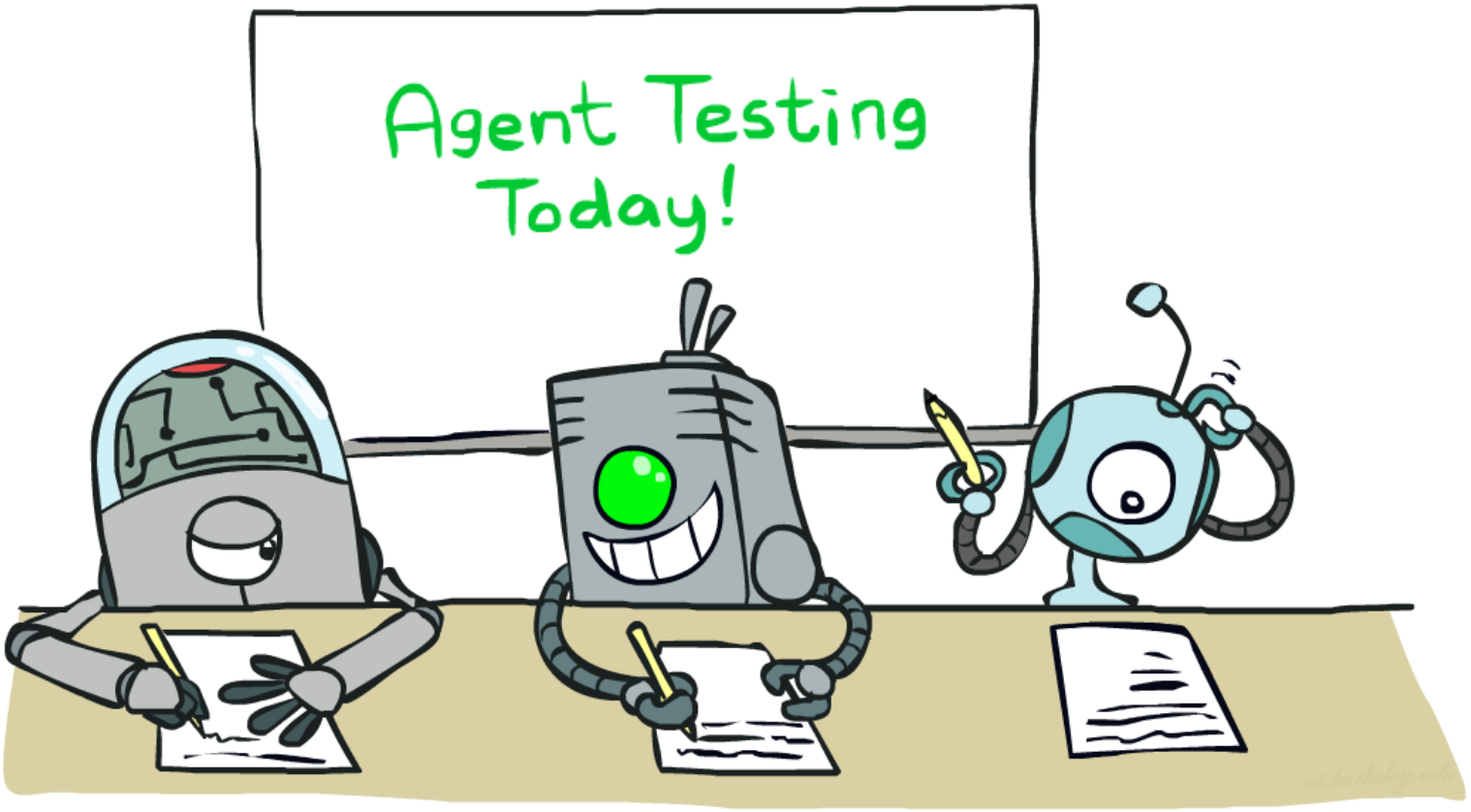
Announcements

- **Midterm 2**
  - Midterm 2 is on **Wednesday 7/29, 12pm-2pm**
  - Practice Midterm released
  - Monday and Tuesday discussions will be replaced with review sections
- Midterm 1 Grades Released
  - Regrades will open tonight, until Monday 7/27 at 11:59pm
- Written Assessment 2 & Homework 5
  - Due **Friday 7/24 at 11:59pm**
- Project 4
  - Due **Friday 7/31 at 11:59pm**

CS 188: Artificial Intelligence

Naïve Bayes

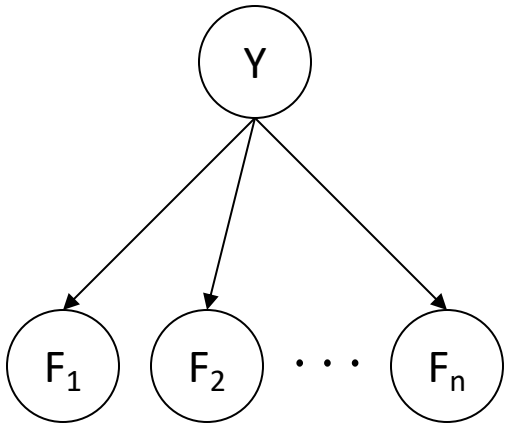


Instructor: Nikita Kitaev --- University of California, Berkeley

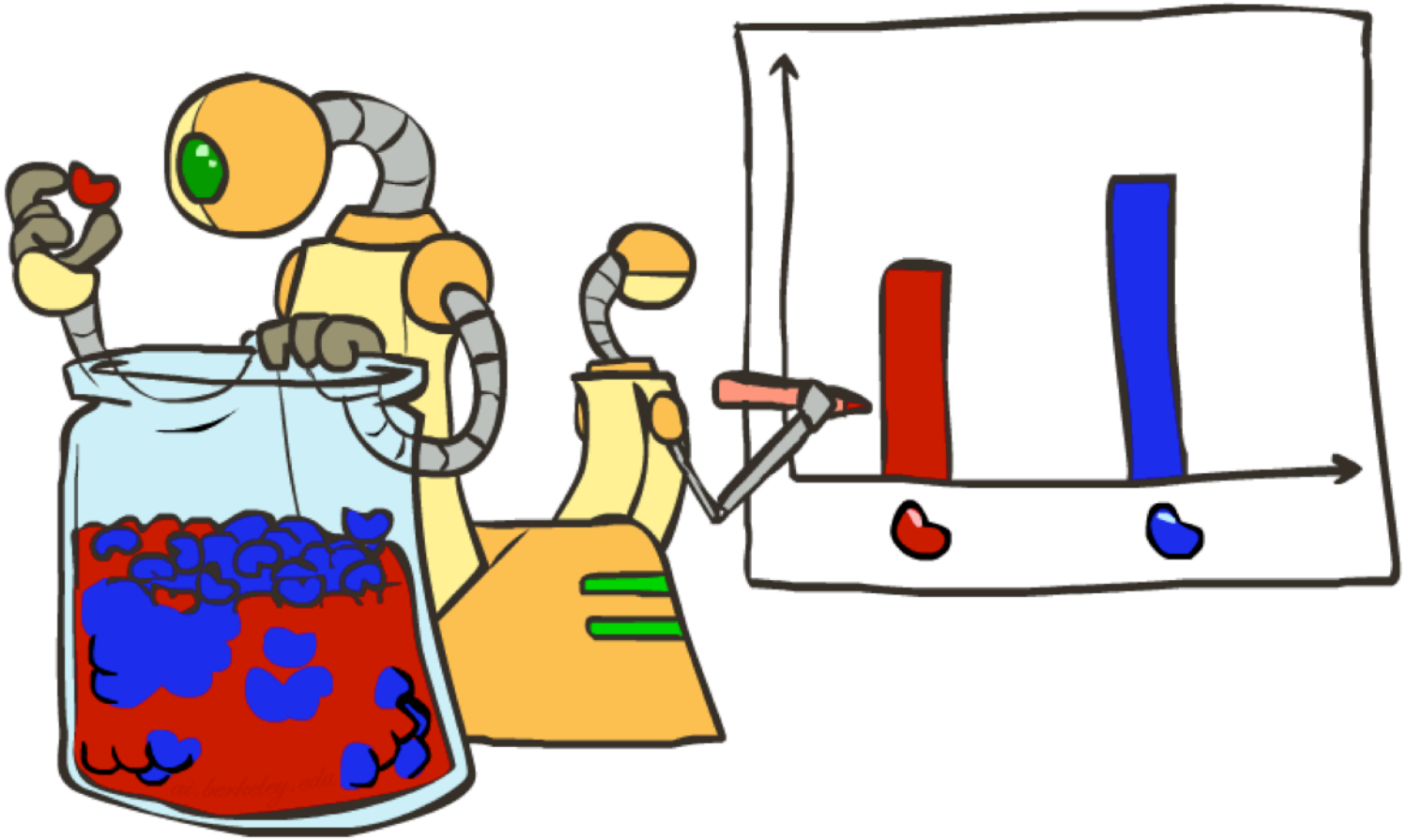
General Naïve Bayes

- A general Naive Bayes model:

$$P(\mathsf{Y}, \mathsf{F}_1 \ldots \mathsf{F}_n) = \quad P(\mathsf{Y}) \prod_i P(\mathsf{F}_i | \mathsf{Y})$$



Parameter Estimation



Maximum Likelihood?

- Relative frequencies are the maximum likelihood estimates

$$\theta_{ML} = \arg\max_{\theta} P(\mathbf{X}|\theta) = \arg\max_{\theta} \prod_i P_\theta(X_i) \Rightarrow P_{\mathsf{ML}}(x) = \frac{\mathsf{count}(x)}{\mathsf{total\ samples}}$$

- Another option is to consider the most likely parameter value given the data

$$\theta_{MAP} = \arg\max_{\theta} P(\theta|\mathbf{X}) = \arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta)/P(\mathbf{X}) = \arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta) \Rightarrow$$

????

Example: Overfitting

$P(\text{features}, C = 2)$

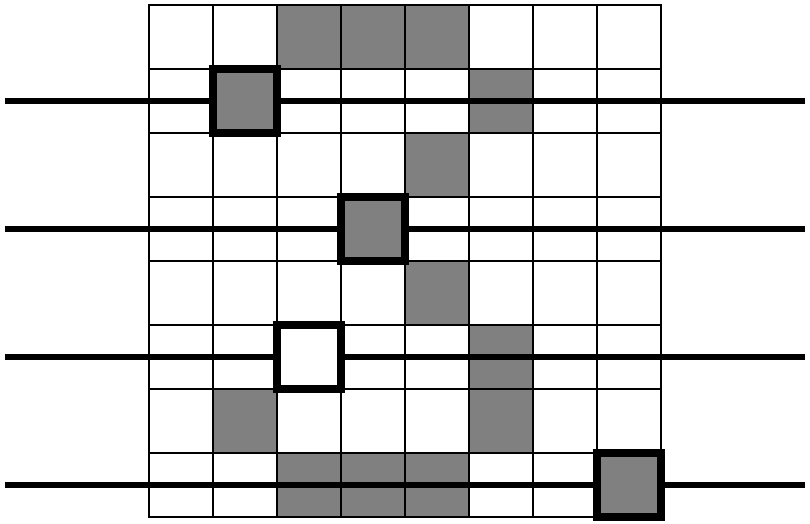
$P(C = 2) = 0.1$

$P(\text{on}|C = 2) = 0.8$

$P(\text{on}|C = 2) = 0.1$

$P(\text{off}|C = 2) = 0.1$

$P(\text{on}|C = 2) = 0.01$



$P(\text{features}, C = 3)$

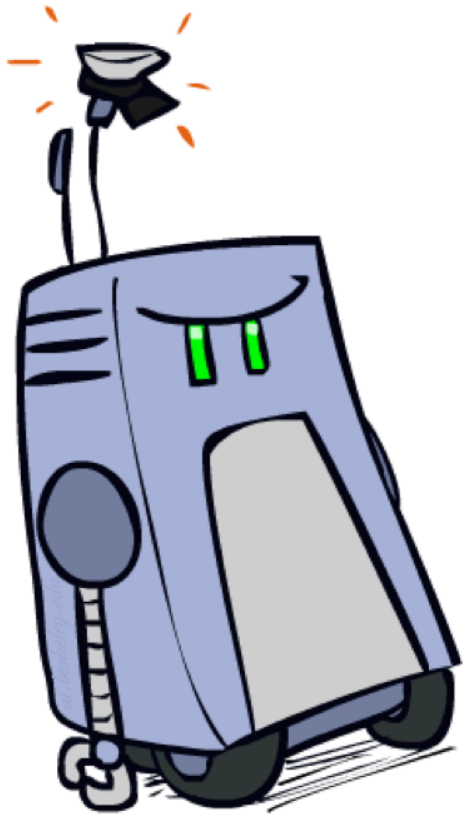
$P(C = 3) = 0.1$

$P(\text{on}|C = 3) = 0.8$

$P(\text{on}|C = 3) = 0.9$

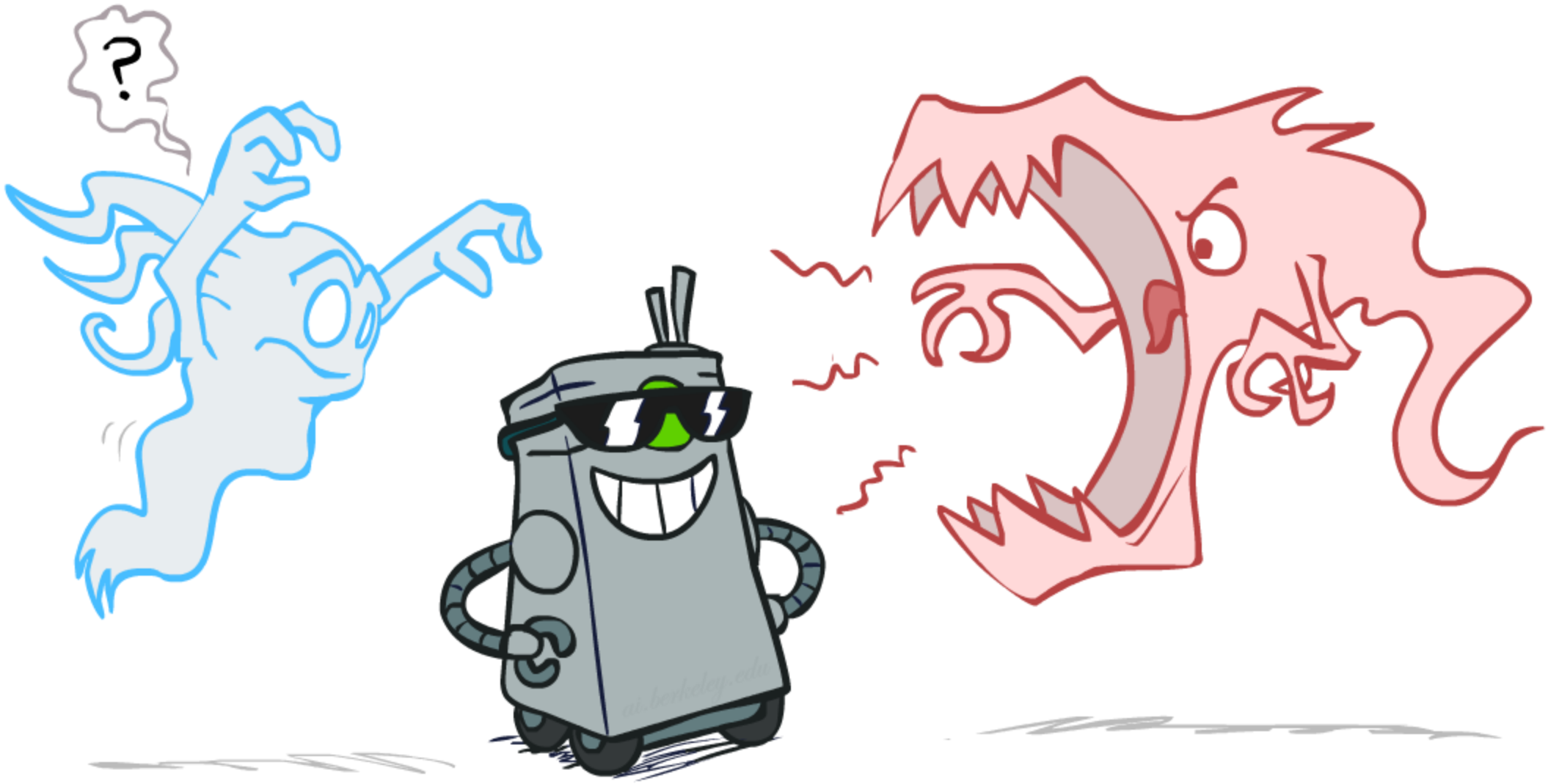
$P(\text{off}|C = 3) = 0.7$

$P(\text{on}|C = 3) = 0.0$

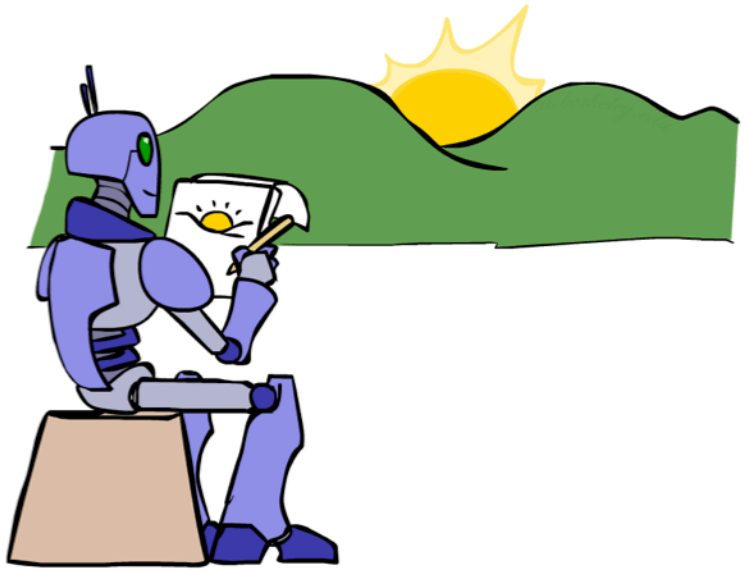


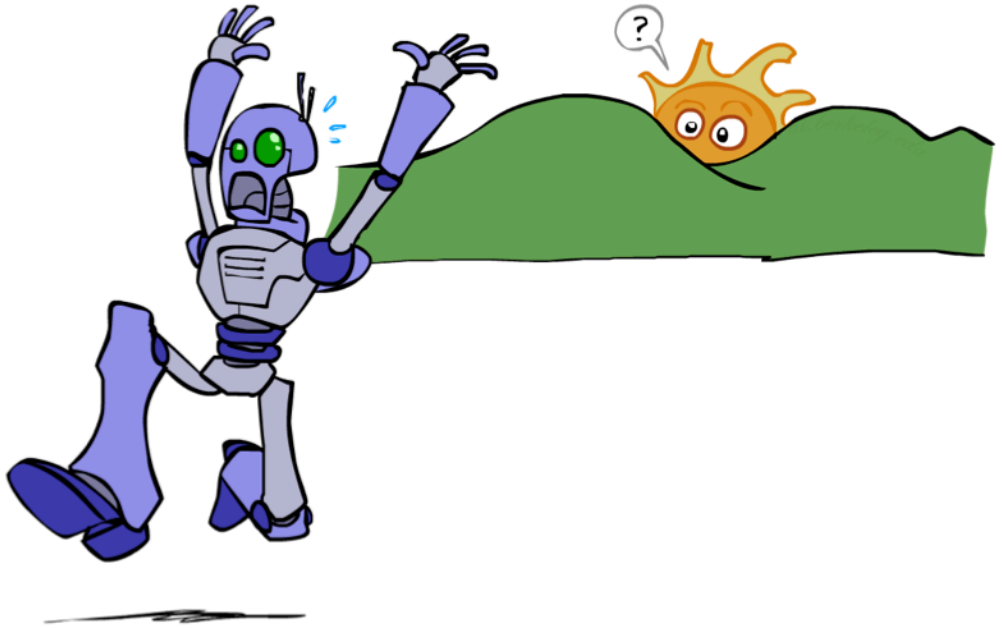
2 wins!!

Smoothing



Unseen Events





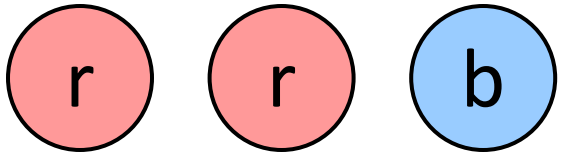
Laplace Smoothing

- Laplace's estimate:
  - Pretend you saw every outcome once more than you actually did

$$P_{LAP}(x) = \frac{c(x)+1}{\sum_x [c(x)+1]}$$

$$= \frac{c(x)+1}{N+|X|}$$

  - Can derive this estimate with *Dirichlet priors* (see cs281a)



$$P_{ML}(X) =$$

$$P_{LAP}(X) =$$

Laplace Smoothing

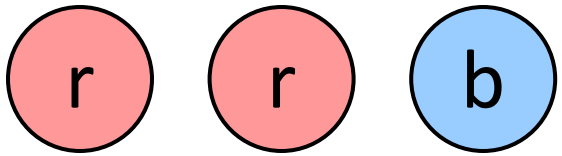
- Laplace's estimate (extended):
  - Pretend you saw every outcome k extra times

$$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$$

  - What's Laplace with k = 0?
  - k is the strength of the prior

- Laplace for conditionals:
  - Smooth each condition independently:

$$P_{LAP,k}(x|y) = \frac{c(x,y) + k}{c(y) + k|X|}$$

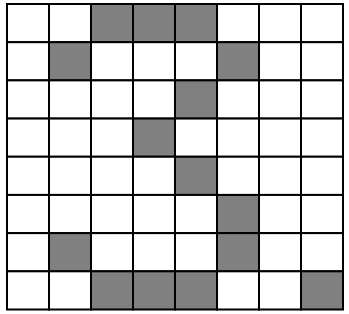


$$P_{LAP,0}(X) =$$

$$P_{LAP,1}(X) =$$

$$P_{LAP,100}(X) =$$

Estimation: Linear Interpolation*



- In practice, Laplace can perform poorly for P(X|Y):
  - When |X| is very large
  - When |Y| is very large
- Another option: linear interpolation
  - Also get the empirical P(X) from the data
  - Make sure the estimate of P(X|Y) isn't too different from the empirical P(X)

$$P_{LIN}(x|y) = \alpha \hat{P}(x|y) + (1.0 - \alpha)\hat{P}(x)$$

  - What if α is 0? 1?
- For even better ways to estimate parameters, as well as details of the math, see cs281a, cs288

Real NB: Smoothing

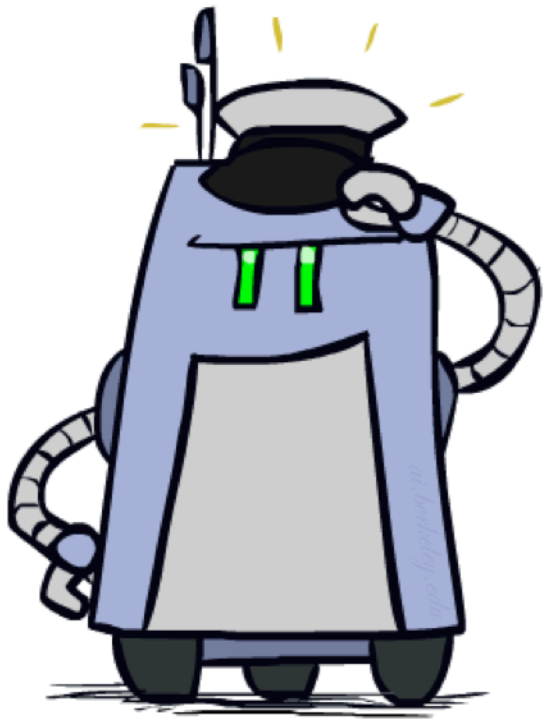
- For real classification problems, smoothing is critical
- New odds ratios:

$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

```
helvetica : 11.4
seems     : 10.8
group     : 10.2
ago       :  8.4
areas     :  8.3
...
```

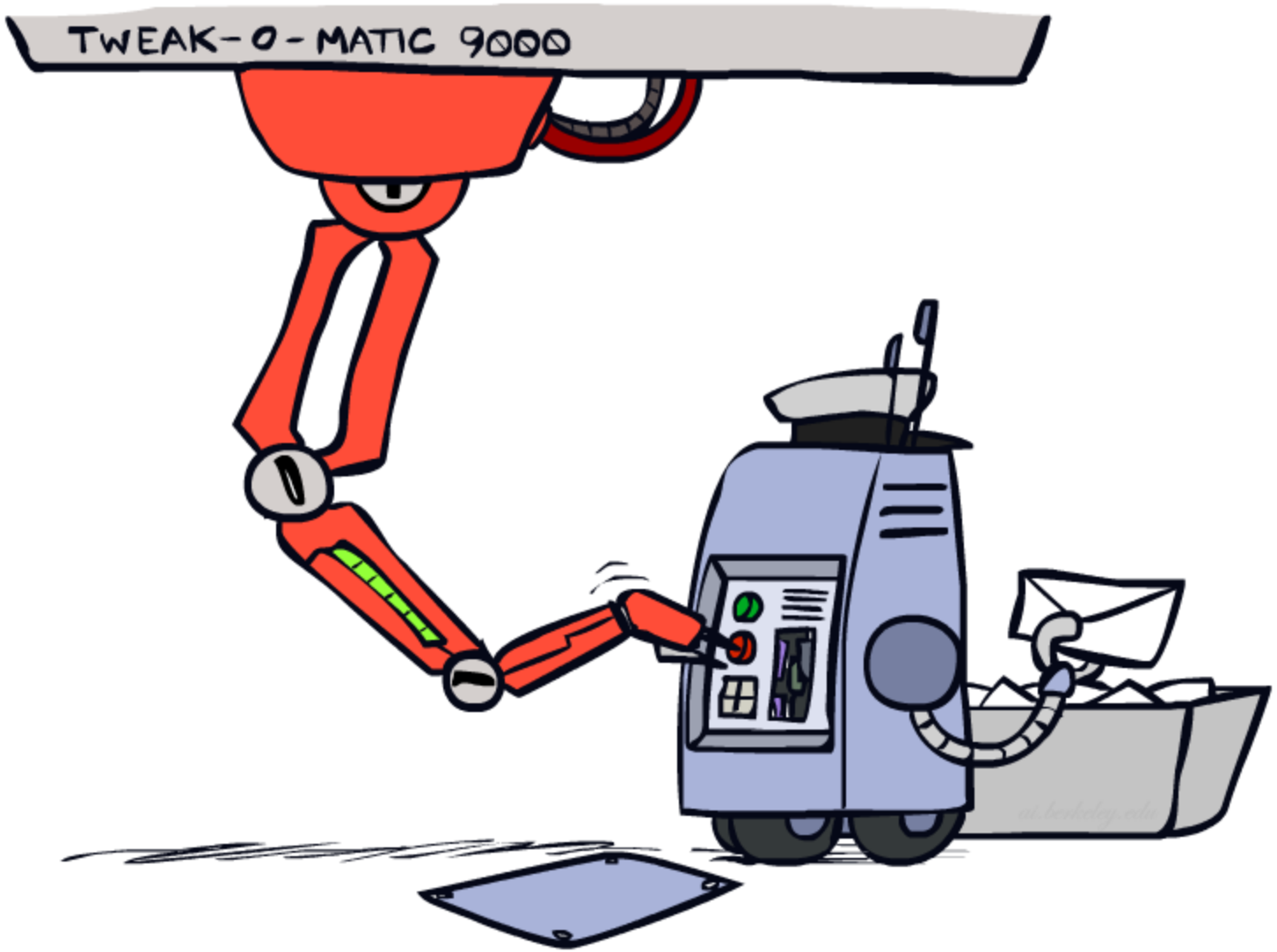
$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

```
verdana : 28.8
Credit  : 28.4
ORDER   : 27.2
<FONT>  : 26.9
money   : 26.5
...
```



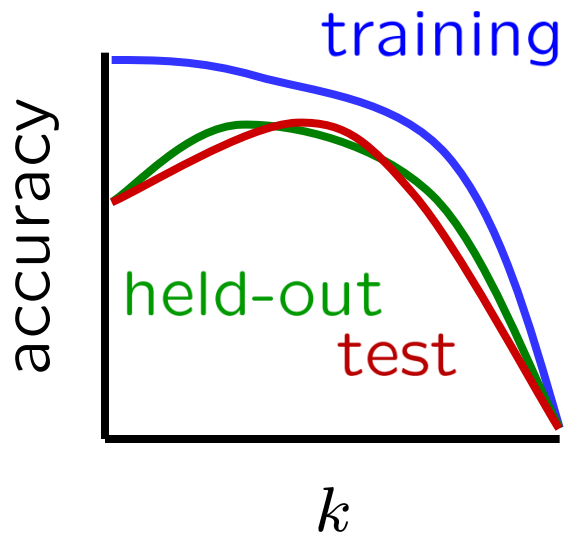
Do these make more sense?

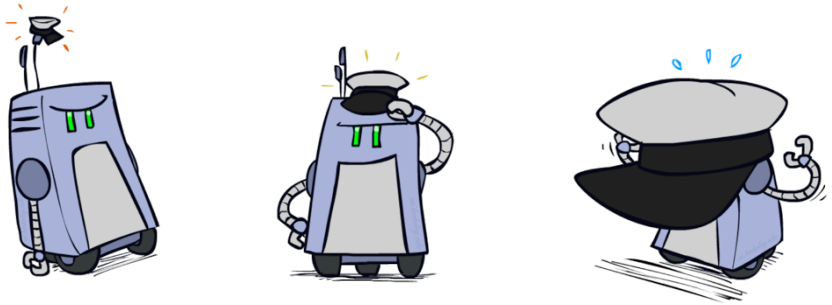
Tuning



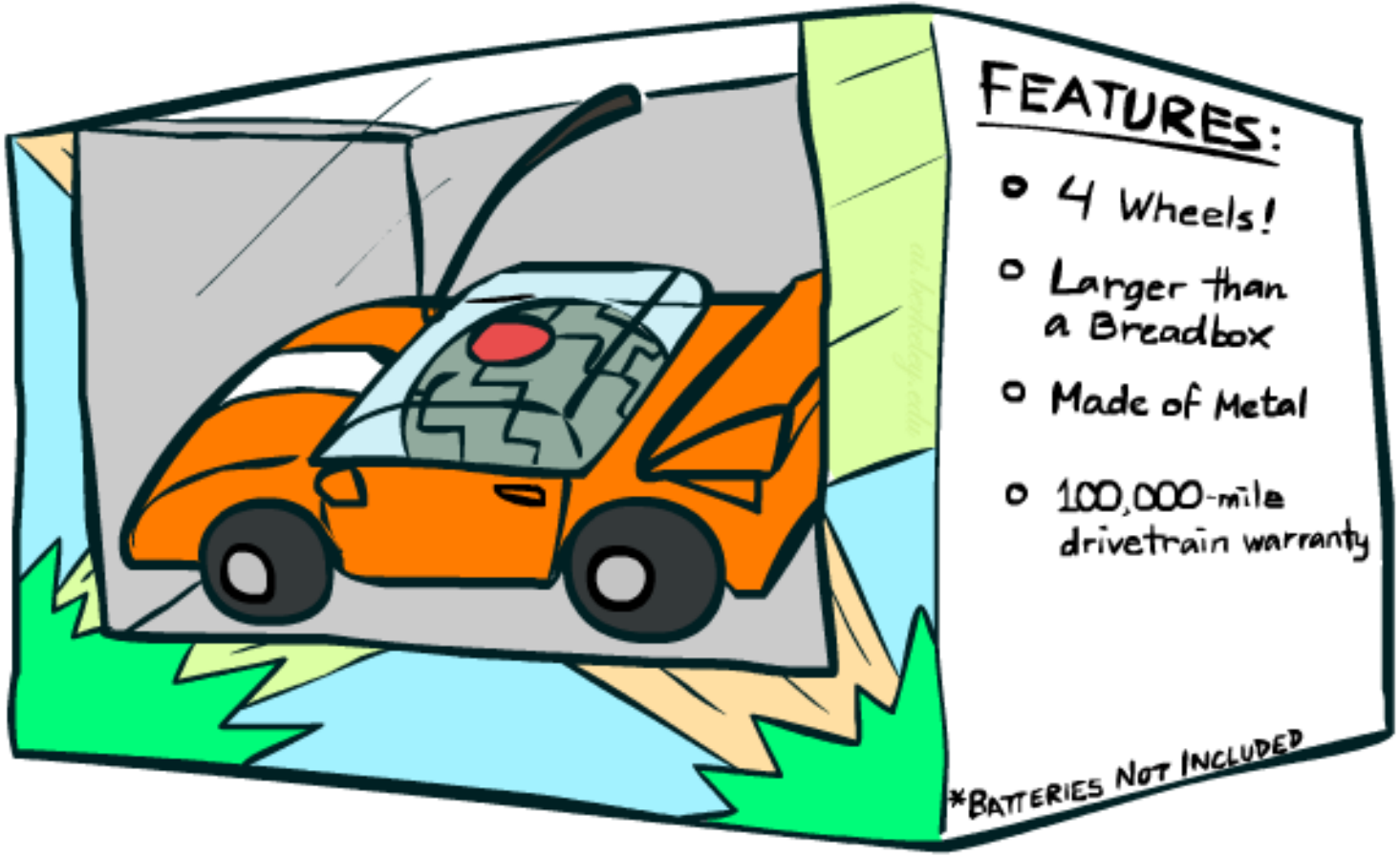
Tuning on Held-Out Data

- Now we've got two kinds of unknowns
  - Parameters: the probabilities P(X|Y), P(Y)
  - Hyperparameters: e.g. the amount / type of smoothing to do, k, α
- What should we learn where?
  - Learn parameters from training data
  - Tune hyperparameters on different data
    - Why?
  - For each value of the hyperparameters, train and test on the held-out data
  - Choose the best value and do a final test on the test data





Features

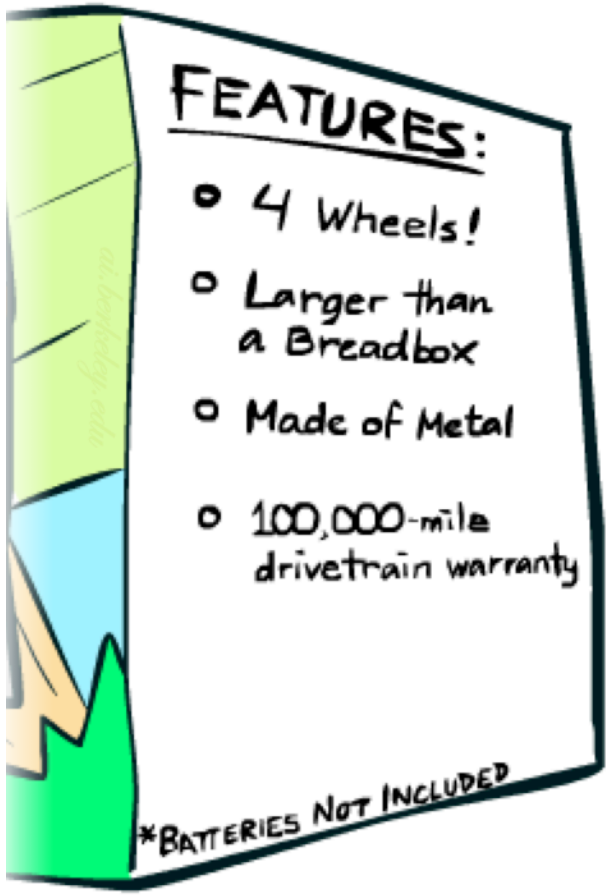

FEATURES:
4 Wheels!
Larger than a Breadbox
Made of Metal
100,000-mile drivetrain warranty
*BATTERIES NOT INCLUDED

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

- Next we'll talk about classifiers which let you easily add arbitrary features more easily, and, later, how to induce new features



Baselines

- First step: get a baseline
  - Baselines are very simple “straw man” procedures
  - Help determine how hard the task is
  - Help know what a “good” accuracy is
- Weak baseline: most frequent label classifier
  - Gives all test instances whatever label was most common in the training set
  - E.g. for spam filtering, might label everything as ham
  - Accuracy might be very high if the problem is skewed
  - E.g. calling everything “ham” gets 66%, so a classifier that gets 70% isn’t very good…
- For real research, usually use previous work as a (strong) baseline

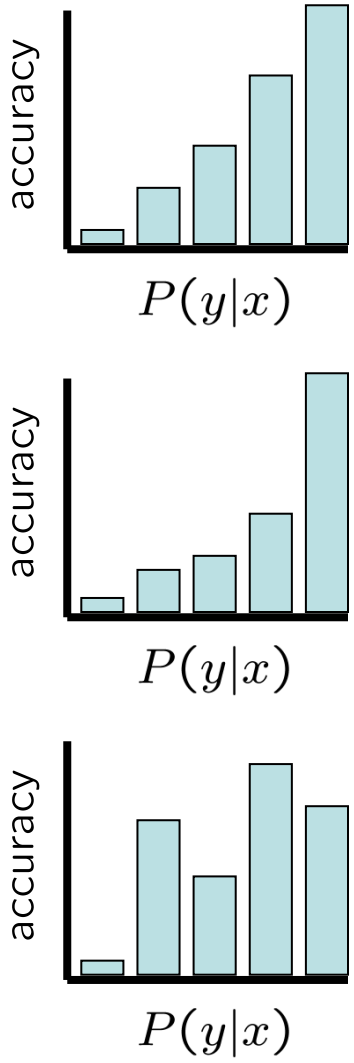
Confidences from a Classifier

- The confidence of a probabilistic classifier:
  - Posterior probability of the top label

$$\text{confidence}(x) = \max_y P(y|x)$$

  - Represents how sure the classifier is of the classification
  - Any probabilistic model will have confidences
  - No guarantee confidence is correct

- Calibration
  - Weak calibration: higher confidences mean higher accuracy
  - Strong calibration: confidence predicts accuracy rate
  - What's the value of calibration?

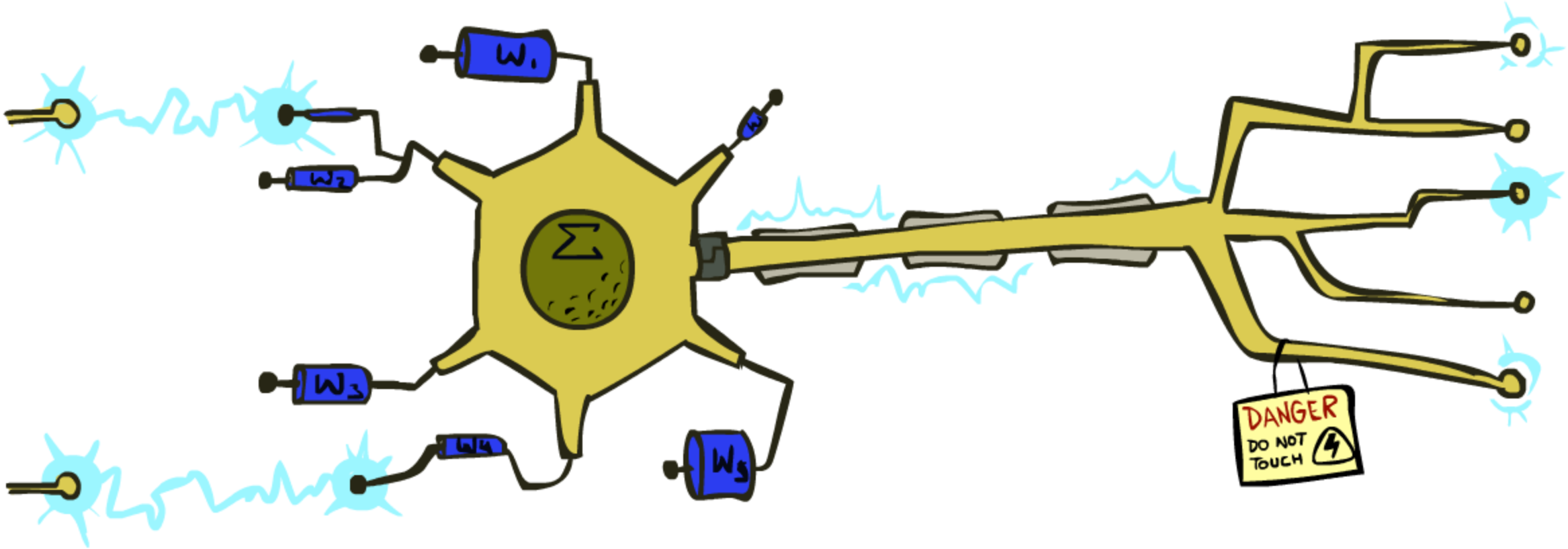


Summary

- Bayes rule lets us do diagnostic queries with causal probabilities
- The naïve Bayes assumption takes all features to be independent given the class label
- We can build classifiers out of a naïve Bayes model using training data
- Smoothing estimates is important in real systems
- Classifier confidences are useful, when you can get them

CS 188: Artificial Intelligence

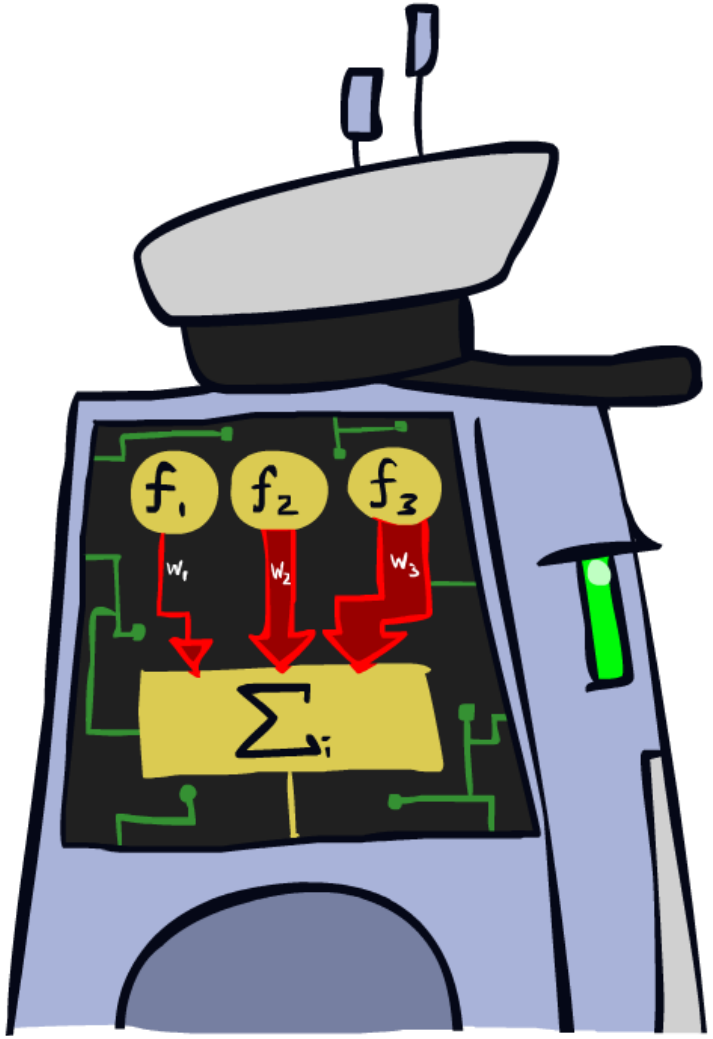
Perceptrons and Logistic Regression



Instructor: Nikita Kitaev

University of California, Berkeley

Linear Classifiers



Feature Vectors

x $\qquad$ $f(x)$ $\qquad$ 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
...
```

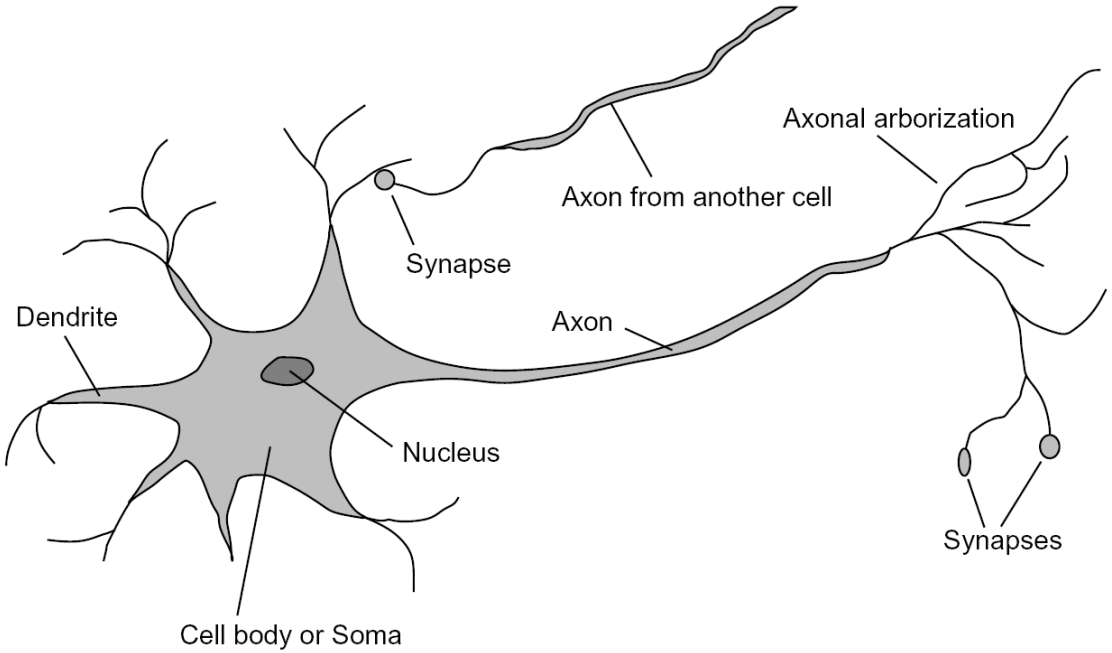
SPAM
or
+

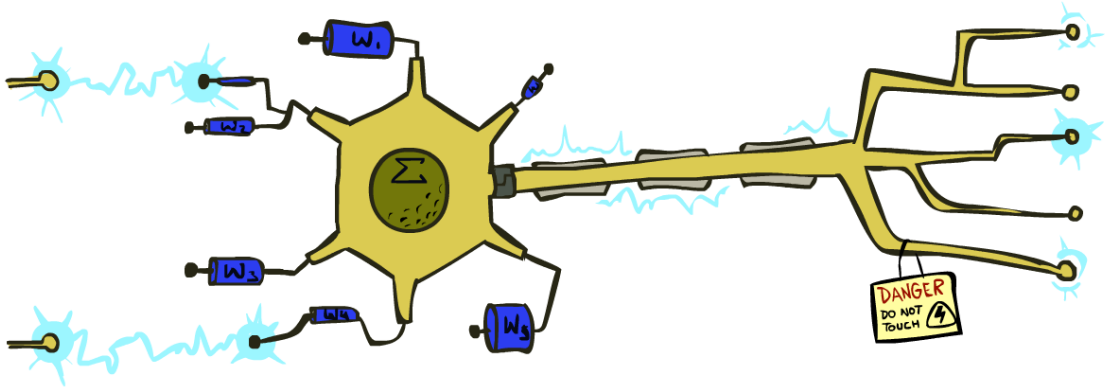
```
PIXEL-7,12   : 1
PIXEL-7,13   : 0
...
NUM_LOOPS    : 1
...
```

“2”

Some (Simplified) Biology

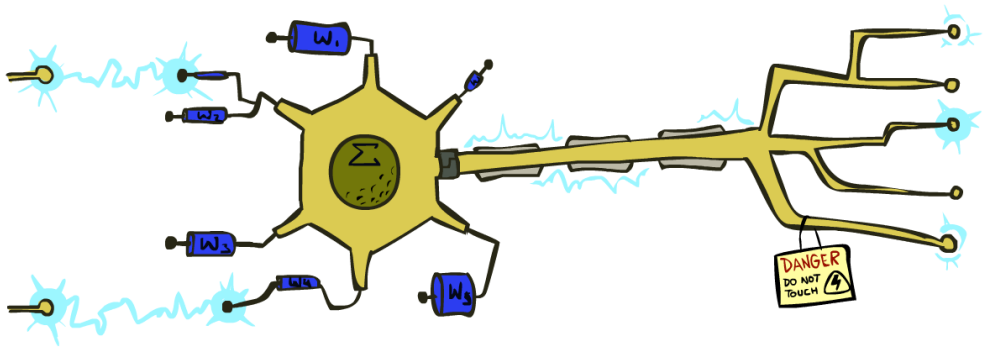
- Very loose inspiration: human neurons





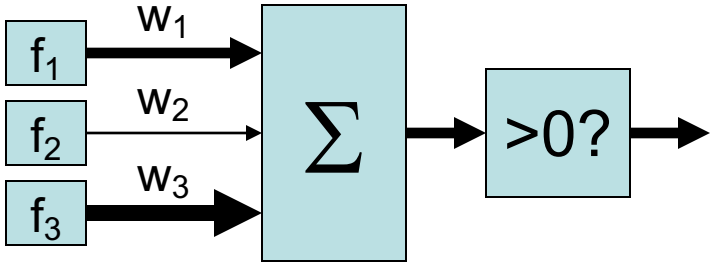
Linear Classifiers

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



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

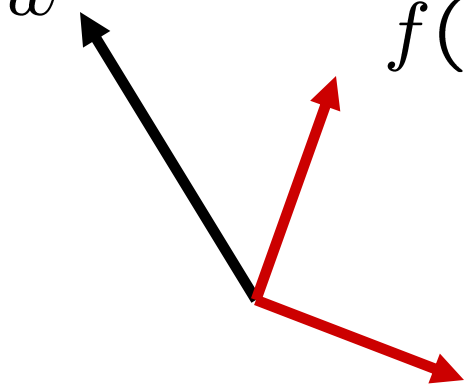
- If the activation is:
  - Positive, output +1
  - Negative, output -1



Weights

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

$$w = \begin{pmatrix} \texttt{\# free} & : 4 \\ \texttt{YOUR\_NAME} & :-1 \\ \texttt{MISSPELLED} & : 1 \\ \texttt{FROM\_FRIEND} & :-3 \\ \texttt{...} & \end{pmatrix}$$

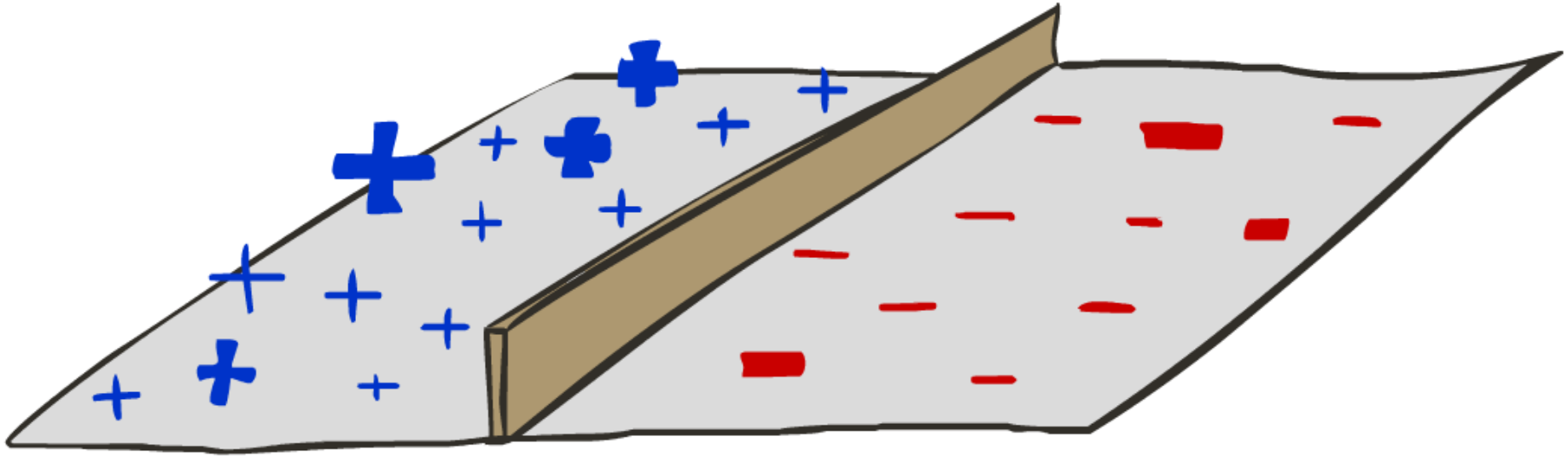


$$f(x_1) = \begin{pmatrix} \texttt{\# free} & : 2 \\ \texttt{YOUR\_NAME} & : 0 \\ \texttt{MISSPELLED} & : 2 \\ \texttt{FROM\_FRIEND} & : 0 \\ \texttt{...} & \end{pmatrix}$$

$$f(x_2) = \begin{pmatrix} \texttt{\# free} & : 0 \\ \texttt{YOUR\_NAME} & : 1 \\ \texttt{MISSPELLED} & : 1 \\ \texttt{FROM\_FRIEND} & : 1 \\ \texttt{...} & \end{pmatrix}$$

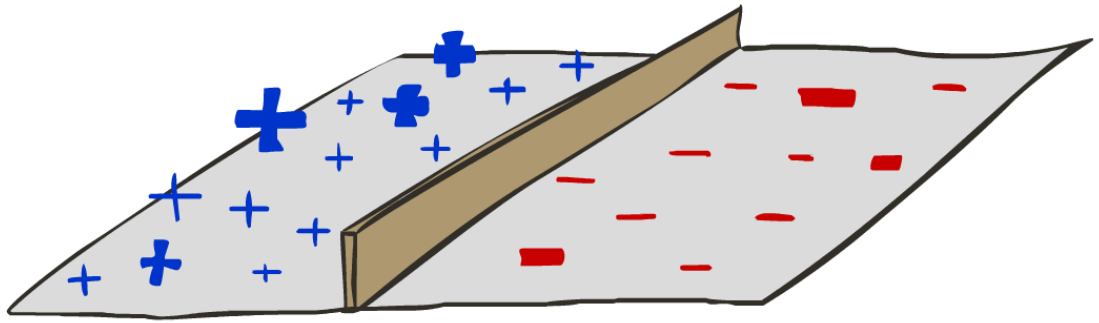
Dot product $w \cdot f$ positive means the positive class

Decision Rules



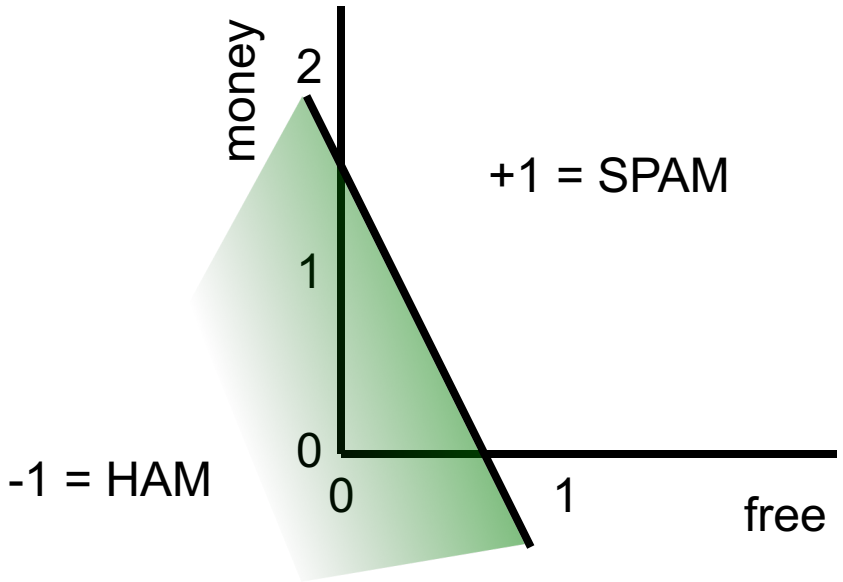
Binary Decision Rule

- 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



w

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

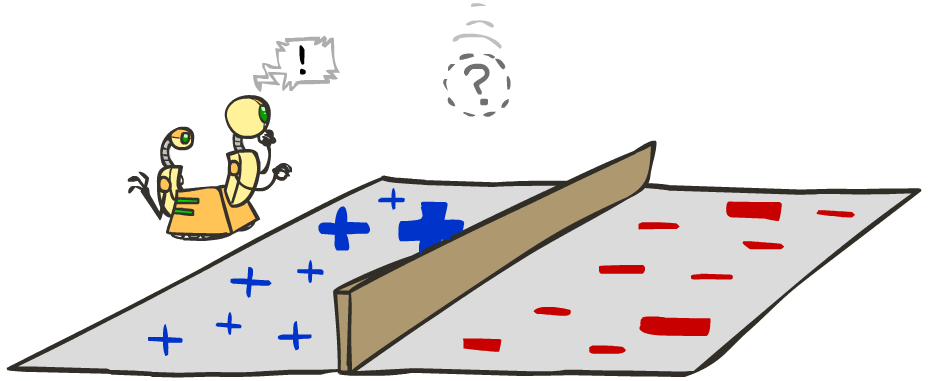


$f \cdot w = 0$

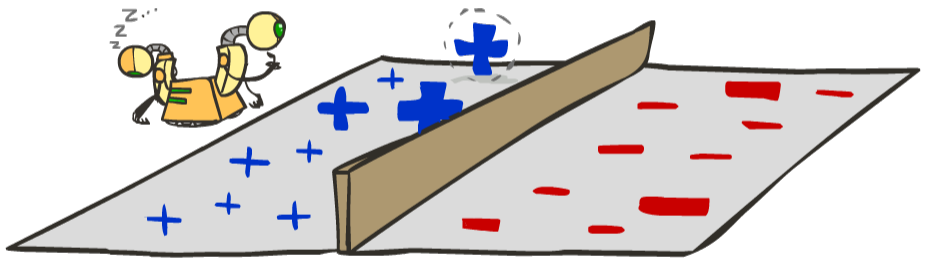
Weight Updates

Learning: Binary Perceptron

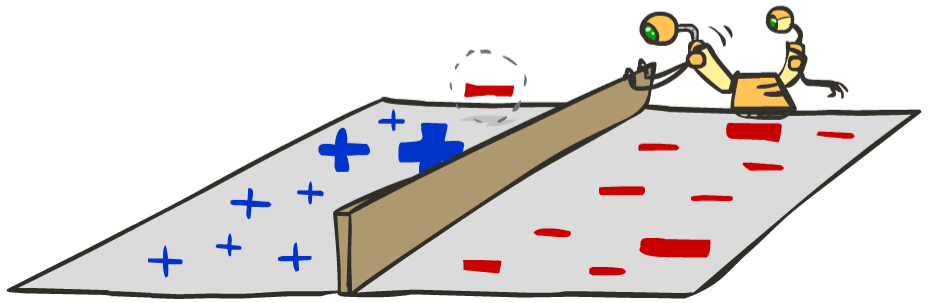
- Start with weights = 0
- For each training instance:
  - Classify with current weights



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



  - If wrong: adjust the weight vector



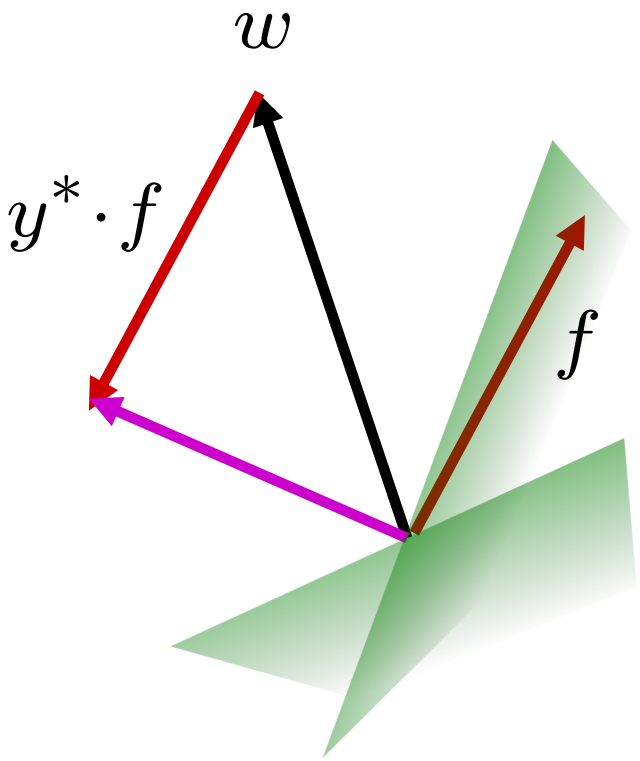
Learning: Binary Perceptron

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

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

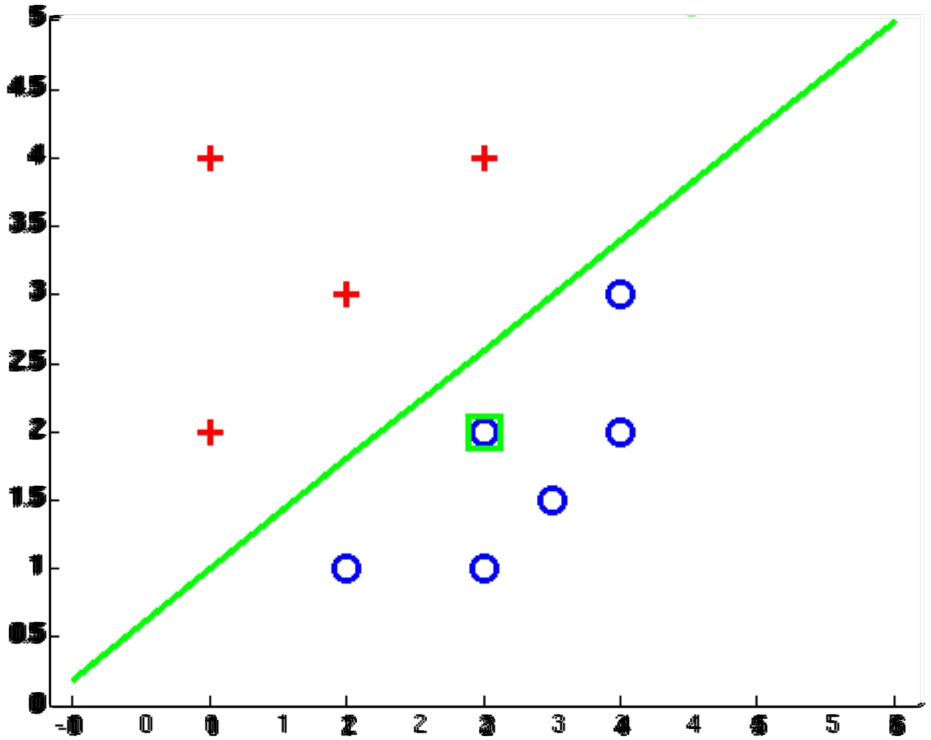
  - If correct (i.e., y=y*), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if y* is -1.

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



Examples: Perceptron

- Separable Case



Multiclass Decision Rule

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

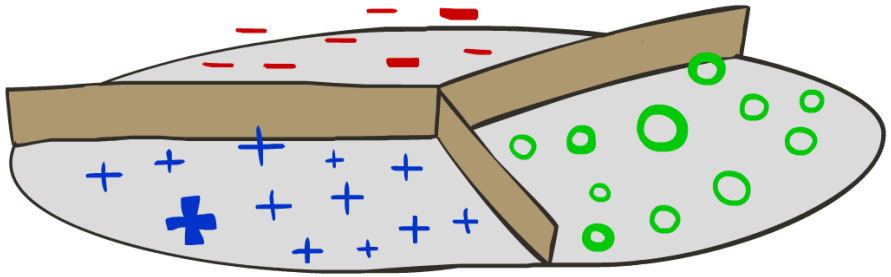
$$w_y$$

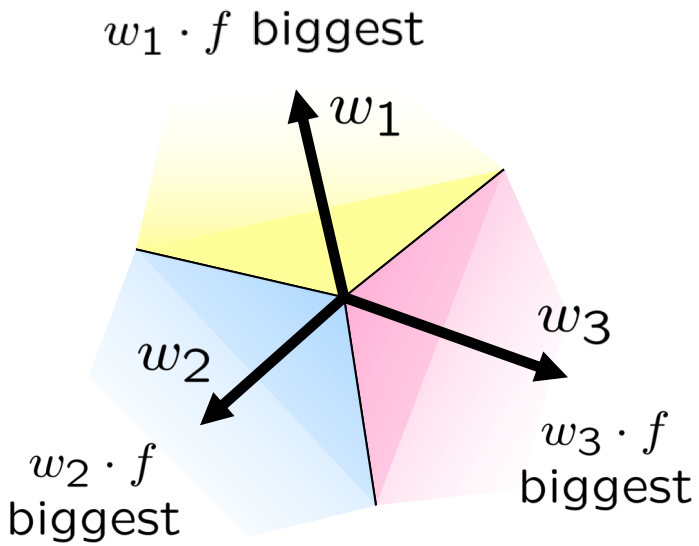
  - Score (activation) of a class y:

$$w_y \cdot f(x)$$

  - 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

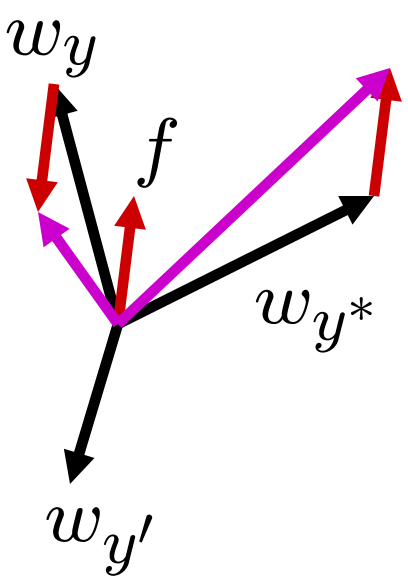
- Start with all weights = 0
- Pick up training examples one by one
- Predict with current weights

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

- 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}

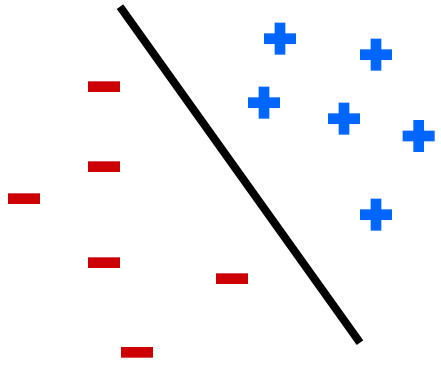
```
BIAS  : 0
win   : 0
game  : 0
vote  : 0
the   : 0
...
```

Properties of Perceptrons

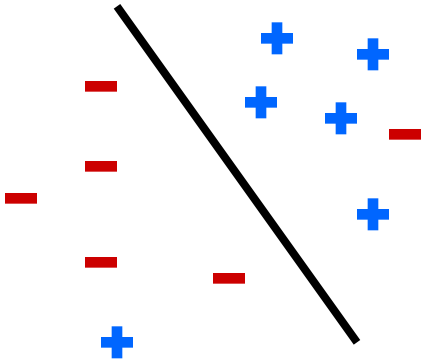
- Separability: true if some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

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

Separable

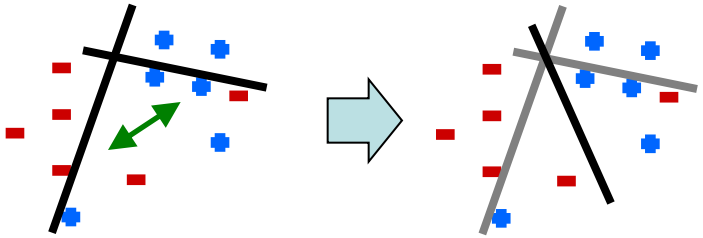


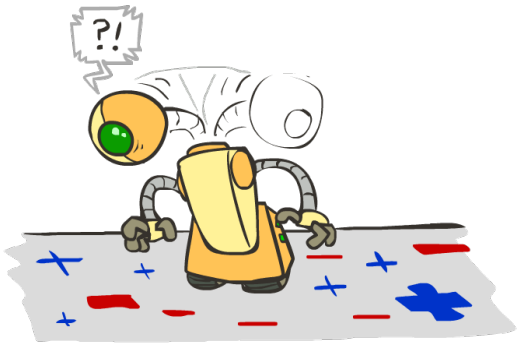
Non-Separable



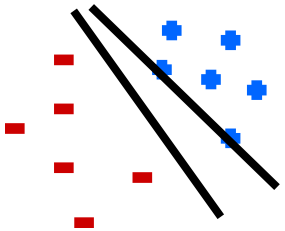
Problems with the Perceptron

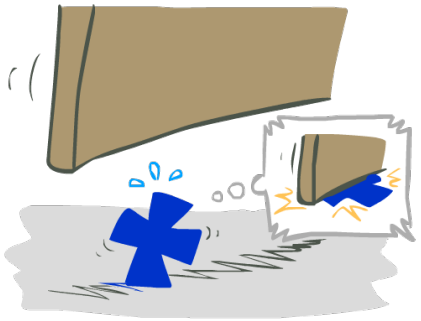
- Noise: if the data isn't separable, weights might thrash
  - Averaging weight vectors over time can help (averaged perceptron)



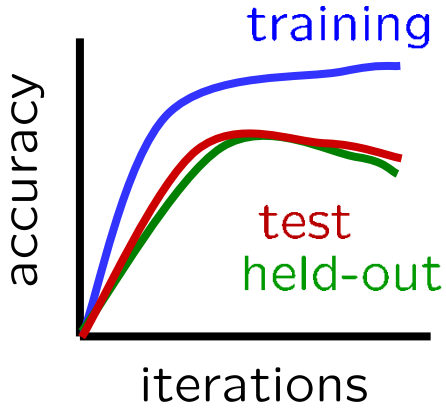


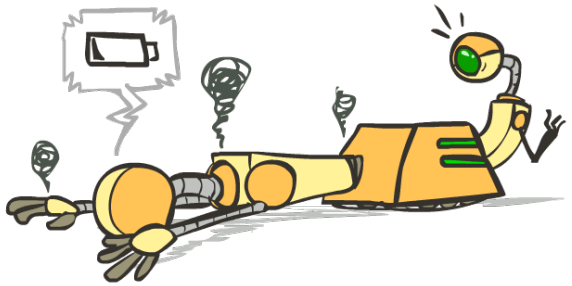
- Mediocre generalization: finds a "barely" separating solution



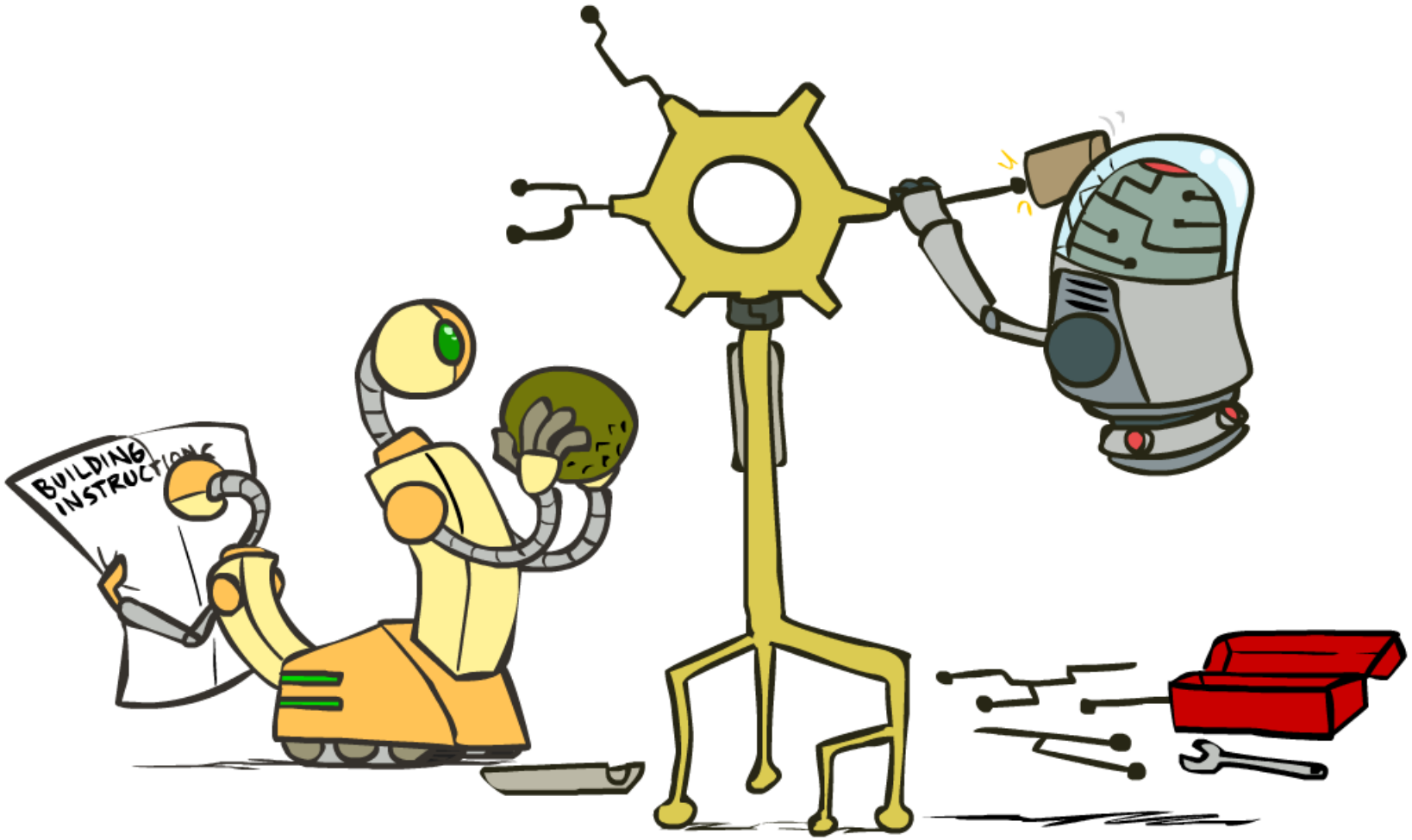


- Overtraining: test / held-out accuracy usually rises, then falls
  - Overtraining is a kind of overfitting



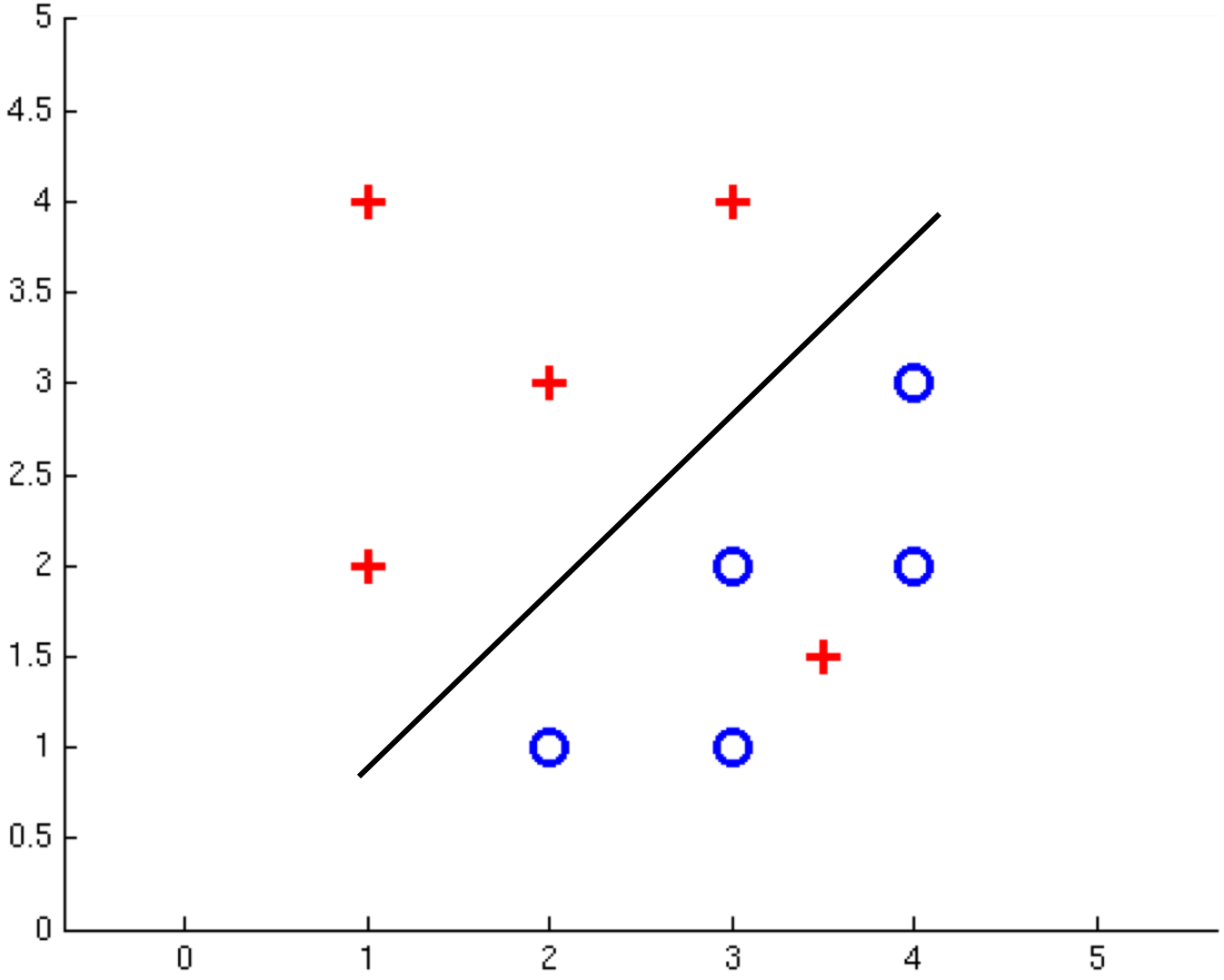


Improving the Perceptron

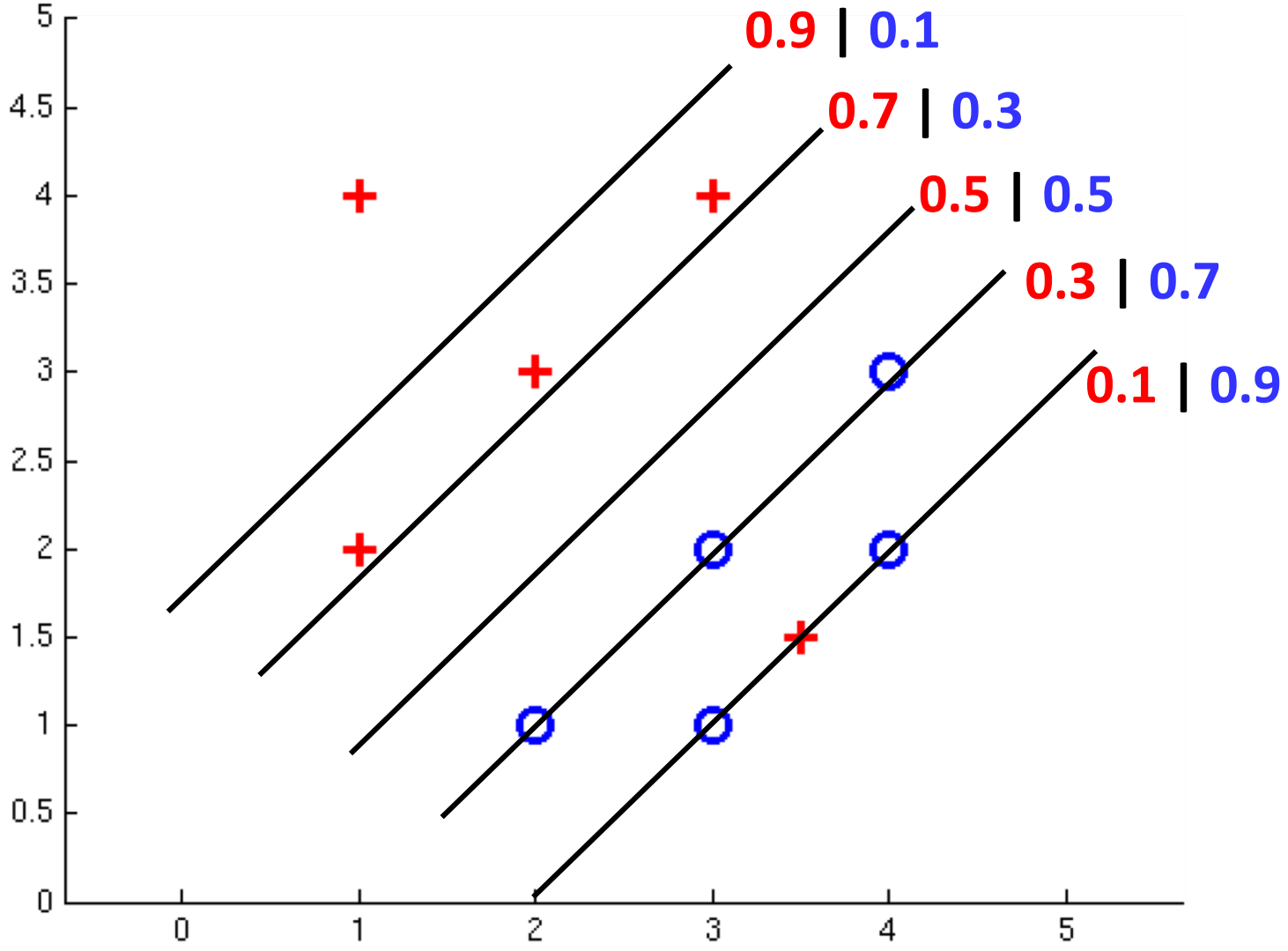


Non-Separable Case: Deterministic Decision

Even the best linear boundary makes at least one mistake



Non-Separable Case: Probabilistic Decision

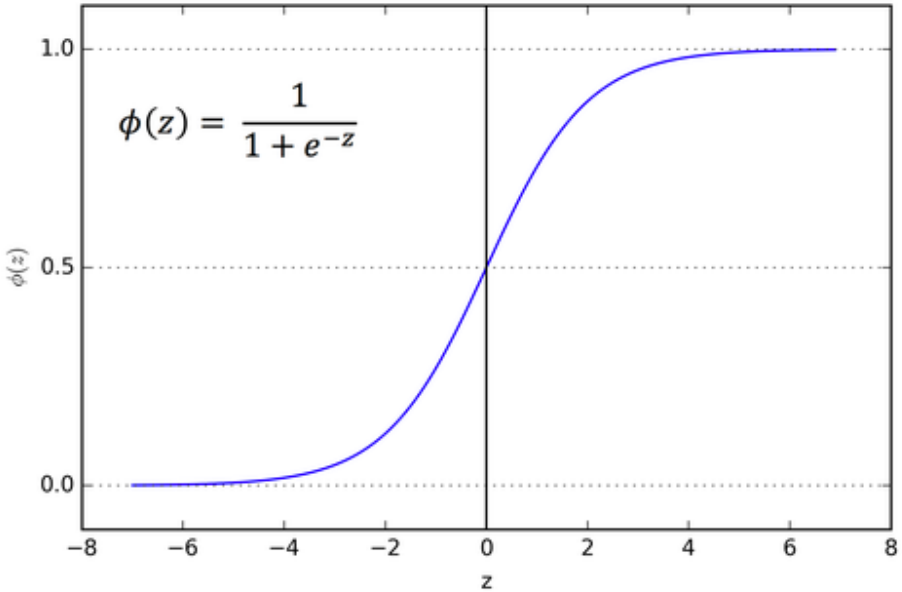


How to get probabilistic decisions?

- Perceptron scoring: $z = w \cdot f(x)$
- If $z = w \cdot f(x)$ very positive → want probability going to 1
- If $z = w \cdot f(x)$ very negative → want probability going to 0

- Sigmoid function

$$\phi(z) = \frac{1}{1 + e^{-z}}$$



Best w?

- Maximum likelihood estimation:

$$\max_w \quad ll(w) = \max_w \quad \sum_i \log P(y^{(i)}|x^{(i)}; w)$$

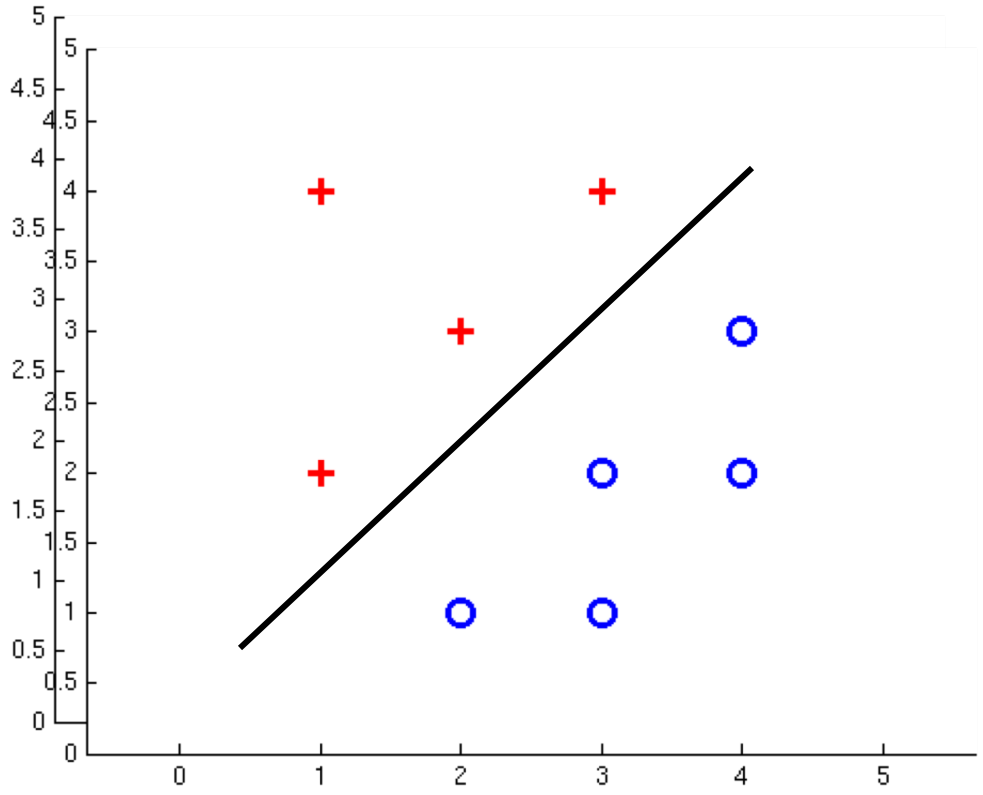
with:

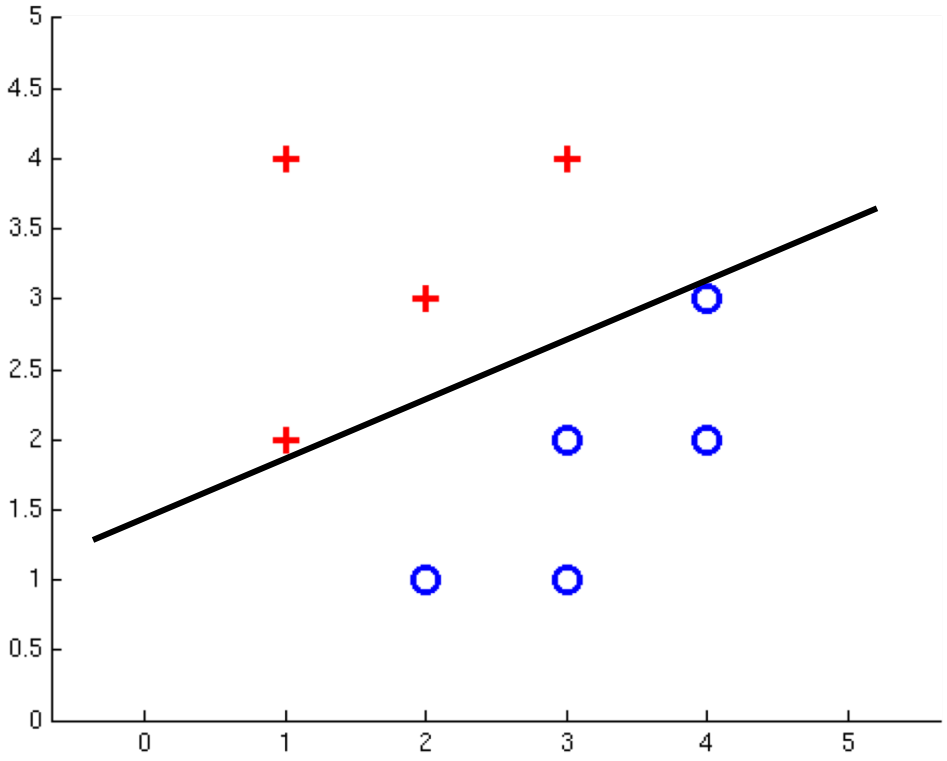
$$P(y^{(i)} = +1|x^{(i)}; w) = \frac{1}{1 + e^{-w \cdot f(x^{(i)})}}$$

$$P(y^{(i)} = -1|x^{(i)}; w) = 1 - \frac{1}{1 + e^{-w \cdot f(x^{(i)})}}$$

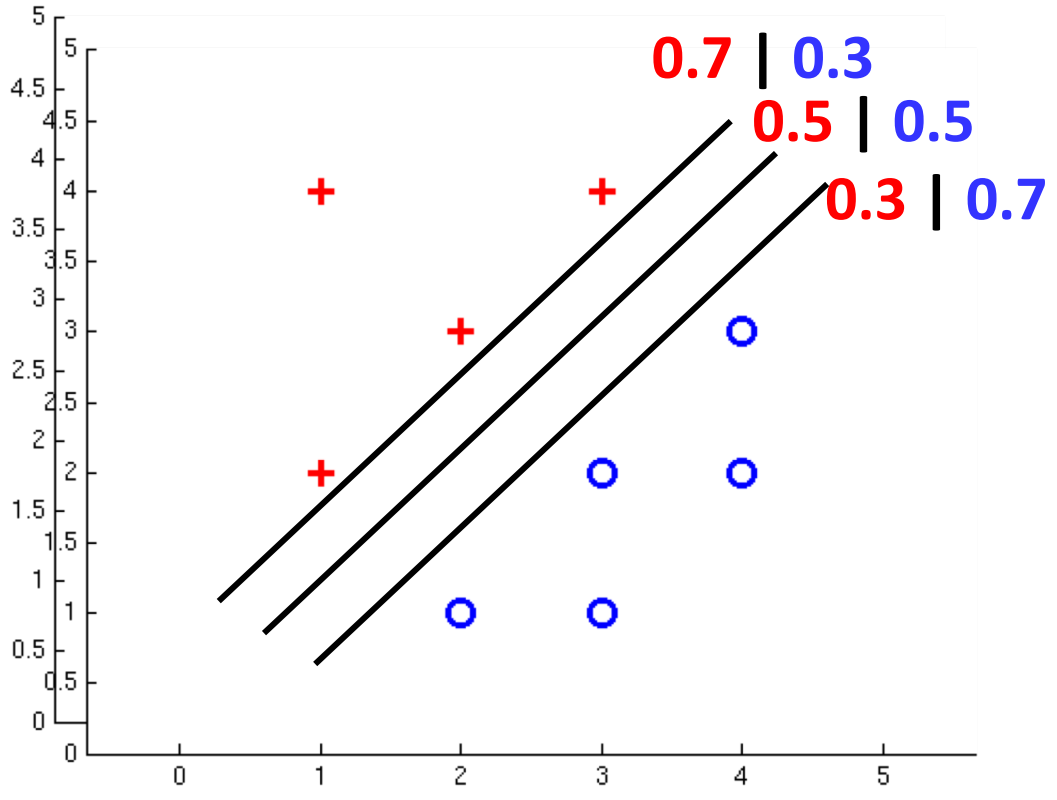
= Logistic Regression

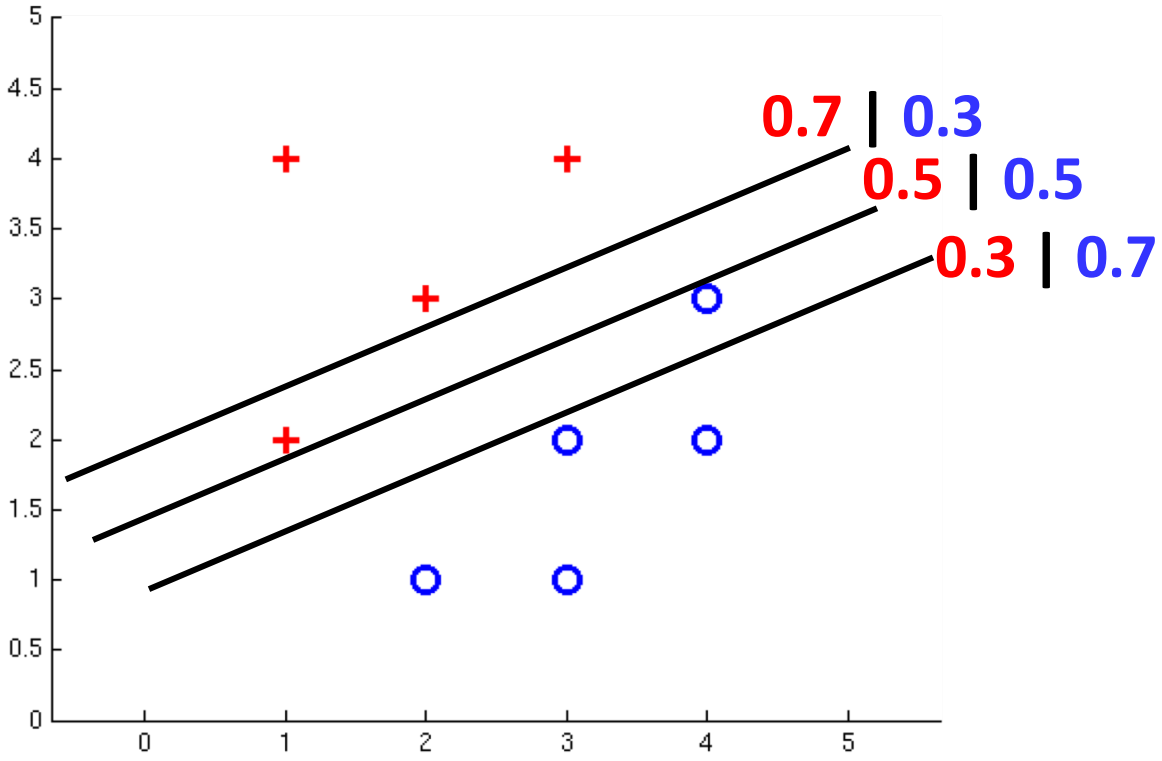
Separable Case: Deterministic Decision – Many Options



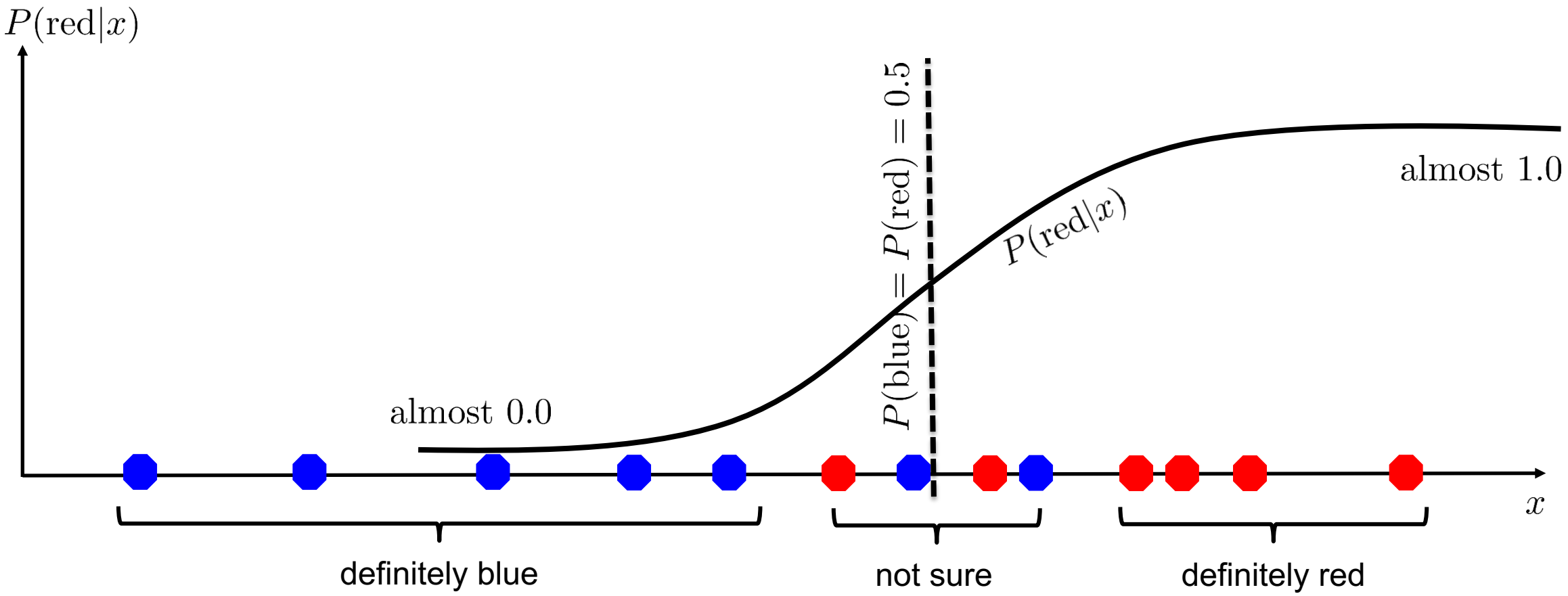


Separable Case: Probabilistic Decision – Clear Preference





A 1D Example

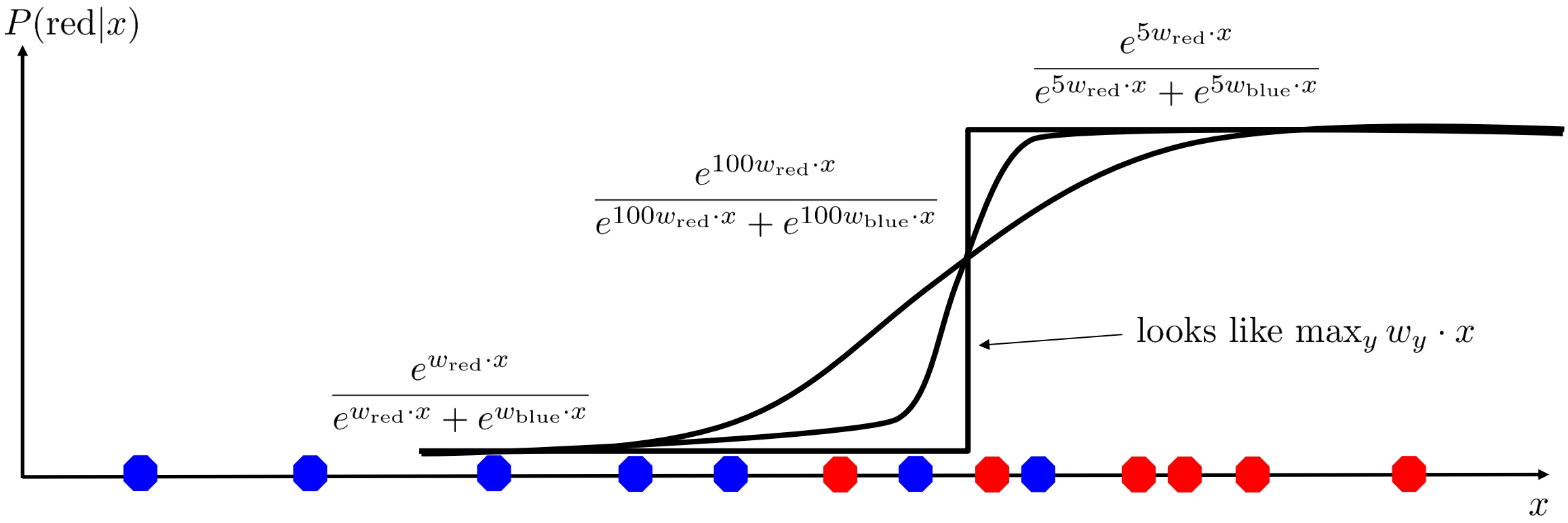


$$P(\text{red}|x) = \frac{e^{w_{\text{red}} \cdot x}}{e^{w_{\text{red}} \cdot x} + e^{w_{\text{blue}} \cdot x}}$$

probability increases exponentially as we move away from boundary

normalizer

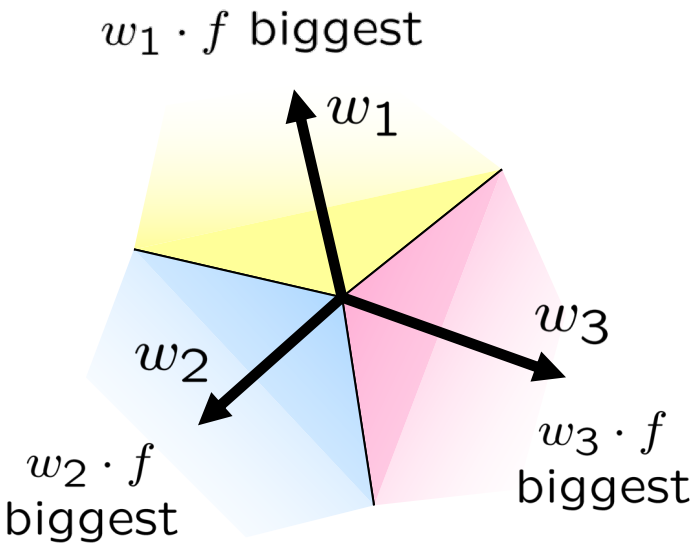
The *Soft* Max



$$P(\text{red}|x) = \frac{e^{w_{\text{red}}\cdot x}}{e^{w_{\text{red}}\cdot x} + e^{w_{\text{blue}}\cdot x}}$$

Multiclass Logistic Regression

- Recall Perceptron:
  - A weight vector for each class: w_y
  - Score (activation) of a class y: $w_y \cdot f(x)$
  - Prediction highest score wins $y = \arg\max_y \; w_y \cdot f(x)$



- How to make the scores into probabilities?

$$\underbrace{z_1, z_2, z_3}_{\text{original activations}} \rightarrow \underbrace{\frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}, \frac{e^{z_2}}{e^{z_1}+e^{z_2}+e^{z_3}}, \frac{e^{z_3}}{e^{z_1}+e^{z_2}+e^{z_3}}}_{\text{softmax activations}}$$

Best w?

- Maximum likelihood estimation:

$$\max_w \quad ll(w) = \max_w \quad \sum_i \log P(y^{(i)}|x^{(i)}; w)$$

with: $$P(y^{(i)}|x^{(i)}; w) = \frac{e^{w_{y^{(i)}} \cdot f(x^{(i)})}}{\sum_y e^{w_y \cdot f(x^{(i)})}}$$

= Multi-Class Logistic Regression

Next Lecture

- Optimization
  - i.e., how do we solve:

$$\max_{w} \quad ll(w) = \max_{w} \quad \sum_{i} \log P(y^{(i)}|x^{(i)}; w)$$