classification by creating a Bayes Net model and performing probabilistic inference
- The Naïve Bayes model assumes all features to be independent given the class label
- Can infer parameters (values for our CPTs) using “training data”
- Smoothing our estimates is important in real systems
  - Zero probabilities in our CPTs is a particularly deadly type of overfitting
  - Use “held-out” data to tune smoothing hyperparameters



Features



FEATURES:
4 Wheels!
Larger than a Breadbox
Made of Metal
100,000-mile drivetrain warranty
*BATTERIES NOT INCLUDED

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?

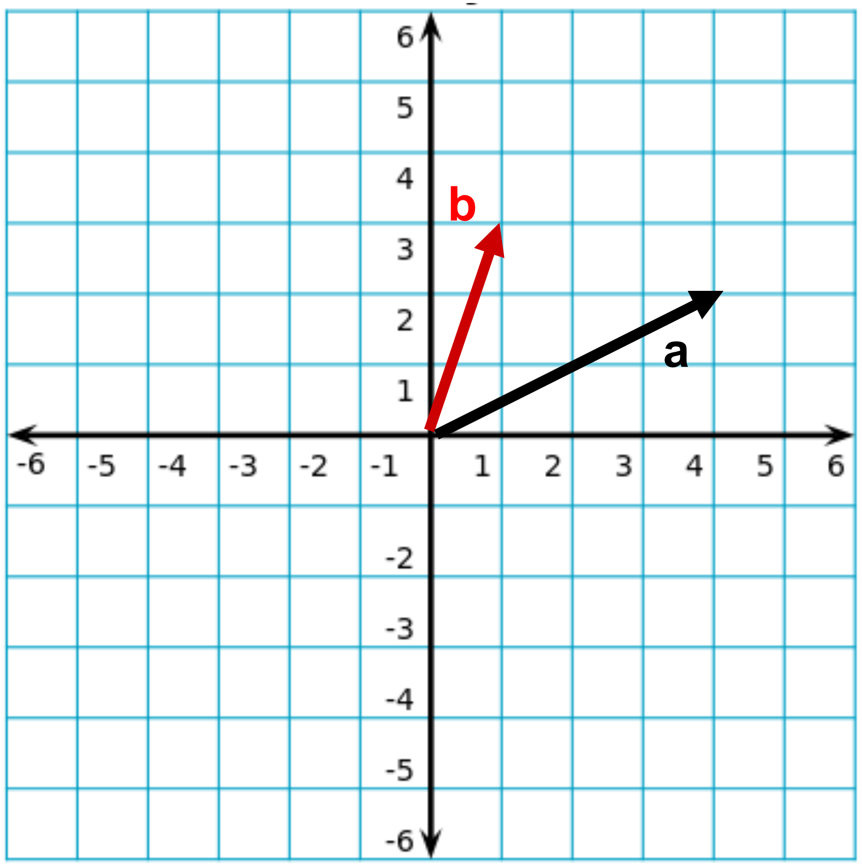
- Need more features– words aren't enough!
  - Have you emailed the sender before?
  - Have 1K other people just gotten the same email?
  - Is the sending information consistent?
  - Is the email in ALL CAPS?
  - Do inline URLs point where they say they point?
  - Does the email address you by (your)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

- Can add these information sources as new variables in the NB model

- This lecture: A new type of classifier which is trained by reacting to errors.

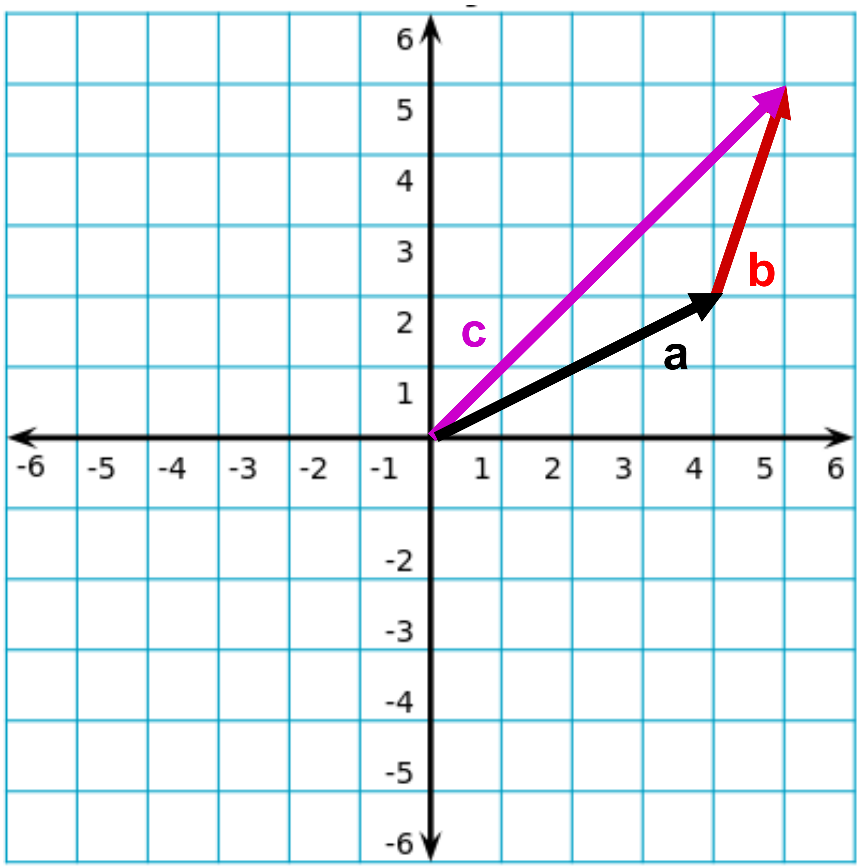
c Very Brief Vector Review

- Suppose we have vectors $\mathbf{a} = \langle 4, 2 \rangle$, $\mathbf{b} = \langle 1, 3 \rangle$
  - What will **c** = **a** + **b** look like?
  - What will **d** = **a** – **b** look like?



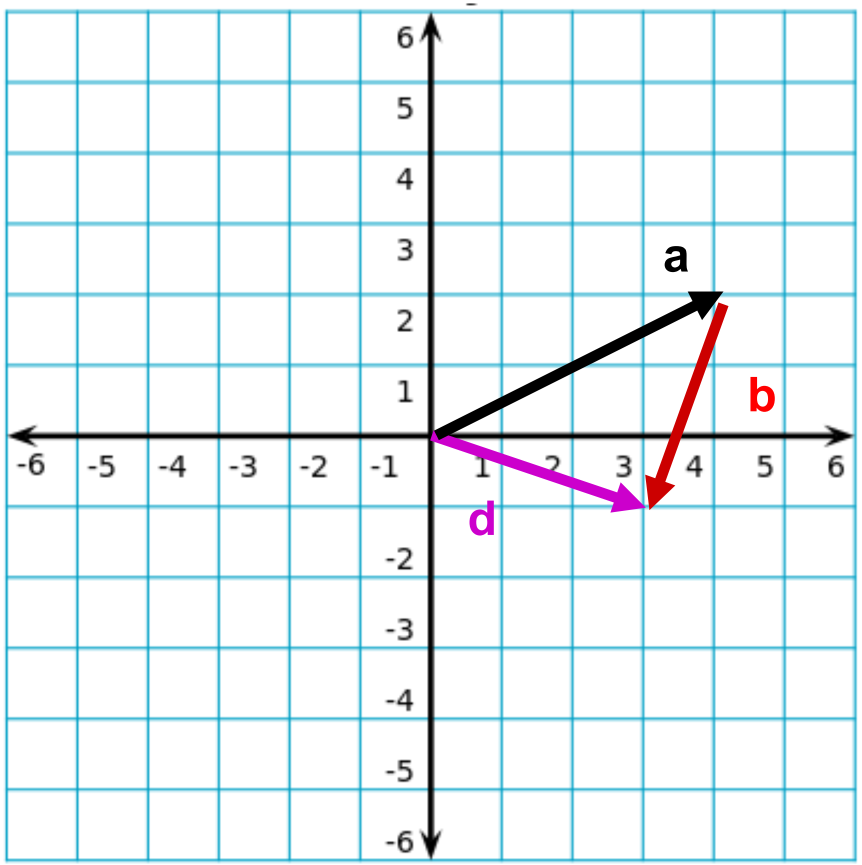
c Very Brief Vector Review

- Suppose we have vectors $\mathbf{a} = \langle 4, 2 \rangle$, $\mathbf{b} = \langle 1, 3 \rangle$
  - What will **c** = **a** + **b** look like?
    - $\mathbf{c} = \langle 5, 5 \rangle$
  - What will **d** = **a** – **b** look like?



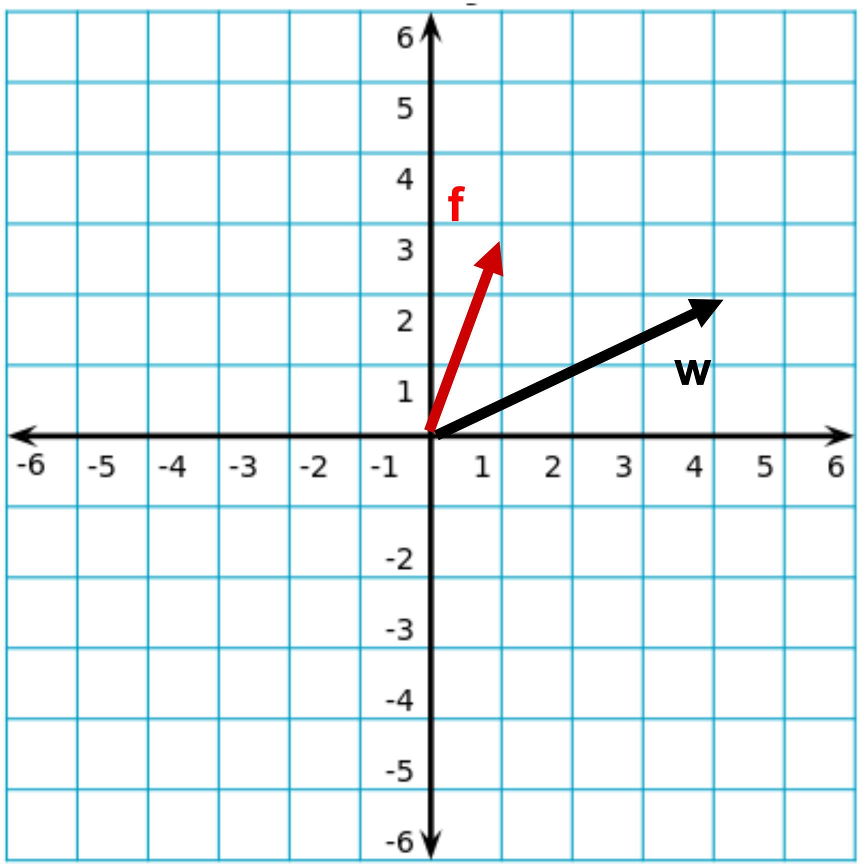
Very Brief Vector Review

- Suppose we have vectors $\mathbf{a} = \langle 4, 2 \rangle$, $\mathbf{b} = \langle 1, 3 \rangle$
  - What will **c** = **a** + **b** look like?
    - $\mathbf{c} = \langle 5, 5 \rangle$
  - What will **d** = **a** – **b** look like?
    - $\mathbf{d} = \langle 3, -1 \rangle$



c Very Brief Vector Review

- Suppose we have vectors $\mathbf{w} = \langle 4, 2 \rangle$ and $\mathbf{f} = \langle 1, 3 \rangle$, what is $\boldsymbol{w} \cdot \mathbf{f}$?
  - 4*1 + 3*2 = 10
  - Angle between vectors is < 90, so dot product is > 0.



- Reminder:
  - $\mathbf{w} \cdot \mathbf{f} = ||\mathbf{w}|| \; ||\mathbf{f}|| \cos(\theta)$
  - $\mathbf{w} \cdot \mathbf{f} = \sum_{i=1}^{n} w_i f_i$

c Very Brief Vector Review

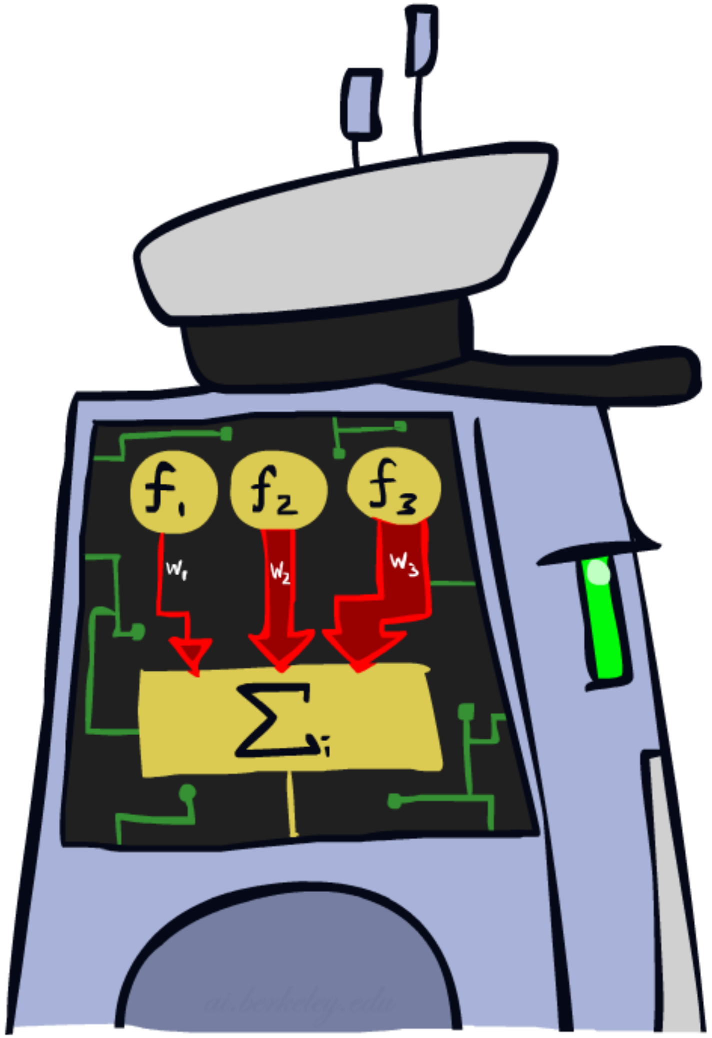
- Suppose we have vectors $\mathrm{w} = \langle -3, 4, 2 \rangle$ and $\mathrm{f} = \langle 1, 0, 1.5 \rangle$, , what is $\boldsymbol{w} \cdot \mathbf{f}$?
  - -3*1 + 4*0 + 2*1.5 = 0
  - Angle between the vectors is 90 degrees, so dot product is zero.

- Reminder: $\mathbf{w} \cdot \mathbf{f} = ||\mathbf{w}|| \; ||\mathbf{f}|| \cos(\theta)$

- Vector plotting app (to visualize):
  - https://academo.org/demos/3d-vector-plotter/

Error-Driven Classification

Linear Classifiers

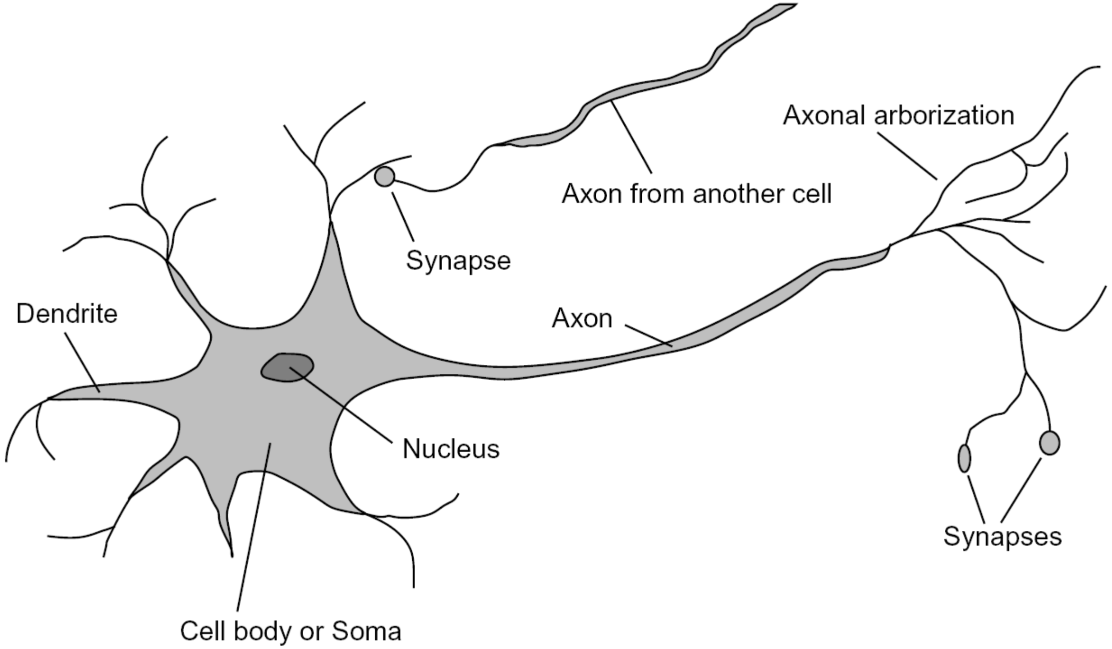


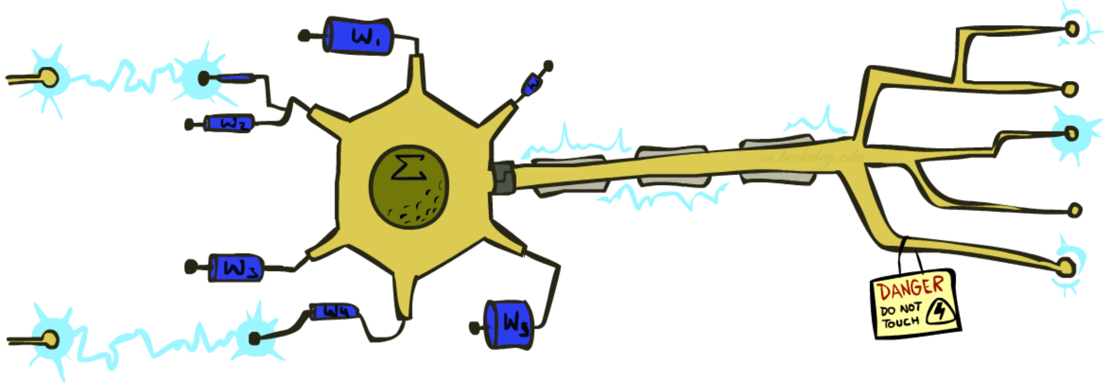
Feature Vectors (from last time)

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 ...	→	SPAM or +
	→	PIXEL-7,12 : 1 PIXEL-7,13 : 0 ... NUM_LOOPS : 1 ...	→	“2”

Some (Simplified) Biology

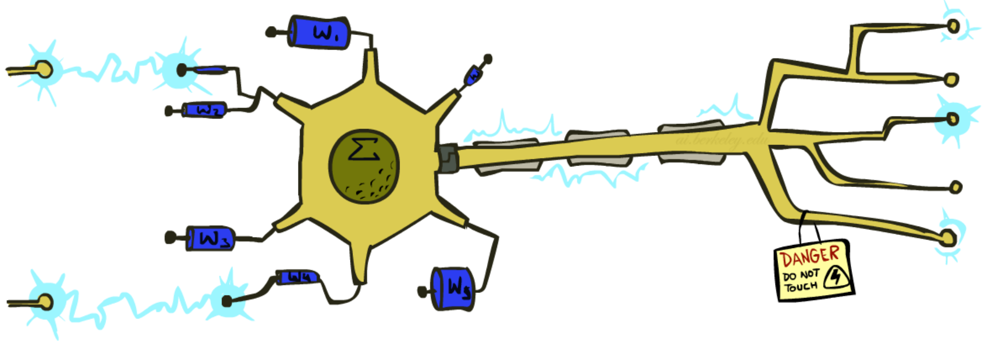
- Very loose inspiration: human neurons
  - Don’t get too excited, the metaphor is pretty far from reality.





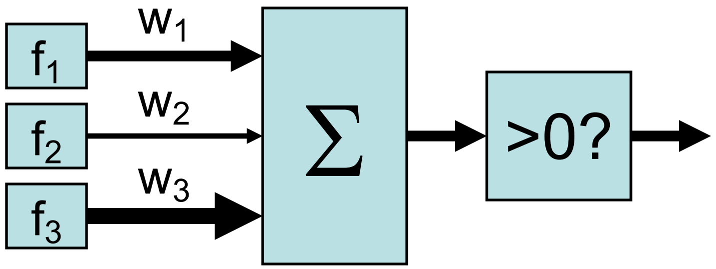
Linear Classifiers

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

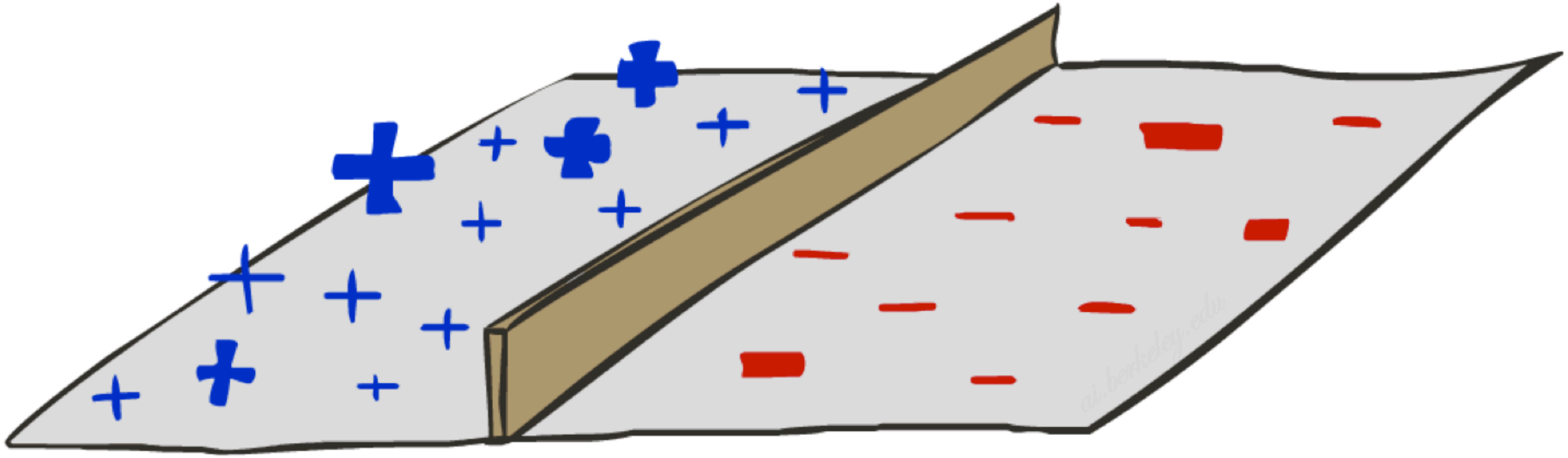


$$\text{activation}_{\mathbf{w}}(\mathrm{x}) = \sum_i w_i f_i(x) = \mathbf{w} \cdot \mathbf{f}(x)$$

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



Decision Rules



Binary Decision Rule

- If we have two classes (i.e. need to make binary decision):
  - Have a weight vector for the “positive” class, e.g. spam

$$\mathbf{w}$$

  - For any given input x, perceptron calculates activation level on that input’s features.

$$\text{activation}_{\mathbf{w}}(\mathrm{x}) = \mathbf{w} \cdot \mathbf{f}(x)$$

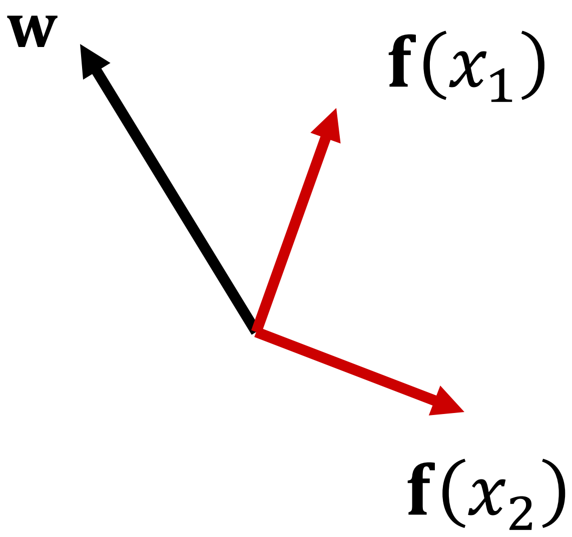
  - If activation is positive, perceptron believes it has seen the positive class.
    - Sort of like this: https://www.youtube.com/watch?v=l7op92W7voE

$$y = \begin{cases} +1, \text{if activation}_{\mathbf{w}}(\mathrm{x}) \geq 0 \\ -1, \text{if activation}_{\mathbf{w}}(\mathrm{x}) < 0 \end{cases}$$

Binary Decision Rule: Geometric Interpretation #1

- Classification rule: if angle between weight vector and feature vector < 90, assume the positive class.

```
d1scount     : 4
YOUR_NAME    :-1
MISSPELLED   : 1
FROM_FRIEND  :-3
...
```




```
d1scount     : 2
YOUR_NAME    : 0
MISSPELLED   : 2
FROM_FRIEND  : 0
...
```

```
d1scount     : 0
YOUR_NAME    : 1
MISSPELLED   : 1
FROM_FRIEND  : 1
...
```

Dot product $\mathbf{w} \cdot f(x)$ *positive means the positive class*

Bias Trick

- Suppose we want to take into account that one class is inherently more likely:
  - Set first entry of weight vector to a so-called “BIAS” value.
  - Set first entry of feature vector to 1, regardless of input.

- Net effect of this trick is to add a constant offset to our activation.
  - Similar to a y-intercept in slope-intercept form of a line.

$$\mathrm{activation}_{\mathbf{w}}(\mathrm{x}) = \sum_{i=0} w_i f_i(x) = \mathbf{w} \cdot \mathbf{f}(x)$$

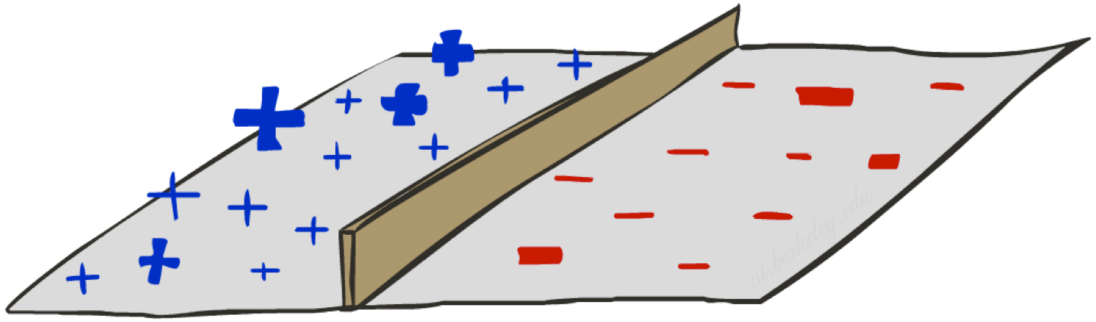
$$= BIAS * 1 + \sum_{i=1} w_i f_i(x)$$

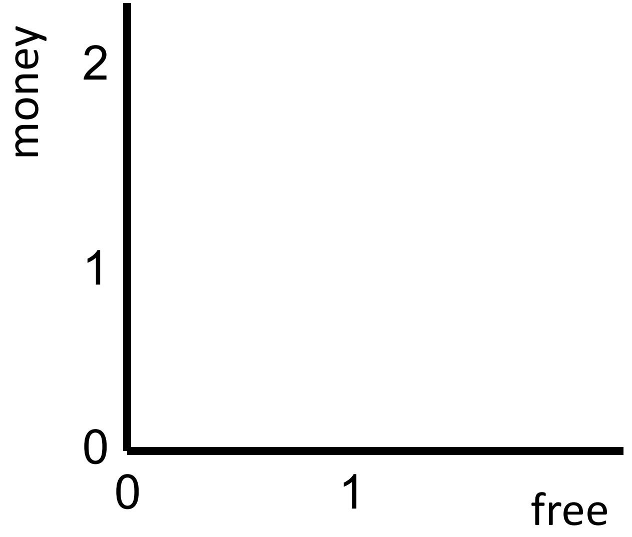
Binary Decision Rule: Geometric Interpretation #2

- 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

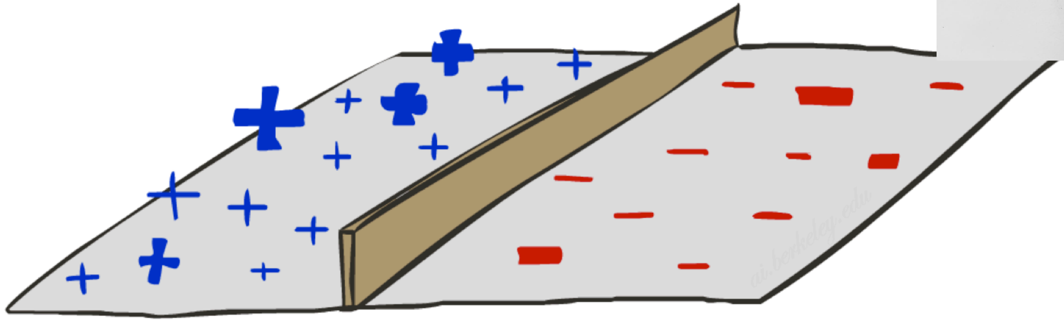
```
BIAS  : -3
free  :  4
money :  2
...
```





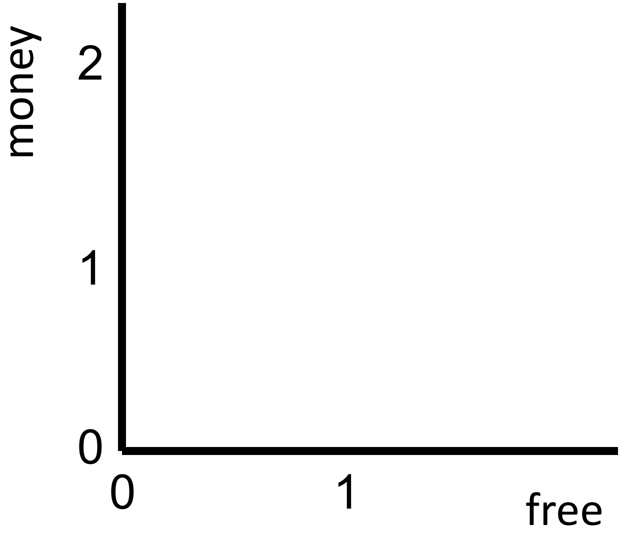
Find the Separating Hyperplane (line)

- 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
...
```

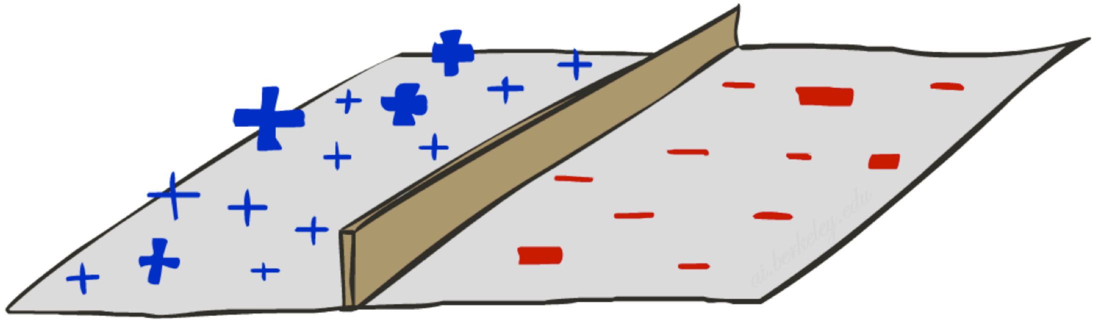


Draw the line that separates "ham" from "spam", and label the two resulting half-planes.

Hint: when is $\mathbf{f} \cdot \boldsymbol{w} = 0$?

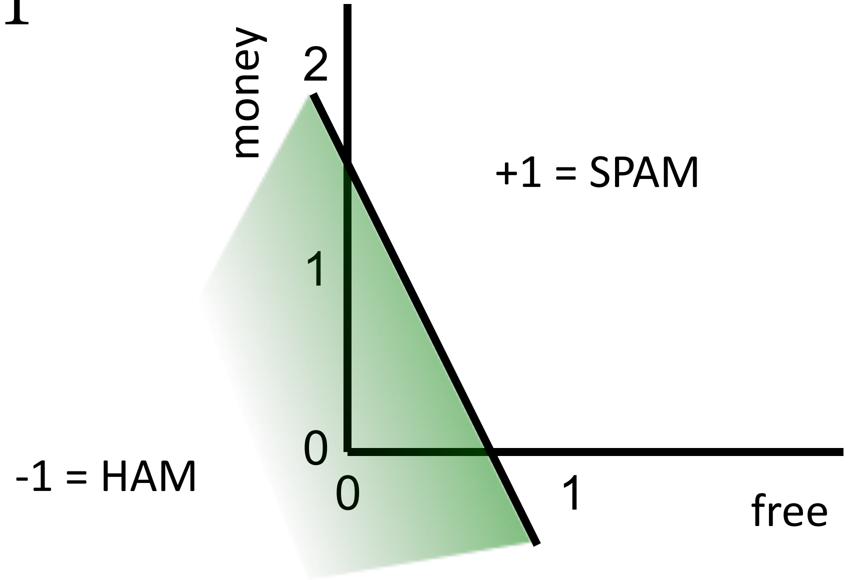
Binary Decision Rule: Geometric Interpretation #2

- 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

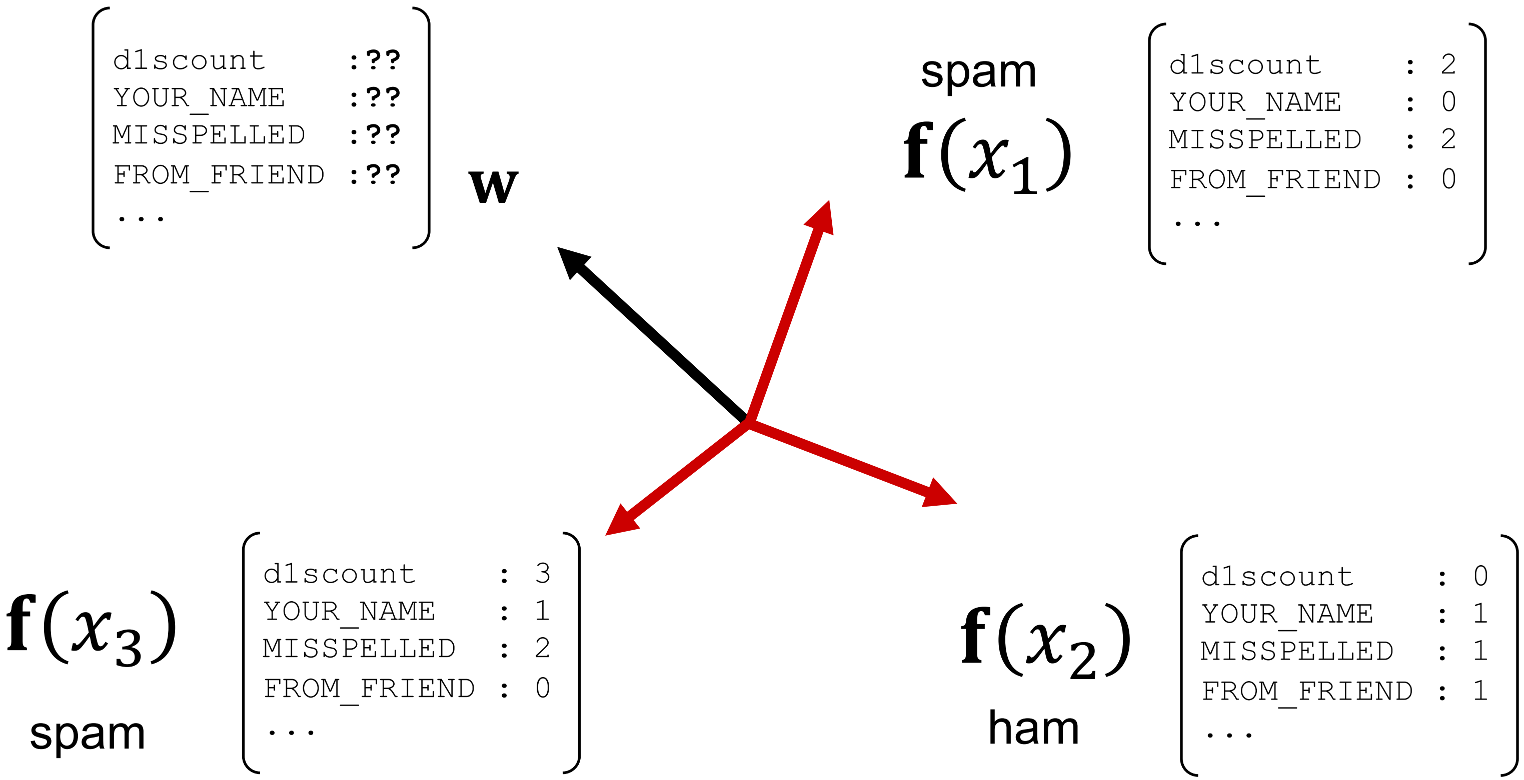
```
BIAS  : -3
free  :  4
money :  2
...
```



$$\mathbf{f} \cdot \mathbf{w} = 0$$

Missing Piece: Learning the Weight Vector

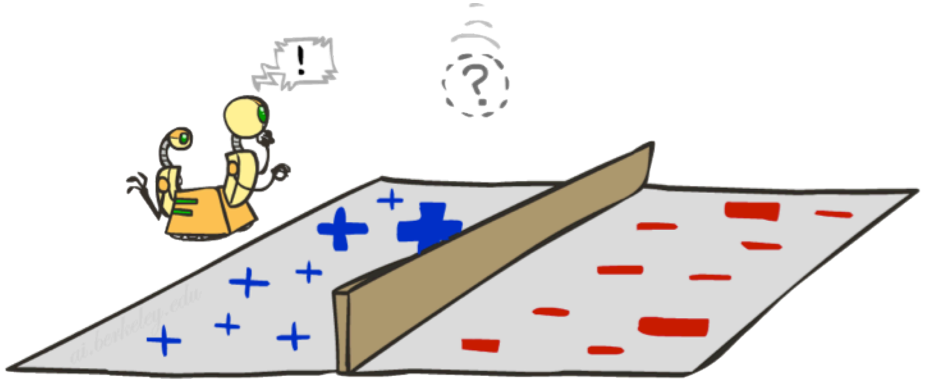
- Need a procedure for learning.
  - Input: huge number of labeled feature vectors
  - Output: weight vector



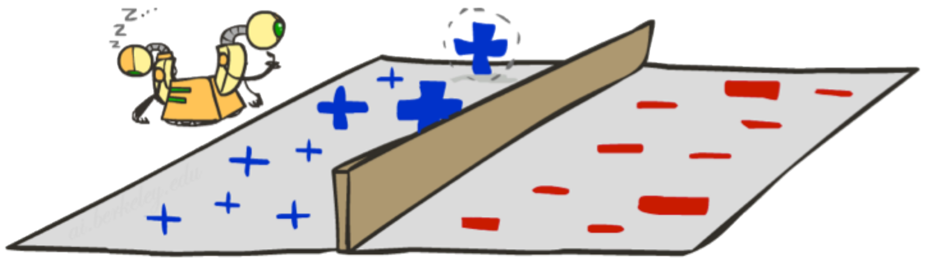
Weight Updates

Learning: Binary Perceptron

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

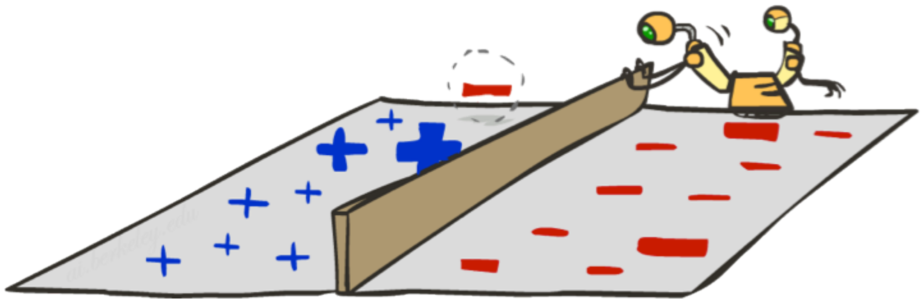


  - If correct (i.e., $y = y^*$), no change!



y^* is the correct value for a given input

  - If wrong: adjust the weight vector

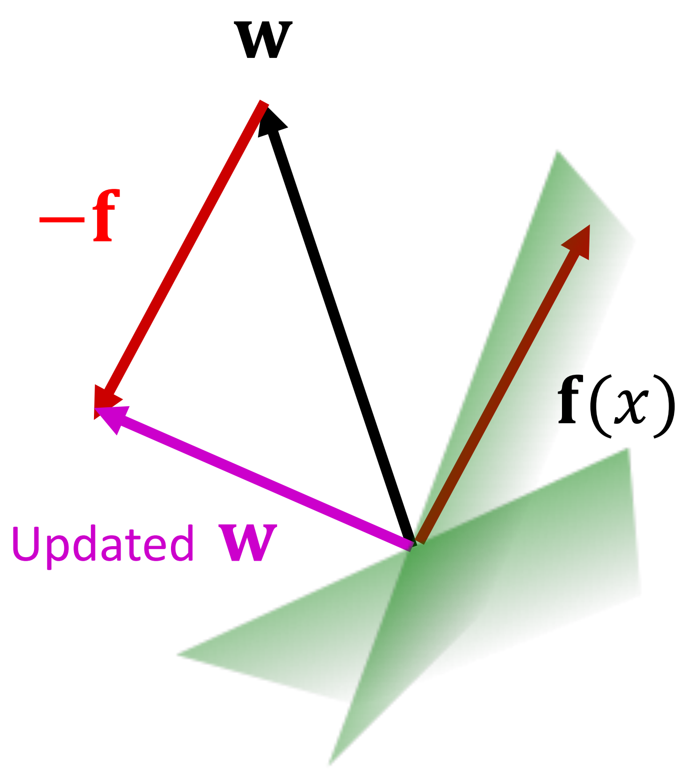


Learning: Binary Perceptron

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

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

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



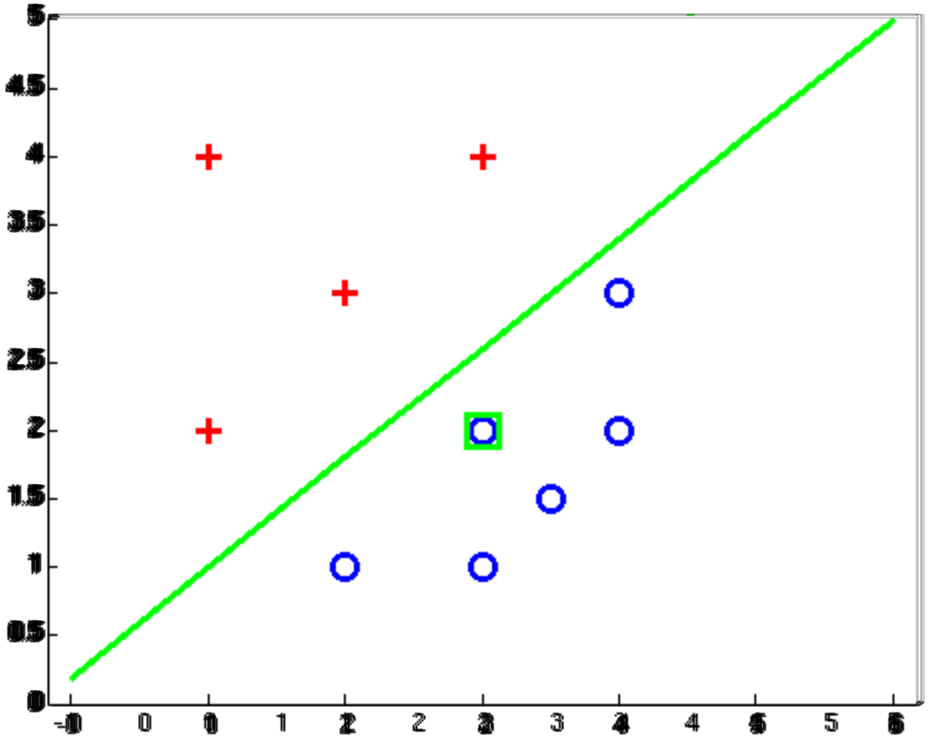
Predicted -, was +: $\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} + \mathbf{f}$

Predicted +, was -: $\mathbf{w}^{\text{new}} = \mathbf{w}^{\text{old}} - \mathbf{f}$

Suppose $y^*(x)$ = ham, a.k.a. -1
But we predicted y(x) = spam/+1

Examples: Perceptron

- Separating plane jumps fairly erratically before settling down.
  - In each image, we’re seeing what happened AFTER updating weight.



Multiclass Decision Rule

- If we have multiple classes:
  - One weight vector for each class:
    - Can think of all vectors forming a matrix.

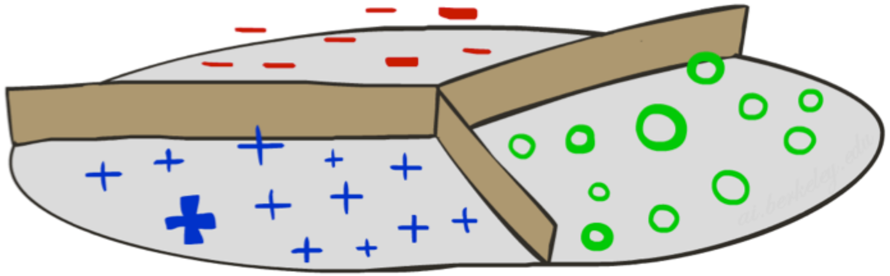
$$\mathbf{w_y} \qquad\qquad W = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_L \end{bmatrix}$$

  - Score (activation) of a class y:

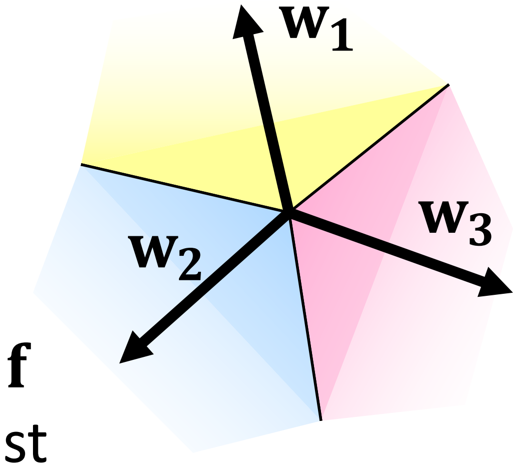
$$\text{activation}_{\mathbf{w_y}}(\mathrm{x}) = \mathbf{w_y} \cdot \mathbf{f}(x)$$

  - Prediction highest score wins

$$y = \arg\max_{y} \mathbf{w_y} \cdot \mathbf{f}(x)$$



$\mathbf{w_1} \cdot \mathbf{f}$ biggest



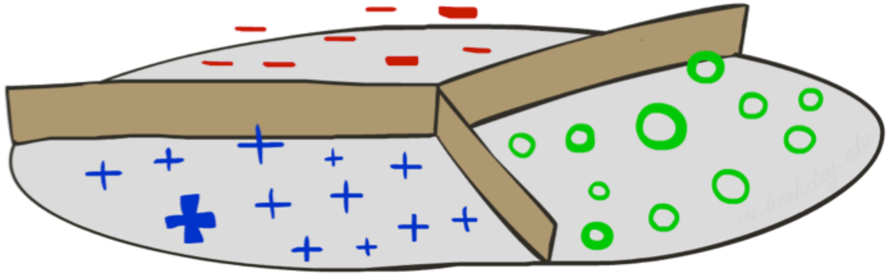
$\mathbf{w_2} \cdot \mathbf{f}$ biggest

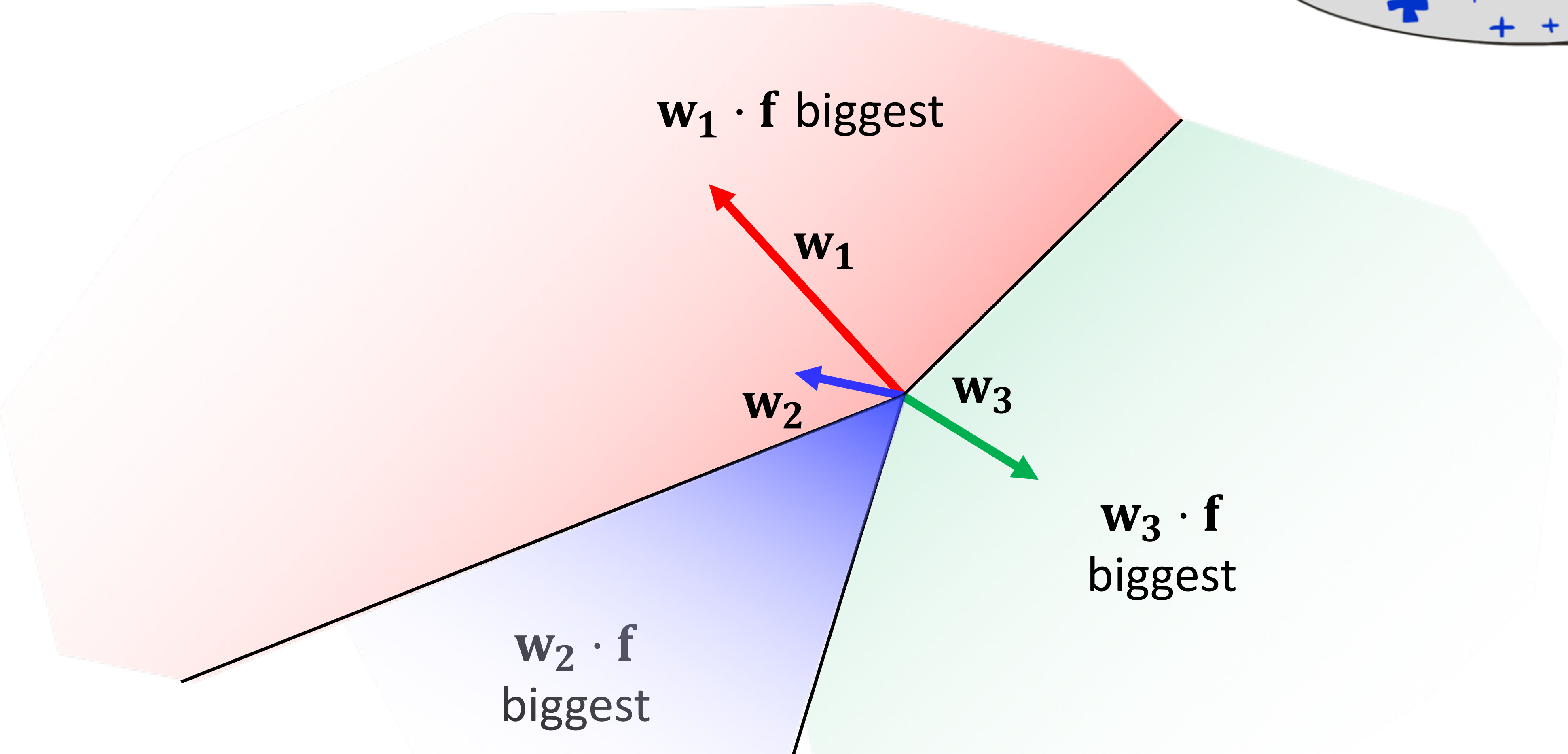
$\mathbf{w_3} \cdot \mathbf{f}$ biggest

Binary = multiclass where the negative class has weight zero

Geometry Warning!

- In the multiclass case, angle is not the only determining factor.
  - Magnitude of weight vectors matter!





Learning: Multiclass Perceptron

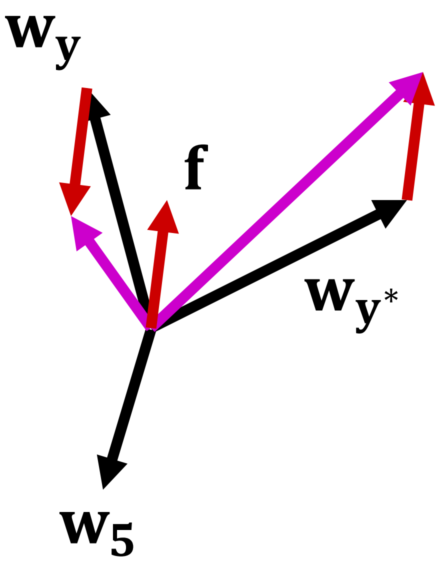
- Start with all weights = 0
- Pick up training examples one by one
- Predict with current weights

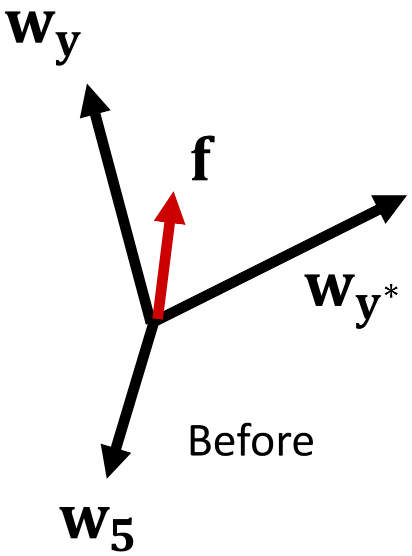
$$y = \arg\max_y \mathbf{w_y} \cdot \mathbf{f}(x)$$

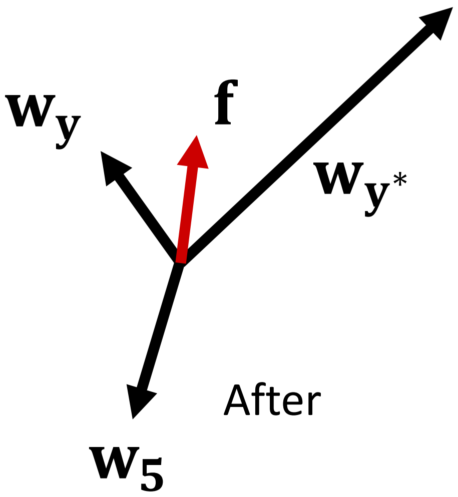
- If correct, no change!
- If wrong: subtract $\mathbf{f}$ from wrong answer, add $\mathbf{f}$ to correct answer, all other weight vectors unchanged.

Wrong: $$\mathbf{w_y} = \mathbf{w_y} - \mathbf{f}(x)$$

Correct: $$\mathbf{w_{y^*}} = \mathbf{w_{y^*}} + \mathbf{f}(x)$$







Perceptron Demo

“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
...
```

Demo: Multiclass Perceptron

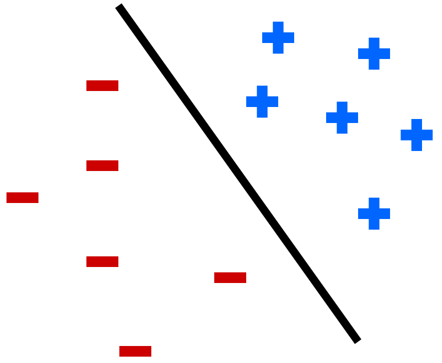
- See webcast.
- Or try it yourself by downloading https://drive.google.com/file/d/0B3UAxDgZnbC_X1h6QnhScVlVVEk/view
  - (won’t display in browser, you have to actually download it and open it locally)

Properties of Perceptrons

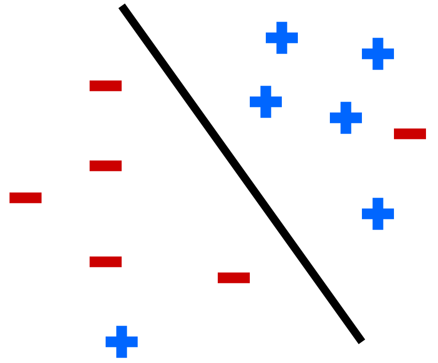
- Separability: true if there exists some set of parameters that gets the training set perfectly correct (i.e. there exists a separating hyperplane)
- Convergence: if the training data is separable, perceptron will eventually converge (binary case)
- Mistake Bound: can mathematically compute the maximum number of mistakes (binary case) as a function of the *margin* or degree of separability [well beyond scope of this course]

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

Separable

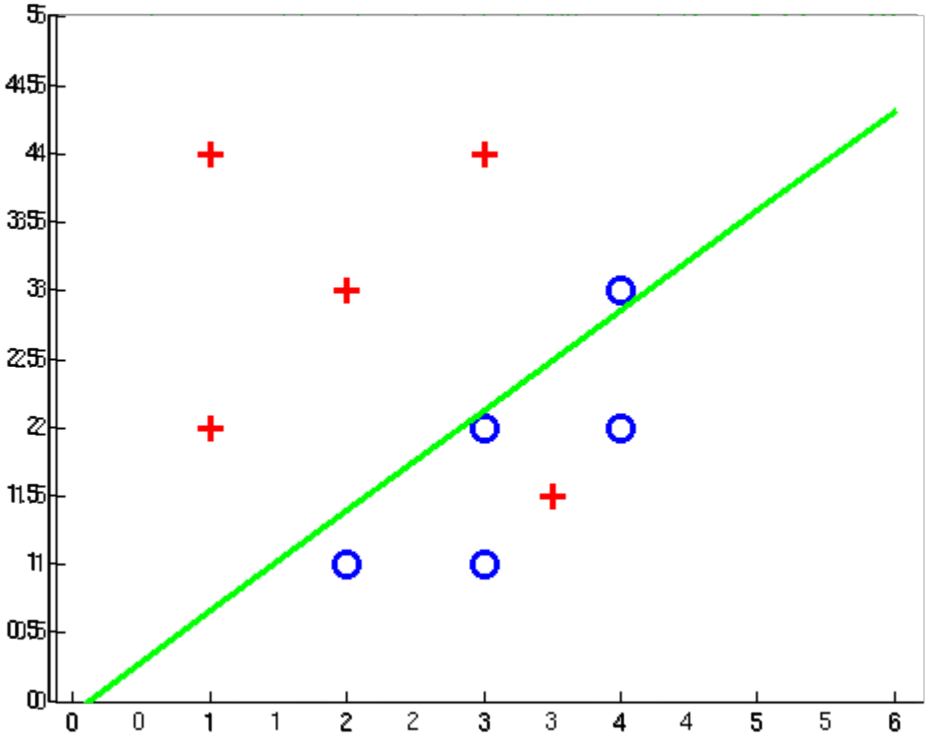


Non-Separable

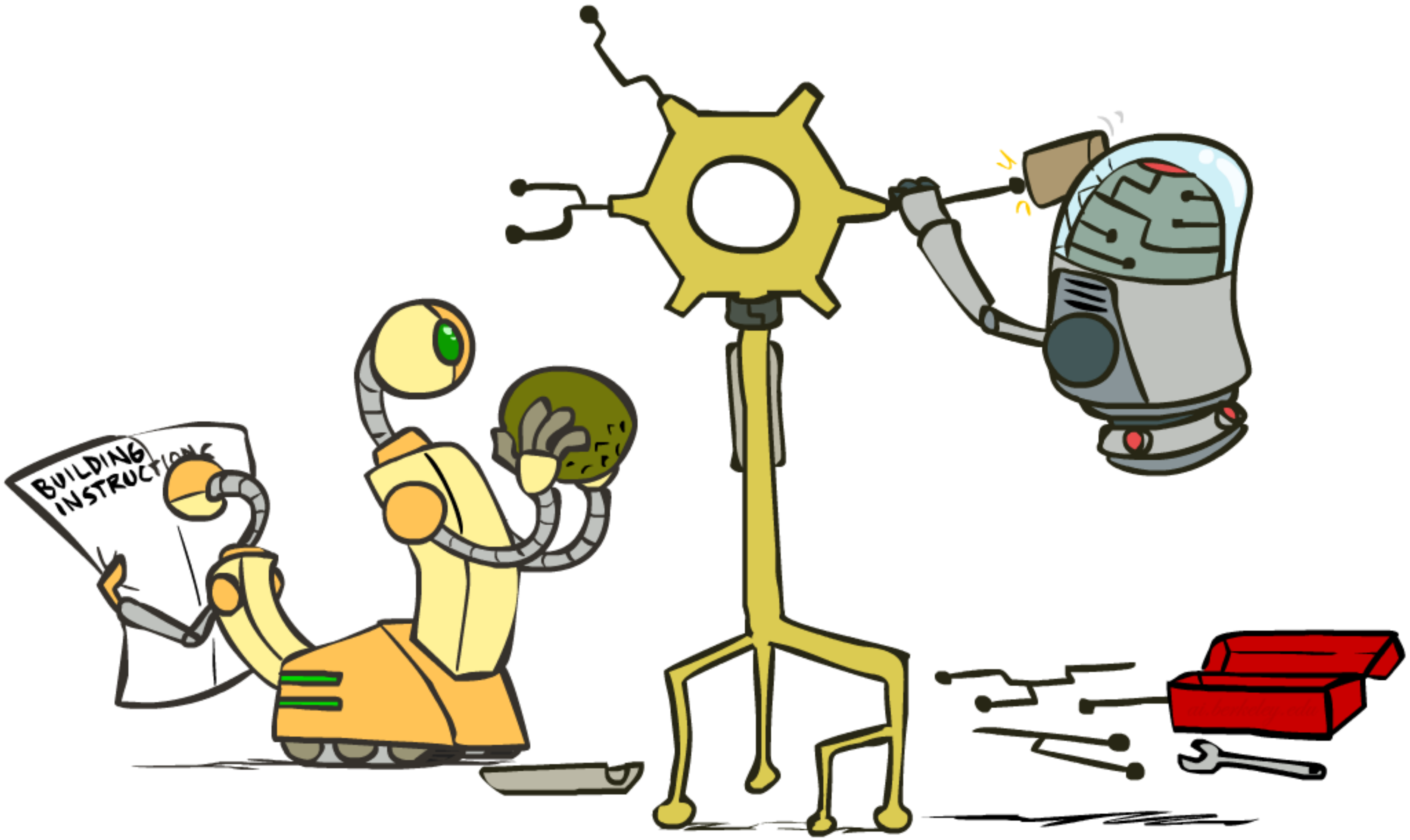


Examples: Perceptron

- Non-Separable Case

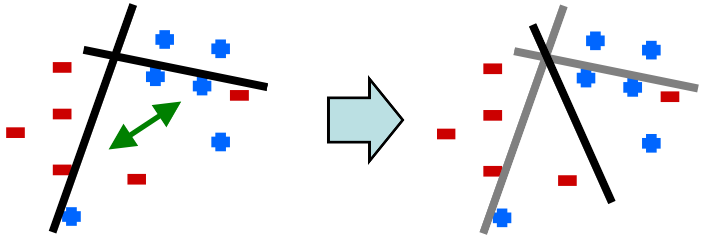


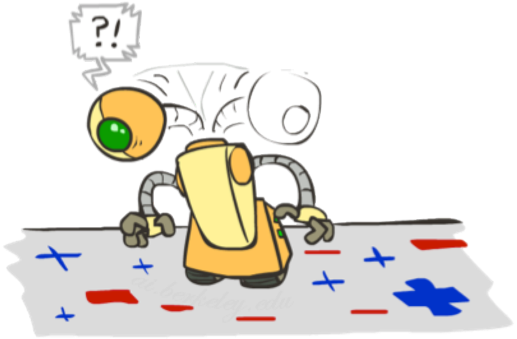
Improving the Perceptron



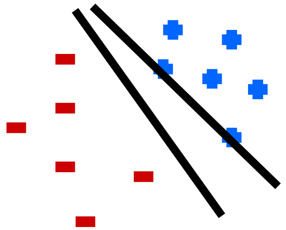
Problems with the Perceptron

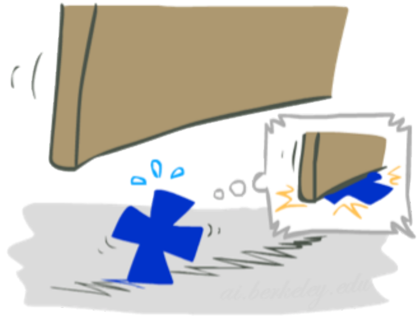
- Noise: if the data isn't separable, weights might thrash
  - Averaging weight vectors over time can help (averaged perceptron)



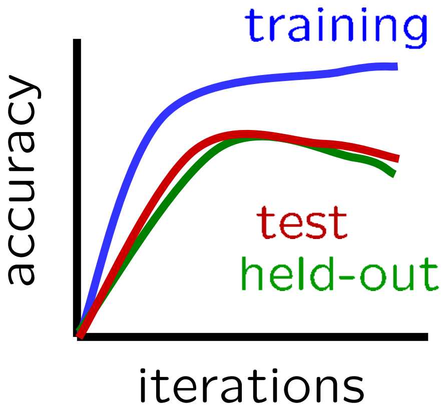


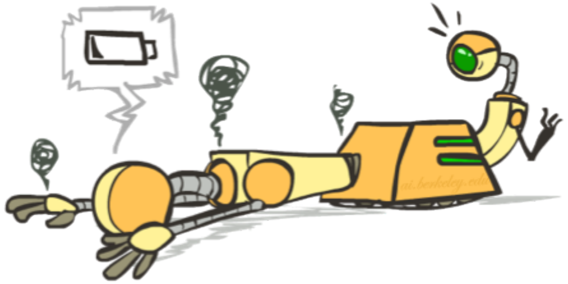
- Mediocre generalization: finds a "barely" separating solution





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

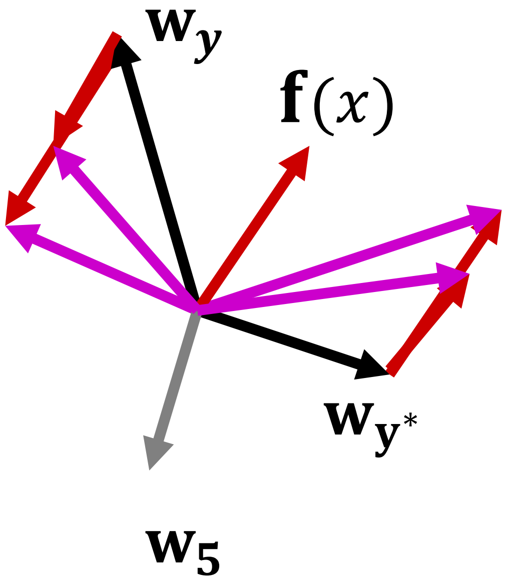




Fixing the Perceptron

- Idea: adjust the weight update to mitigate these effects

- Issue: when we saw x, predicted y, but answer was y^*, we did the "lazy" thing, causing huge swings:
  - Added $\mathbf{f}(x)$ to $\mathbf{w}_{\mathbf{y}^*}$
  - Subtracted $\mathbf{f}(x)$ from $\mathbf{w}_{\mathbf{y}}$



- MIRA*: choose an update size that fixes the current mistake…
- … but, minimizes the change to $\mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_L \end{bmatrix}$

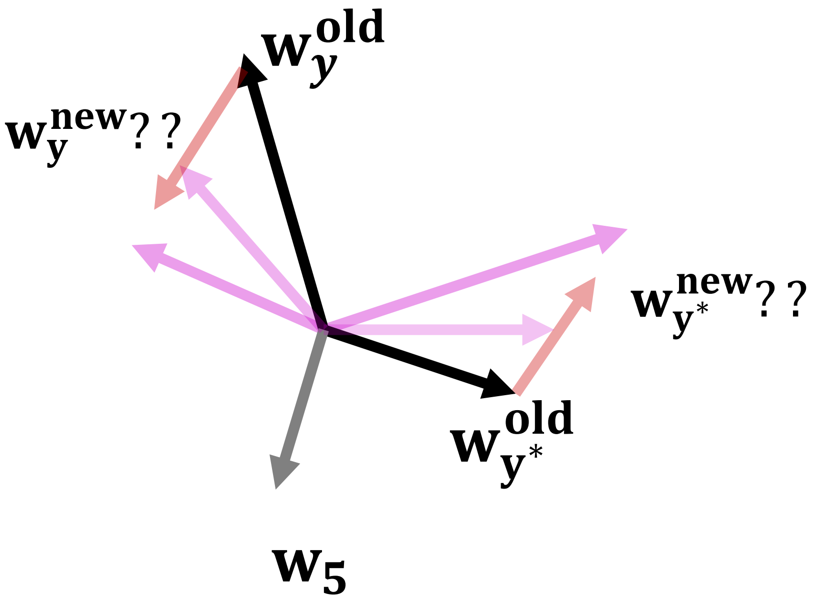
- To do this:
  - Still going to change only $\mathbf{w}_{\mathbf{y}}$ and $\mathbf{w}_{\mathbf{y}^*}$ i.e. bystander vectors like $\mathbf{w}_5$ are left untouched.
  - Still going to change in the same direction, just by less!

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 $\mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_L \end{bmatrix}$

- Handy idea: Define τ to be how much we want to use $\mathbf{f}(x)$, i.e.
  - $\mathbf{w}_{\mathbf{y}}^{\mathbf{new}} = \mathbf{w}_{\boldsymbol{y}}^{\mathbf{old}} - \tau\mathbf{f}(x)$
  - $\mathbf{w}_{\mathbf{y}^*}^{\mathbf{new}} = \mathbf{w}_{\boldsymbol{y}^*}^{\mathbf{old}} + \tau\mathbf{f}(x)$



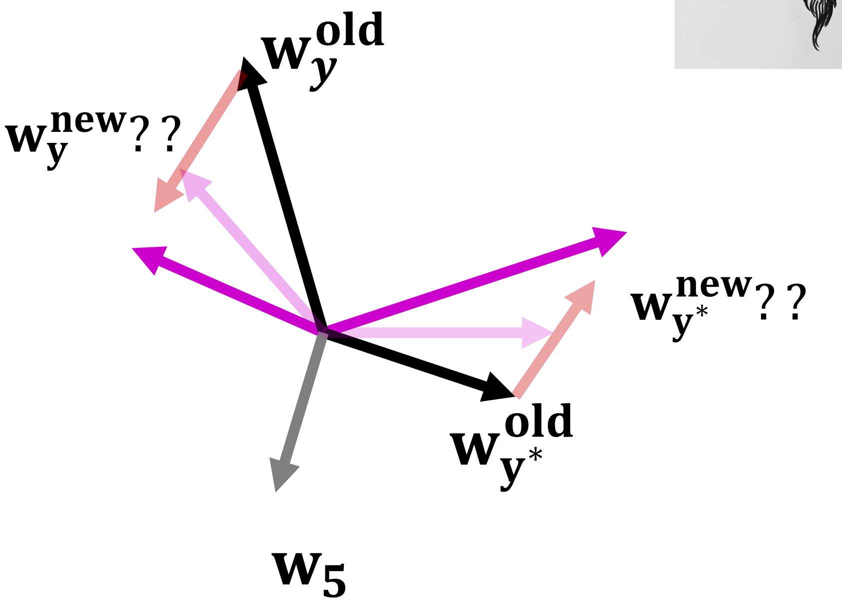
Sanity Check Question

- Idea: adjust the weight update to mitigate these effects

- MIRA*: choose an update size that fixes the current mistake…
- … but, minimizes the change to $\mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_L \end{bmatrix}$

- Handy idea: Define τ to be how much we want to use $\mathbf{f}(x)$, i.e.
  - $\mathbf{w}_{\mathbf{y}}^{\mathbf{new}} = \mathbf{w}_{\boldsymbol{y}}^{\mathbf{old}} - \tau\mathbf{f}(x)$
  - $\mathbf{w}_{\mathbf{y}^*}^{\mathbf{new}} = \mathbf{w}_{\boldsymbol{y}^*}^{\mathbf{old}} + \tau\mathbf{f}(x)$

- What happens if we set $\tau = 0$? $\tau = 1$?

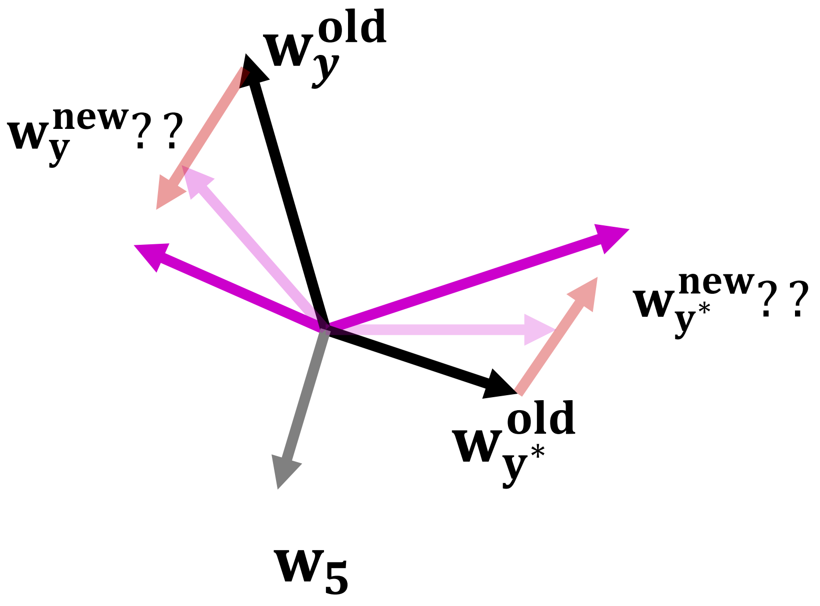


Sanity Check Question

- Idea: adjust the weight update to mitigate these effects

- MIRA*: choose an update size that fixes the current mistake…
- … but, minimizes the change to $\mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_L \end{bmatrix}$

- Handy idea: Define τ to be how much we want to use $\mathbf{f}(x)$, i.e.
  - $\mathbf{w}_{\mathbf{y}}^{\mathbf{new}} = \mathbf{w}_{\boldsymbol{y}}^{\mathbf{old}} - \tau \mathbf{f}(x)$
  - $\mathbf{w}_{\mathbf{y}^*}^{\mathbf{new}} = \mathbf{w}_{\boldsymbol{y}^*}^{\mathbf{old}} + \tau \mathbf{f}(x)$



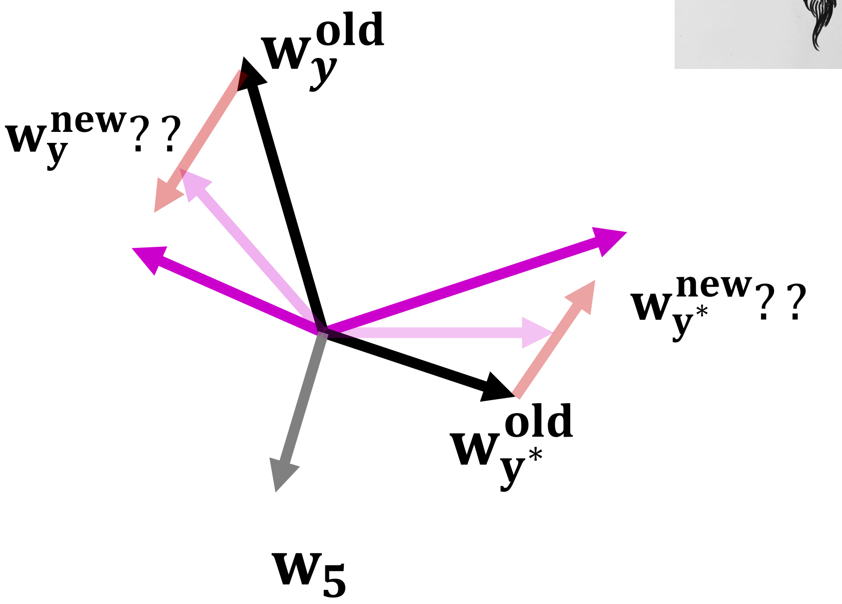
- What happens if we set $\tau = 0$? $\tau = 1$?
  - $\tau = 0$: Our algorithm never learns.
  - $\tau = 1$: We have the original perceptron algorithm, which shoves too hard.

Measuring Fixed Mistake

- Idea: adjust the weight update to mitigate these effects

- MIRA*: choose an update size that fixes the current mistake…
- … but, minimizes the change to $\mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_L \end{bmatrix}$

- Handy idea: Define τ to be how much we want to use $\mathbf{f}(x)$, i.e.
  - $\mathbf{w}_{\mathbf{y}}^{\mathbf{new}} = \mathbf{w}_{\boldsymbol{y}}^{\mathbf{old}} - \tau \mathbf{f}(x)$
  - $\mathbf{w}_{\mathbf{y}^*}^{\mathbf{new}} = \mathbf{w}_{\boldsymbol{y}^*}^{\mathbf{old}} + \tau \mathbf{f}(x)$



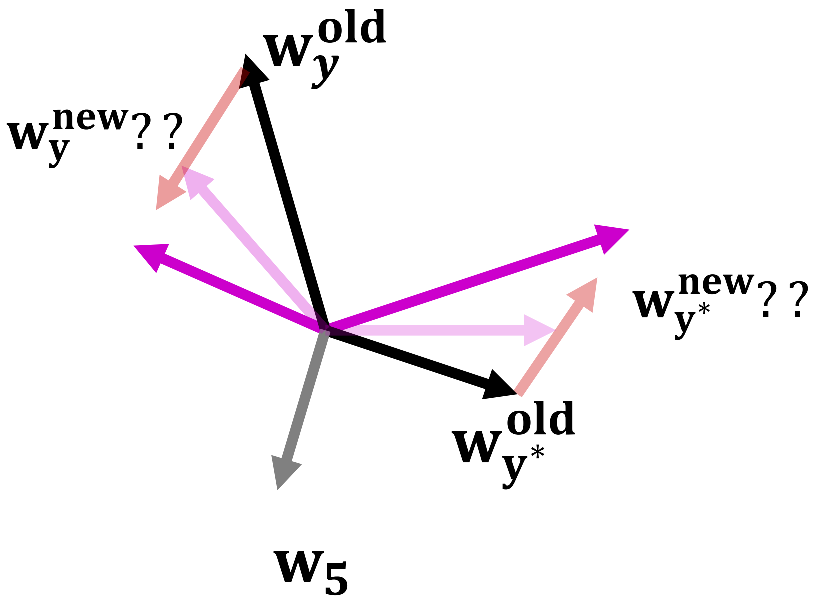
- Give an expression that indicates we've fixed the current mistake.

Measuring Fixed Mistake

- Idea: adjust the weight update to mitigate these effects

- MIRA*: choose an update size that fixes the current mistake…
- … but, minimizes the change to $\mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_L \end{bmatrix}$

- Handy idea: Define τ to be how much we want to use $\mathbf{f}(x)$, i.e.
  - $\mathbf{w}_{\mathbf{y}}^{\mathbf{new}} = \mathbf{w}_{\boldsymbol{y}}^{\mathbf{old}} - \tau \mathbf{f}(x)$
  - $\mathbf{w}_{\mathbf{y}^*}^{\mathbf{new}} = \mathbf{w}_{\boldsymbol{y}^*}^{\mathbf{old}} + \tau \mathbf{f}(x)$



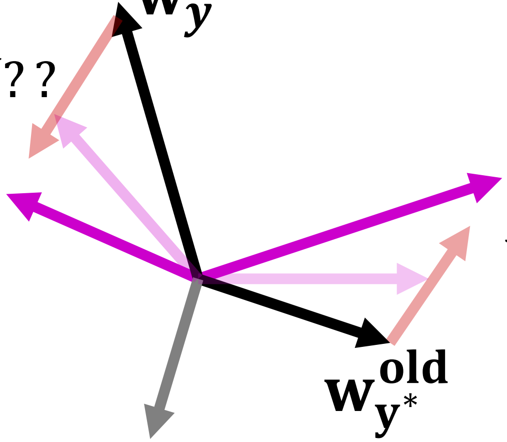
- Give an expression that indicates we've fixed the current mistake.

$$\mathbf{w}_{\mathbf{y}^*}^{\mathbf{new}} \cdot \mathbf{f}(x) \geq \mathbf{w}_{\boldsymbol{y}}^{\mathbf{new}} \cdot \mathbf{f}(x)$$

Measuring Change in W

- Idea: adjust the weight update to mitigate these effects

- MIRA*: choose an update size that fixes the current mistake…
- … but, minimizes the change to $\mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_L \end{bmatrix}$



- Handy idea: Define τ to be how much we want to use $\mathbf{f}(x)$, i.e.
  - $\mathbf{w}_\mathbf{y}^{\text{new}} = \mathbf{w}_y^{\text{old}} - \tau\mathbf{f}(x)$
  - $\mathbf{w}_{\mathbf{y}^*}^{\text{new}} = \mathbf{w}_{y^*}^{\text{old}} + \tau\mathbf{f}(x)$

- Give an expression that indicates we've fixed the current mistake. $\mathbf{w}_{\mathbf{y}^*}^{\text{new}} \cdot \mathbf{f}(x) \geq \mathbf{w}_y^{\text{new}} \cdot \mathbf{f}(x)$

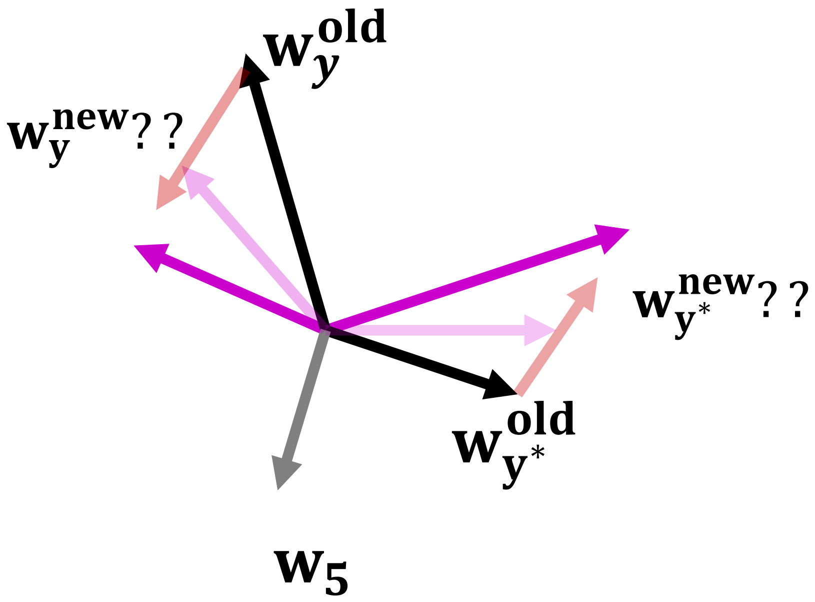
- Give an expression that computes the change in $\mathbf{W}$ (hard question with obvious answer!).

Measuring Change in W

- Idea: adjust the weight update to mitigate these effects

- MIRA*: choose an update size that fixes the current mistake…
- … but, minimizes the change to $\mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_L \end{bmatrix}$

- Handy idea: Define τ to be how much we want to use $\mathbf{f}(x)$, i.e.
  - $\mathbf{w}_{\mathbf{y}}^{\mathbf{new}} = \mathbf{w}_{\boldsymbol{y}}^{\mathbf{old}} - \tau\mathbf{f}(x)$
  - $\mathbf{w}_{\mathbf{y}^*}^{\mathbf{new}} = \mathbf{w}_{\boldsymbol{y}^*}^{\mathbf{old}} + \tau\mathbf{f}(x)$



- Give an expression that indicates we've fixed the current mistake. $\mathbf{w}_{\mathbf{y}^*}^{\mathbf{new}} \cdot \mathbf{f}(x) \geq \mathbf{w}_{\boldsymbol{y}}^{\mathbf{new}} \cdot \mathbf{f}(x)$

- Give an expression that computes the change in **W**. $\sum_{i=0}^{L} \left\| \mathbf{w}_{\mathbf{i}}^{\mathbf{new}} - \mathbf{w}_{\mathbf{i}}^{\mathbf{old}} \right\|$
  - There are many other reasonable answers!

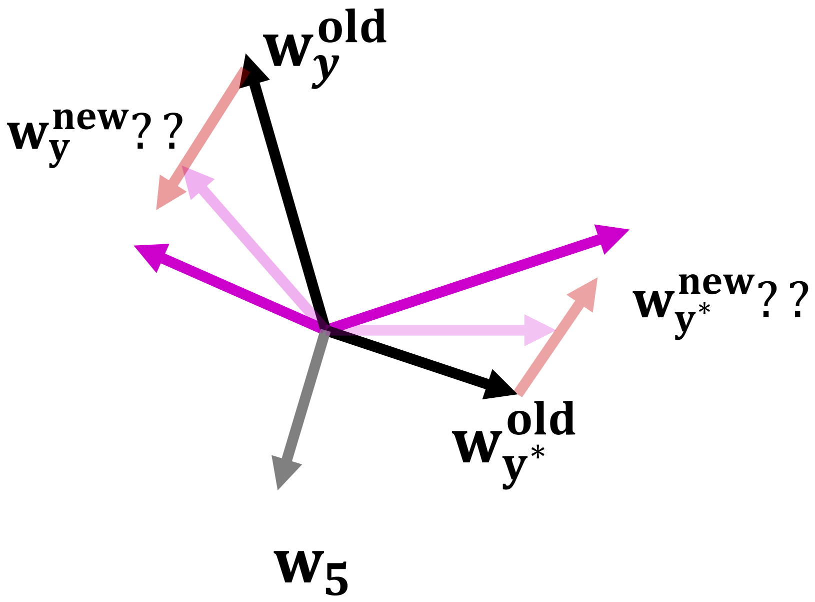
MIRA

- 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

$$\min_{\mathbf{W}} \sum_{i=0}^{L} \left\| \mathbf{w}_{\mathbf{i}}^{\mathbf{new}} - \mathbf{w}_{\mathbf{i}}^{\mathbf{old}} \right\|$$

$$s.t. \quad \mathbf{w}_{\mathbf{y}^*}^{\mathbf{new}} \cdot \mathbf{f}(\mathrm{x}) \geq \mathbf{w}_{\boldsymbol{y}}^{\mathbf{new}} \cdot \mathbf{f}(x)$$



We saw x, predicted y, but y^* was the right answer, so we adjust weights as:

$$\mathbf{w}_{\mathbf{y}}^{\mathbf{new}} = \mathbf{w}_{\boldsymbol{y}}^{\mathbf{old}} - \tau\mathbf{f}(x)$$
$$\mathbf{w}_{\mathbf{y}^*}^{\mathbf{new}} = \mathbf{w}_{\boldsymbol{y}^*}^{\mathbf{old}} + \tau\mathbf{f}(x)$$

Slight Reframing of MIRA

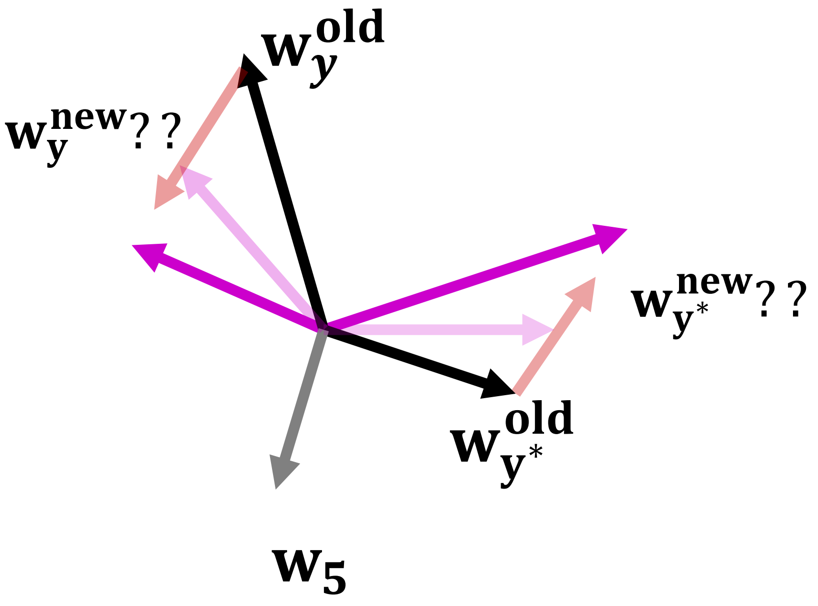
- Idea: adjust the weight update to mitigate these effects

- MIRA*: choose an update size that fixes the current mistake…
- … but, minimizes the change to w

$$\min_{\mathbf{w}} \frac{1}{2} \sum_{i=0}^{L} \left\| \mathbf{w}_{\mathbf{i}}^{\mathbf{new}} - \mathbf{w}_{\mathbf{i}}^{\mathbf{old}} \right\|^2$$

$$s.t. \quad \mathbf{w}_{\mathbf{y}^*}^{\mathbf{new}} \cdot \mathbf{f}(\mathrm{x}) \geq \mathbf{w}_{y}^{\mathbf{new}} \cdot \mathbf{f}(x) + 1$$

- This version of MIRA does a couple of slightly different things:
  - Insists that we correct by an arbitrary margin of 1.
  - Minimizes square of norm (doesn't change algorithm, but...)



We saw x, predicted y, but y^* was the right answer, so we adjust weights as:

$$\mathbf{w}_{\mathbf{y}}^{\mathbf{new}} = \mathbf{w}_{y}^{\mathbf{old}} - \tau \mathbf{f}(x)$$
$$\mathbf{w}_{\mathbf{y}^*}^{\mathbf{new}} = \mathbf{w}_{y^*}^{\mathbf{old}} + \tau \mathbf{f}(x)$$

Minimum Correcting Update

$$\min_{\mathbf{w}} \frac{1}{2} \sum_{i=0}^{L} \left\| \mathbf{w}_{\mathbf{i}}^{\mathbf{new}} - \mathbf{w}_{\mathbf{i}}^{\mathbf{old}} \right\|^2$$

$$s.t. \quad \mathbf{w}_{\mathbf{y}^*}^{\mathbf{new}} \cdot \mathbf{f}(\mathrm{x}) \geq \mathbf{w}_{\boldsymbol{y}}^{\mathbf{new}} \cdot \mathbf{f}(x) + 1$$

(equivalent to, use substitution)

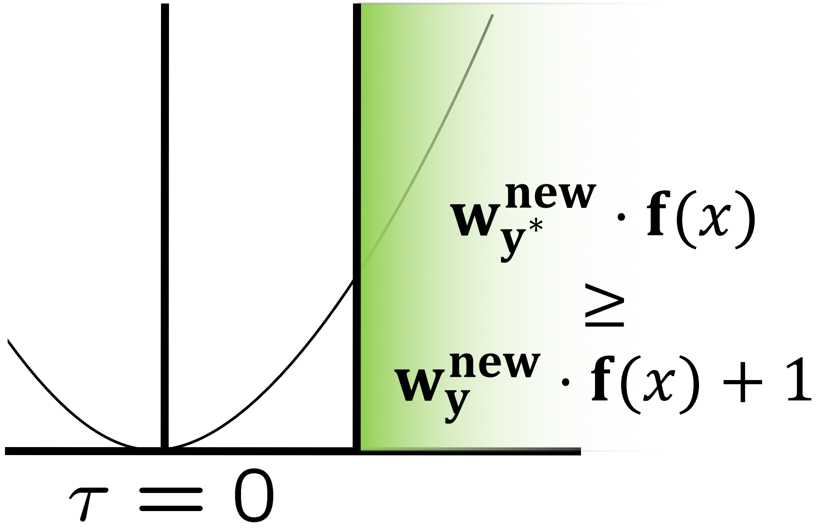
$$\min_{\tau} \sum_{i=0}^{L} \|\tau \mathbf{f}(x)\|^2$$

$$s.t. \quad \mathbf{w}_{\mathbf{y}^*}^{\mathbf{new}} \cdot \mathbf{f}(\mathrm{x}) \geq \mathbf{w}_{\boldsymbol{y}}^{\mathbf{new}} \cdot \mathbf{f}(x) + 1$$

(substitute, solve for when condition is =)

$$\left(\mathbf{w}_{\mathbf{y}^*}^{\mathbf{old}} + \tau \mathbf{f}(\mathrm{x})\right) \cdot \mathbf{f}(\mathrm{x}) = \left(\mathbf{w}_{\boldsymbol{y}}^{\mathbf{old}} - \tau \mathbf{f}(\mathrm{x})\right) \cdot \mathbf{f}(x) + 1$$

$$\boxed{\begin{aligned} \mathbf{w}_{\mathbf{y}}^{\mathbf{new}} &= \mathbf{w}_{\boldsymbol{y}}^{\mathbf{old}} - \tau \mathbf{f}(x) \\ \mathbf{w}_{\mathbf{y}^*}^{\mathbf{new}} &= \mathbf{w}_{\boldsymbol{y}^*}^{\mathbf{old}} + \tau \mathbf{f}(x) \end{aligned}}$$



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

Minimum Correcting Update

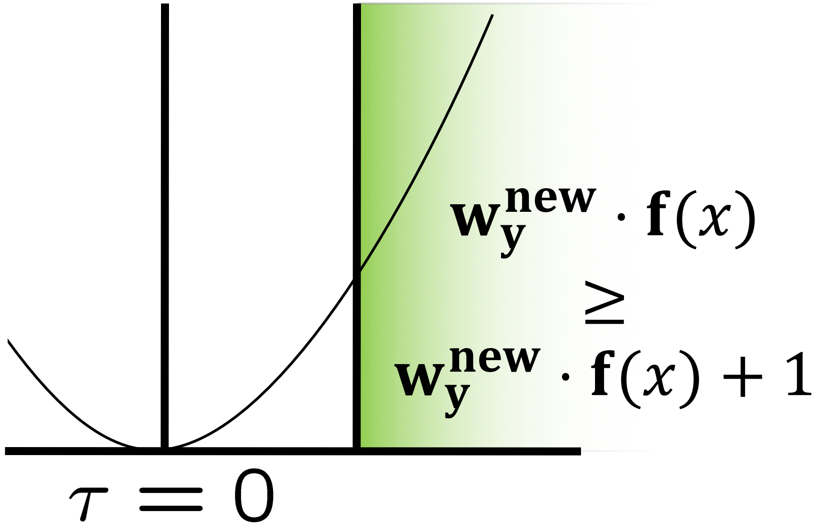
$$\left(\mathbf{w}_{\mathbf{y}^*}^{\mathbf{old}} + \tau\mathbf{f}(\mathrm{x})\right)\cdot\mathbf{f}(\mathrm{x}) = \left(\mathbf{w}_{y}^{\mathbf{old}} - \tau\mathbf{f}(\mathrm{x})\right)\cdot\mathbf{f}(x) + 1$$

$$\mathbf{w}_{\mathbf{y}}^{\mathbf{new}} = \mathbf{w}_{y}^{\mathbf{old}} - \tau\mathbf{f}(x)$$
$$\mathbf{w}_{\mathbf{y}^*}^{\mathbf{new}} = \mathbf{w}_{\mathbf{y}^*}^{\mathbf{old}} + \tau\mathbf{f}(x)$$

(solve for τ, just algebra)

$$\tau = \frac{\left(\mathbf{w}_{y}^{\mathbf{old}} - \mathbf{w}_{\mathbf{y}^*}^{\mathbf{old}}\right)\cdot\mathbf{f}(x) + 1}{2\mathbf{f}(\mathrm{x})\cdot\mathbf{f}(x)}$$

- Or in English:
  - Shove the weight vector for the right prediction in the direction of the input-under-test, using τ to say how hard
  - Shove the weight vector for the wrong prediction away from the input-under-test, using τ to say how hard
  - The equation above lets us calculate a better τ than the one we used before, which was 1.



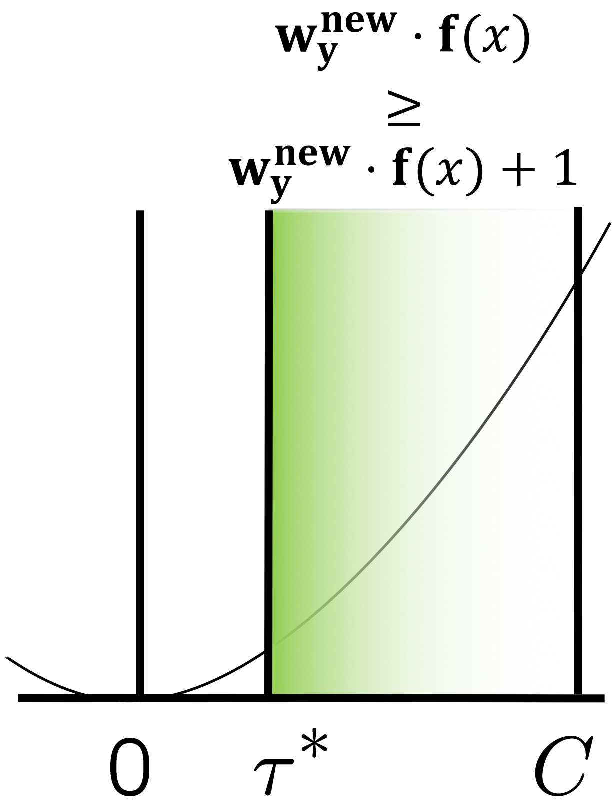
min not τ=0, or would not have made an error, so min will be where equality holds

Maximum Step Size

- In practice, don't want 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

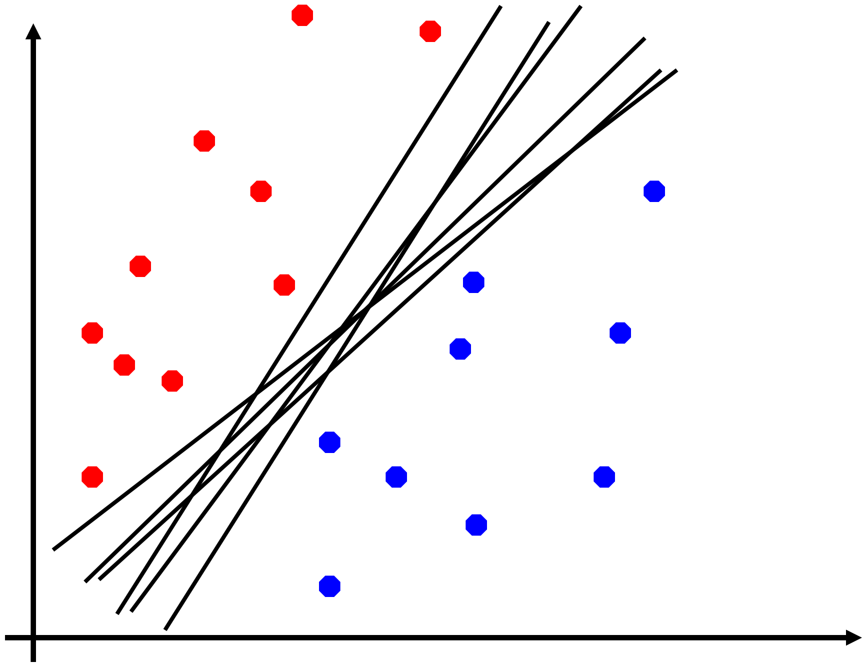
$$\tau = \min\left(\frac{\left(\mathbf{w}_y^{old} - \mathbf{w}_{\mathbf{y}^*}^{old}\right)\cdot \mathbf{f}(x) + 1}{2\mathbf{f}(\mathrm{x})\cdot \mathbf{f}(x)}, C\right)$$

  - Corresponds to an optimization that assumes non-separable data
  - Usually converges faster than perceptron
  - Usually better, especially on noisy data (i.e. data with mislabeled values)



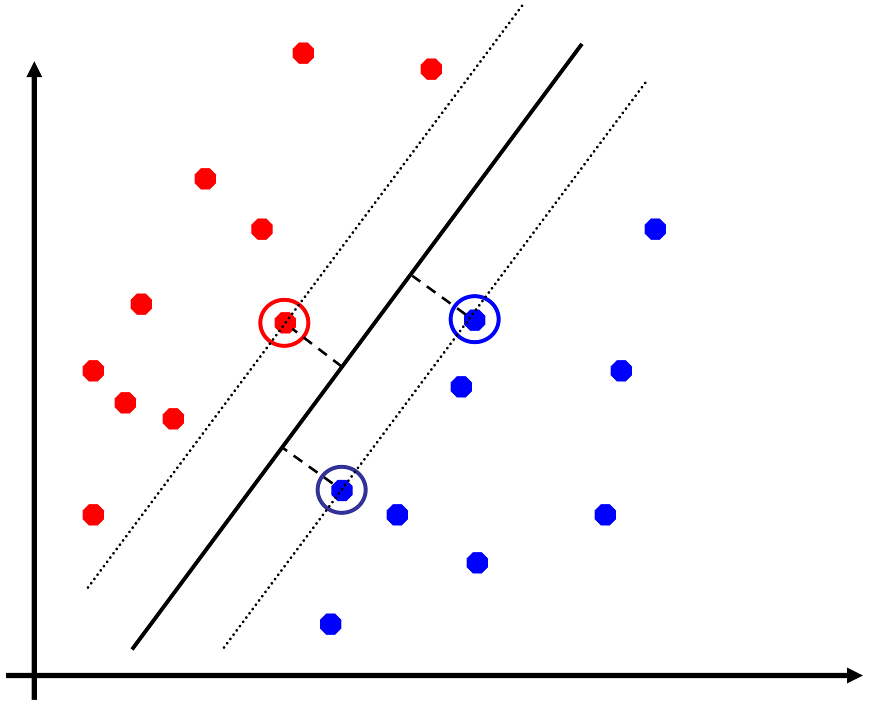
Linear Separators

- 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
  - Turns out that this is equivalent to minimizing the magnitudes of the weight vectors (!!)
- Basically, SVMs are MIRA where you optimize over all examples at once



MIRA, repeat until satisfied:

$$\min_{\mathbf{W}} \frac{1}{2} \left\| \mathbf{W}^{\mathbf{new}} - \mathbf{W}^{\mathbf{old}} \right\|^2$$

$$s.t. \quad \mathbf{w}_{\mathbf{y}^*}^{\mathbf{new}} \cdot \mathbf{f}(\mathrm{x}) \geq \mathbf{w}_{\boldsymbol{y}}^{\mathbf{new}} \cdot \mathbf{f}(x) + 1$$

SVM, find the W such that: (via link)

$$\min_{\mathbf{W}} \frac{1}{2} \|\boldsymbol{W}\|^2$$

$$s.t. \ \forall i, y: \boldsymbol{w}_{\boldsymbol{y}^*} \cdot \mathbf{f}(\mathrm{x_i}) \geq \boldsymbol{w}_{\boldsymbol{y}} \cdot \mathbf{f}(x_i) + 1$$

Classification: Comparison

- Naïve Bayes
  - Builds a model training data
  - Gives prediction probabilities
  - Strong assumptions about feature independence
  - One pass through data (counting)

- Perceptrons / MIRA:
  - Makes less assumptions about data
  - Mistake-driven learning
  - Multiple passes through data (prediction)
  - Often more accurate

Note to Future Prospective 189 Students

- We took it very easy on the linear algebra today!
- If you plan on taking 189, make sure you're feeling solid on linear algebra.
  - Refresh your math 54 (or equivalent knowledge).
- Not a bad idea to take Math 110 first, but not totally necessary.

MIRA, repeat until satisfied:

$$\min_{\mathbf{W}} \frac{1}{2} \left\| \mathbf{W}^{\mathbf{new}} - \mathbf{W}^{\mathbf{old}} \right\|^2$$

$$s.t. \quad \mathbf{w}_{\mathbf{y}^*}^{\mathbf{new}} \cdot \mathbf{f}(\mathrm{x}) \geq \mathbf{w}_{\boldsymbol{y}}^{\mathbf{new}} \cdot \mathbf{f}(x) + 1$$

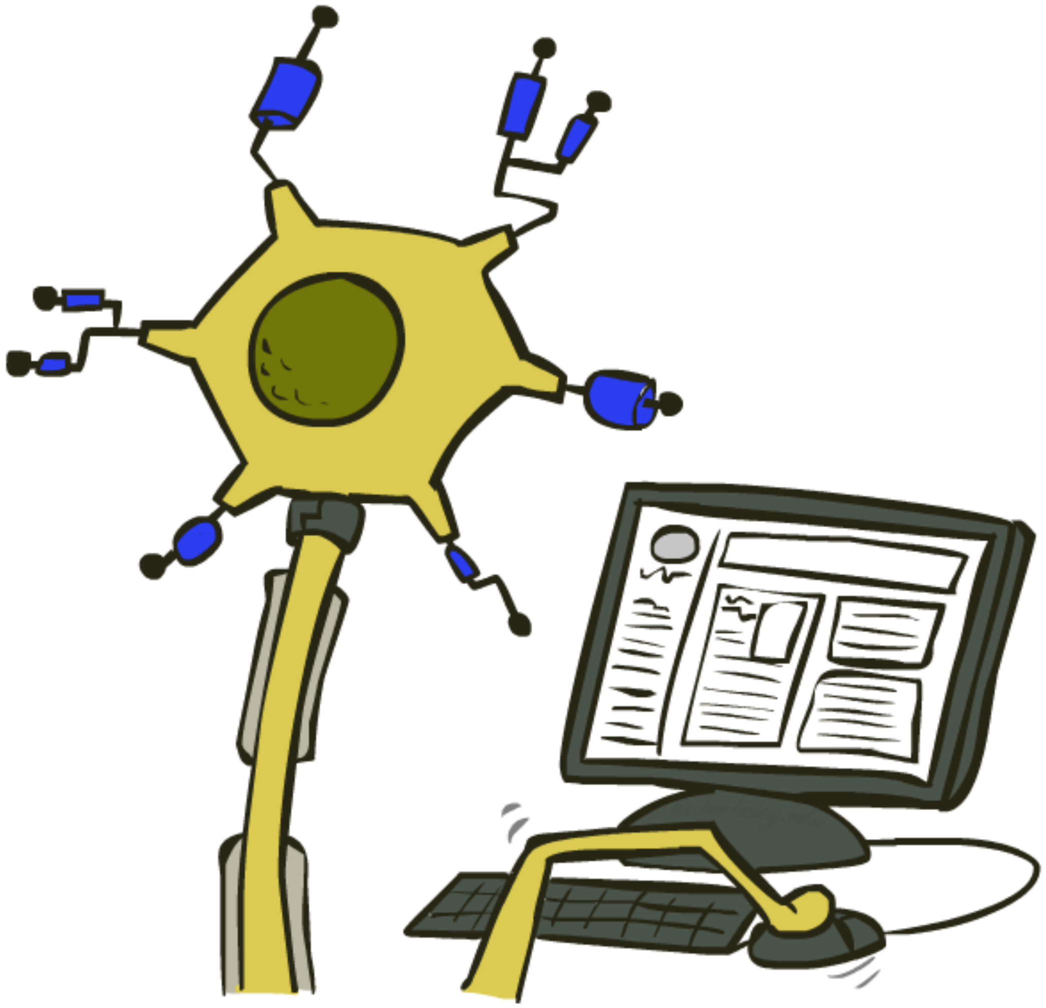
SVM, find the W such that:

$$\min_{\mathbf{W}} \frac{1}{2} \|\boldsymbol{W}\|^2$$

$$s.t. \ \ \forall i, y{:}\, \boldsymbol{w}_{\boldsymbol{y}^*} \cdot \mathbf{f}(\mathrm{x}_\mathrm{i}) \geq \boldsymbol{w}_{\boldsymbol{y}} \cdot \mathbf{f}(x_i) + 1$$

Contest 2

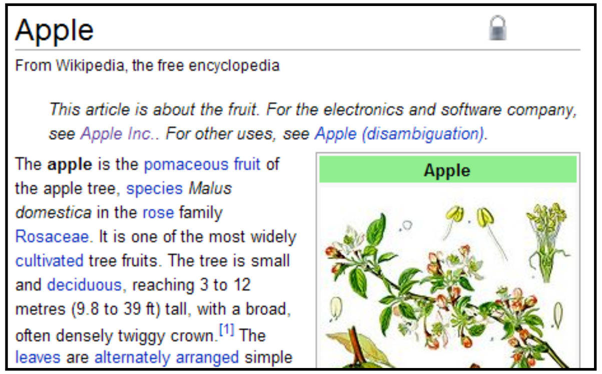
Web Search



Extension: Web Search

- Information retrieval:
  - Given information needs, produce information
  - Includes, e.g. web search, question answering, and classic IR
- Web search: not exactly classification, but rather ranking

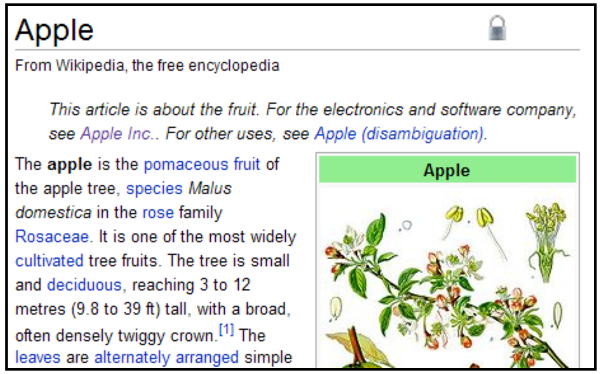
x = “Apple Computers”



Feature-Based Ranking

x = "Apple Computer"

$f(x,$



$) = [0.3\ 5\ 0\ 0\ \ldots]$

$f(x,$

$) = [0.8\ 4\ 2\ 1\ \ldots]$

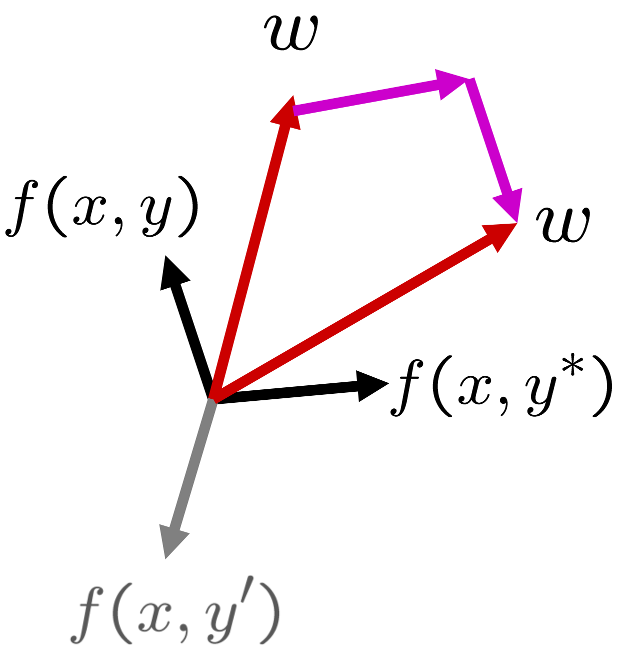
Perceptron for Ranking

- Inputs x
- Candidates y
- Many feature vectors: $f(x, y)$
- One weight vector: w
  - Prediction:

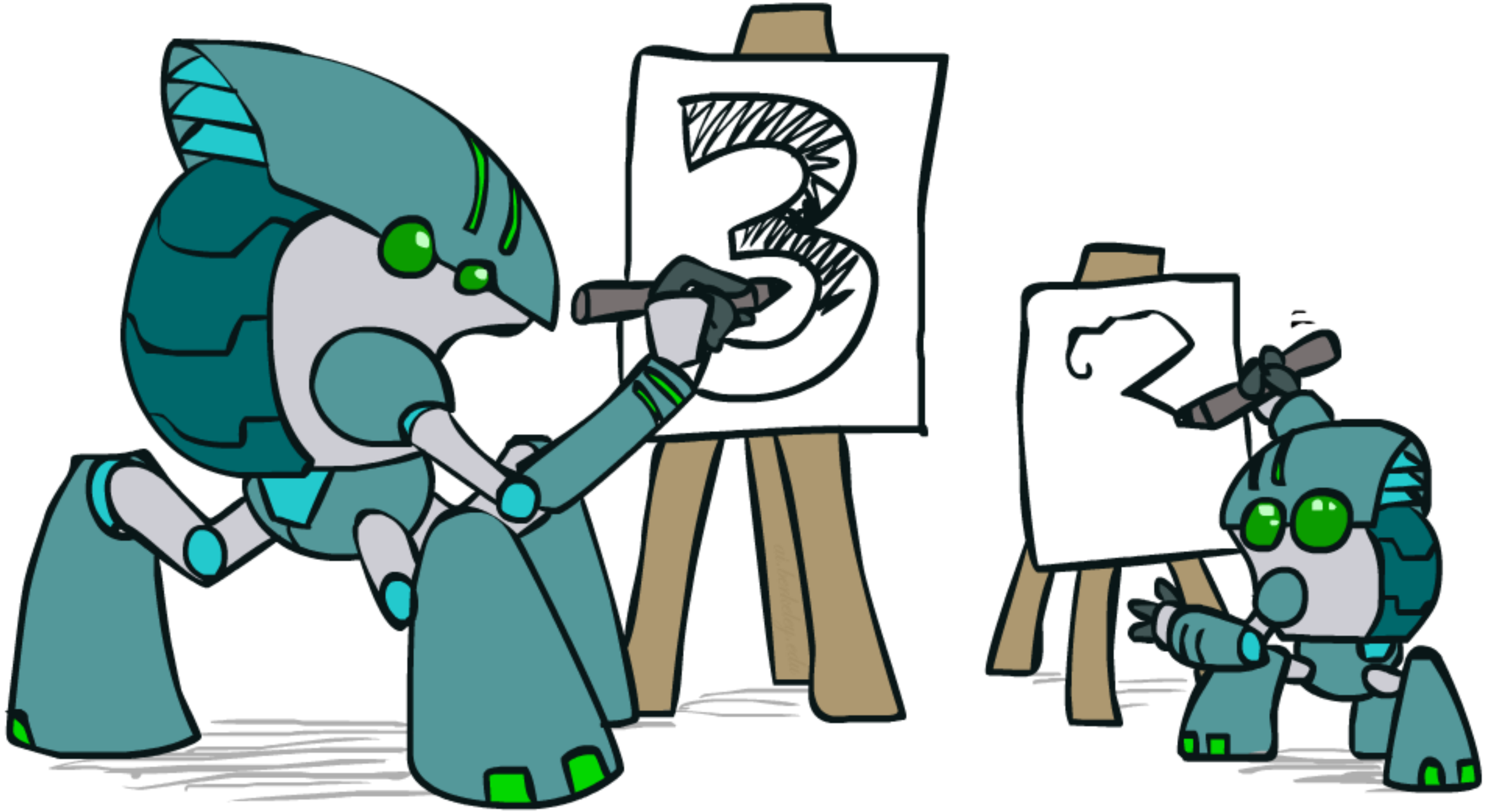
$$y = \text{arg max}_y \; w \cdot f(x, y)$$

  - Update (if wrong):

$$w = w + f(x, y^*) - f(x, y)$$

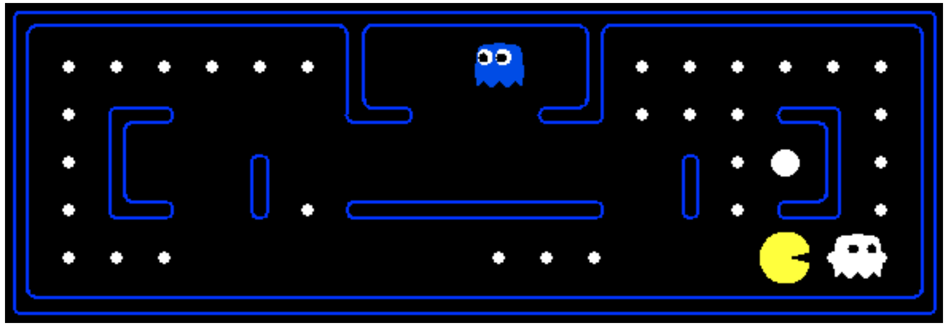


Apprenticeship (time permitting)



Pacman Apprenticeship!

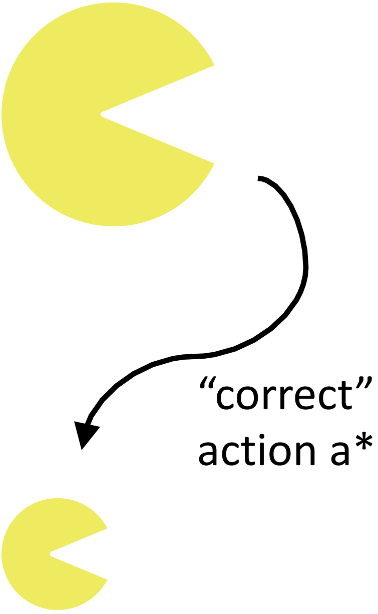
- Examples are states s



- Candidates are pairs (s,a)
- "Correct" actions: those taken by expert
- Features defined over (s,a) pairs: f(s,a)
- Score of a q-state (s,a) given by:

$$w \cdot f(s,a)$$

- How is this VERY different from reinforcement learning?



$$\forall a \neq a^*, \\ w \cdot f(a^*) > w \cdot f(a)$$

[Demo: Pacman Apprentice (L22D1,2,3)]

Video of Demo Pacman Apprentice

Next: Kernels and Clustering