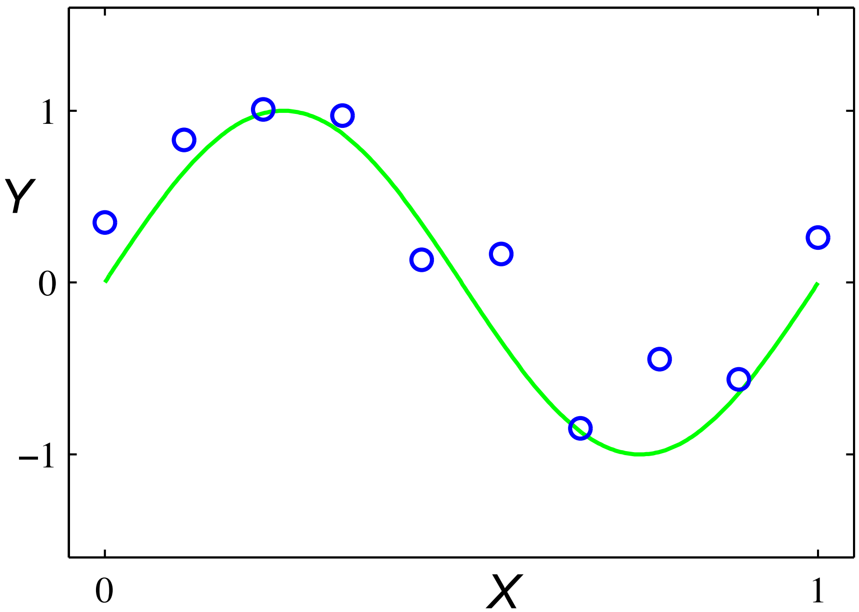
Introduction to Learning
Lecture 2

David Sontag
New York University

Slides adapted from Luke Zettlemoyer, Vibhav Gogate, Pedro Domingos, and Carlos Guestrin

Second example: Regression

Dataset: 10 (X,Y) points generated from a sin function, with noise



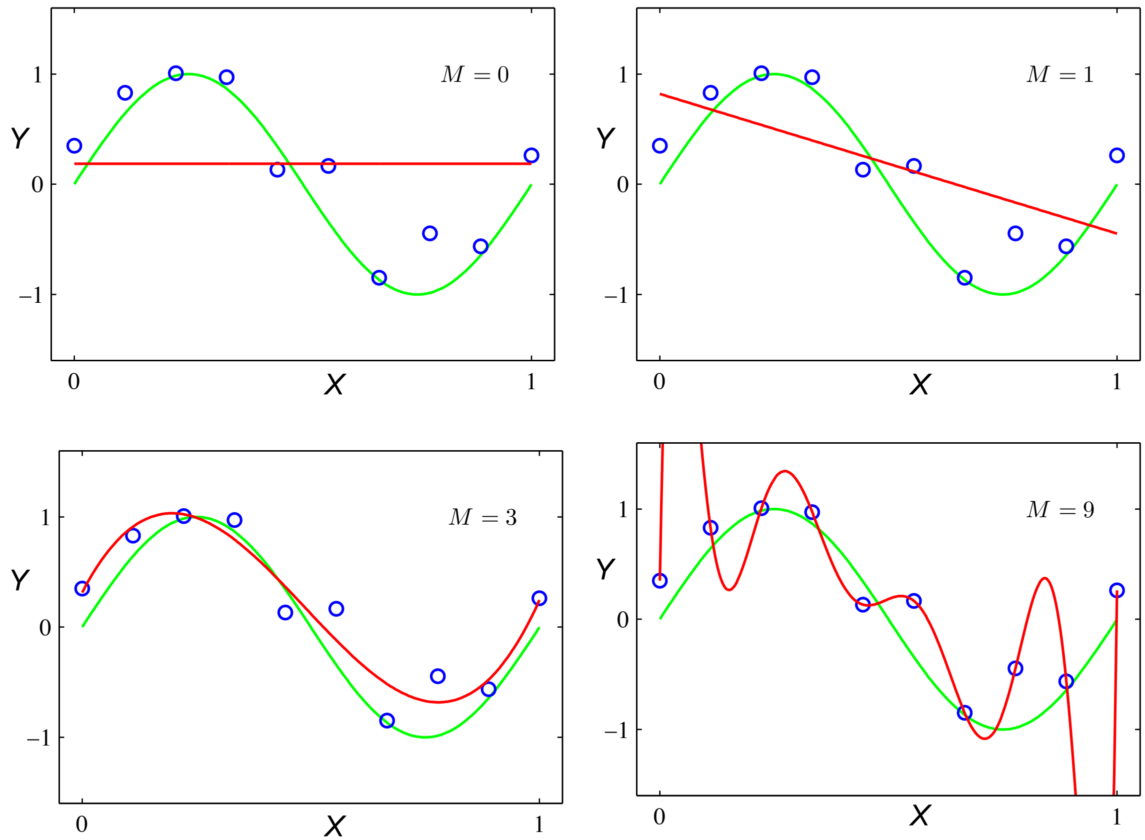
- Regression:
  - $f : X \rightarrow Y$
  - $X = \Re$
  - $Y = \Re$

[Bishop]

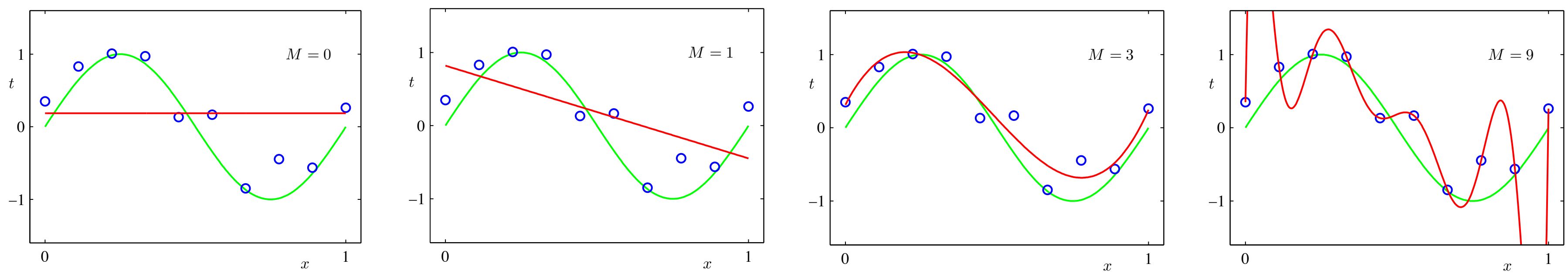
Degree-M Polynomials

How about letting *f* be a degree M polynomial?

•Which one is **best**?



Hypo. Space: Degree-N Polynomials



We measure error using a *loss function* $L(y, \hat{y})$

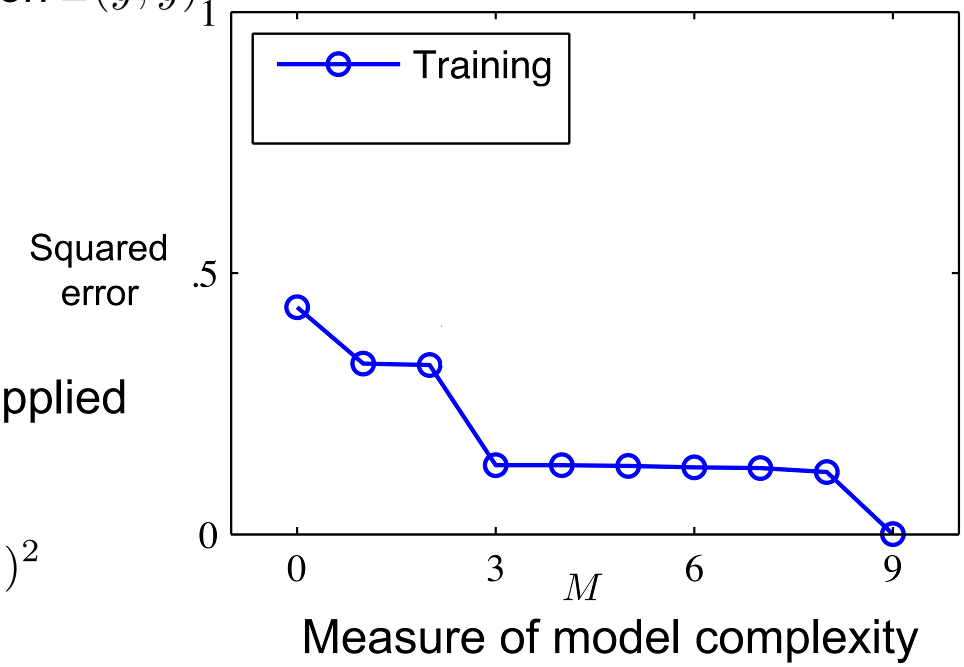
For regression, a common choice is squared loss:

$$L(y_i, f(x_i)) = (y_i - f(x_i))^2$$

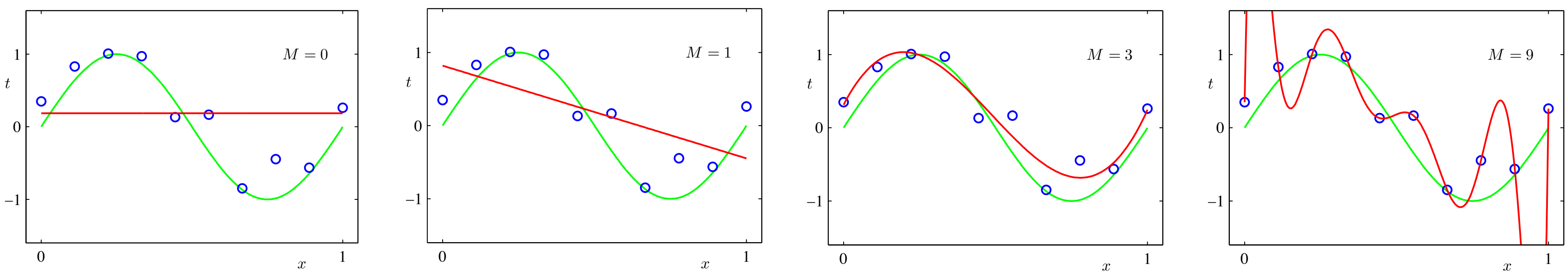
The *empirical loss* of the function *f* applied to the training data is then:

$$\frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)) = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2$$

Learning curve



Hypo. Space: Degree-N Polynomials



We measure error using a *loss function* $L(y, \hat{y})$

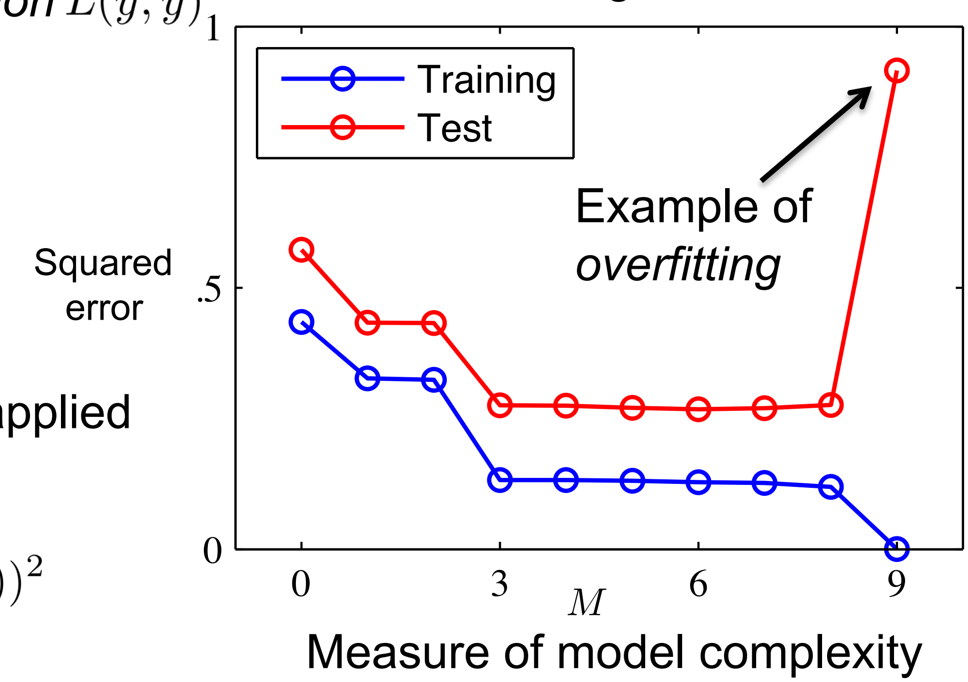
For regression, a common choice is squared loss:

$$L(y_i, f(x_i)) = (y_i - f(x_i))^2$$

The *empirical loss* of the function *f* applied to the training data is then:

$$\frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)) = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2$$

Learning curve



Binary classification

- Input: email
- Output: spam/ham
- Setup:
  - Get a large collection of example emails, each labeled "spam" or "ham"
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future emails

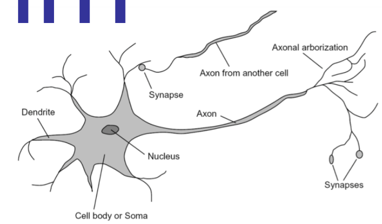
- Features: The attributes used to make the ham / spam decision
  - Words: FREE!
  - Text Patterns: $dd, CAPS
  - Non-text: SenderInContacts
  - …

> Dear Sir.
>
> First, I must solicit your confidence in this transaction, this is by virture of its nature as being utterly confidencial and top secret. …

> TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.
>
> 99 MILLION EMAIL ADDRESSES FOR ONLY $99

> Ok, Iknow this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

The perceptron algorithm



- 1957: Perceptron algorithm invented by Rosenblatt

  Wikipedia: “A handsome bachelor, he drove a classic MGA sports… for several years taught an interdisciplinary undergraduate honors course entitled "Theory of Brain Mechanisms" that drew students equally from Cornell's Engineering and Liberal Arts colleges…this course was a melange of ideas .. experimental brain surgery on epileptic patients while conscious, experiments on .. the visual cortex of cats, ... analog and digital electronic circuits that modeled various details of neuronal behavior (i.e. the perceptron itself, as a machine).”

  – Built on work of Hebbs (1949); also developed by Widrow-Hoff (1960)

- 1960: Perceptron Mark 1 Computer – hardware implementation

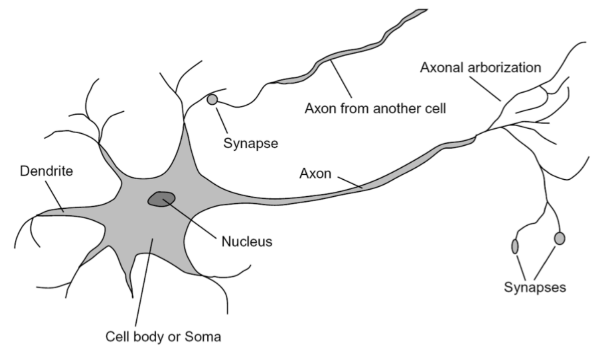
- 1969: Minksky & Papert book shows perceptrons limited to *linearly separable* data, and Rosenblatt dies in boating accident

- 1970’s: Learning methods for two-layer neural networks

[William Cohen]

Linear Classifiers

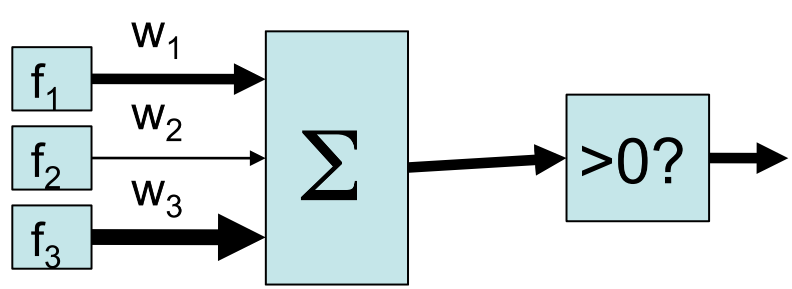
- Inputs are feature values
- Each feature has a weight
- Sum is the activation



Important note: changing notation!

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

- If the activation is:
  - Positive, output *class 1*
  - Negative, output *class 2*



Example: Spam

- Imagine 3 features (spam is "positive" class):
  1. free (number of occurrences of "free")
  2. money (occurrences of "money")
  3. BIAS (intercept, always has value 1)

$$w \cdot f(x)$$

$$\sum_i w_i \cdot f_i(x)$$

x

"free money"

$f(x)$

```
BIAS  :  1
free  :  1
money :  1
...
```

w

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

$$(1)(-3) \quad +$$
$$(1)(4) \quad +$$
$$(1)(2) \quad +$$
$$\ldots$$
$$= 3$$

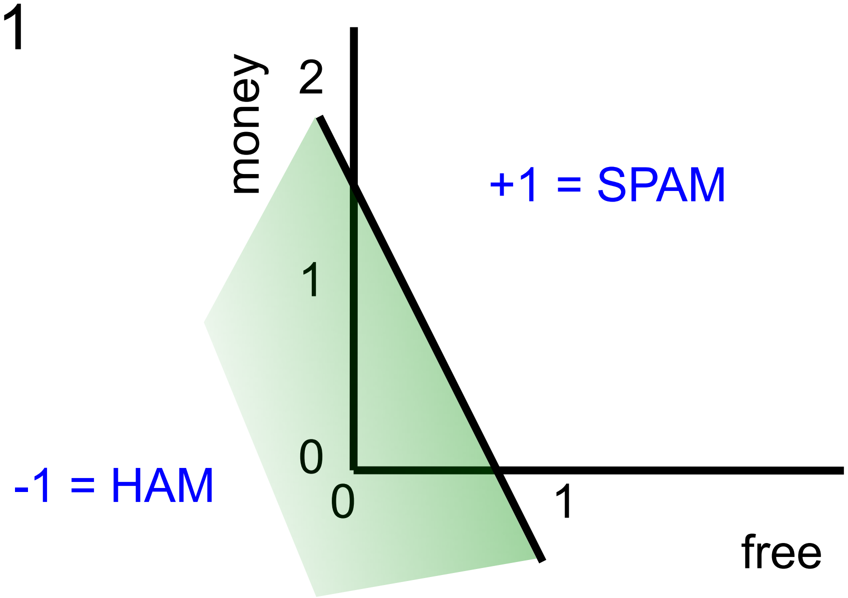
w.f(x) > 0 ➔ SPAM!!!

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
...		



$f \cdot w = 0$

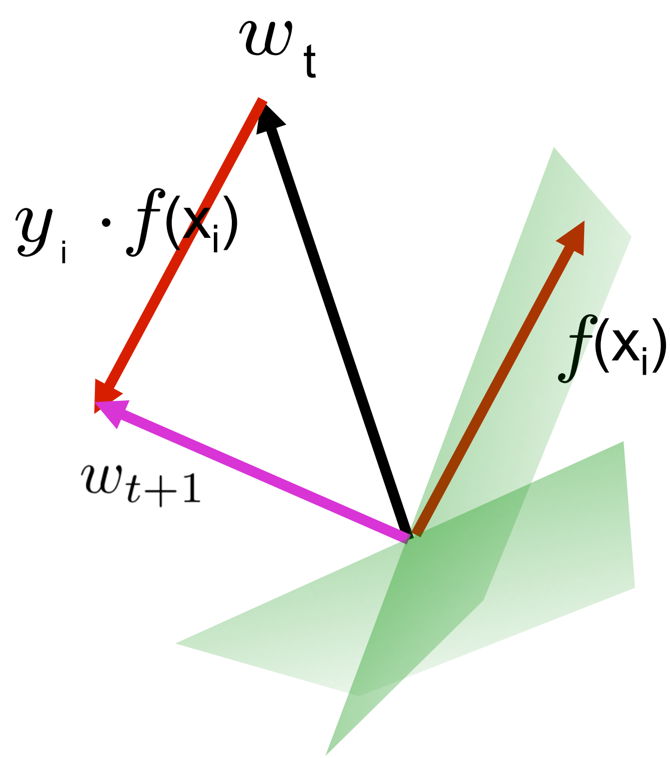
The perceptron algorithm

- Start with weight vector = $\vec{0}$
- For each training instance (x_i, y_i):
  - Classify with current weights

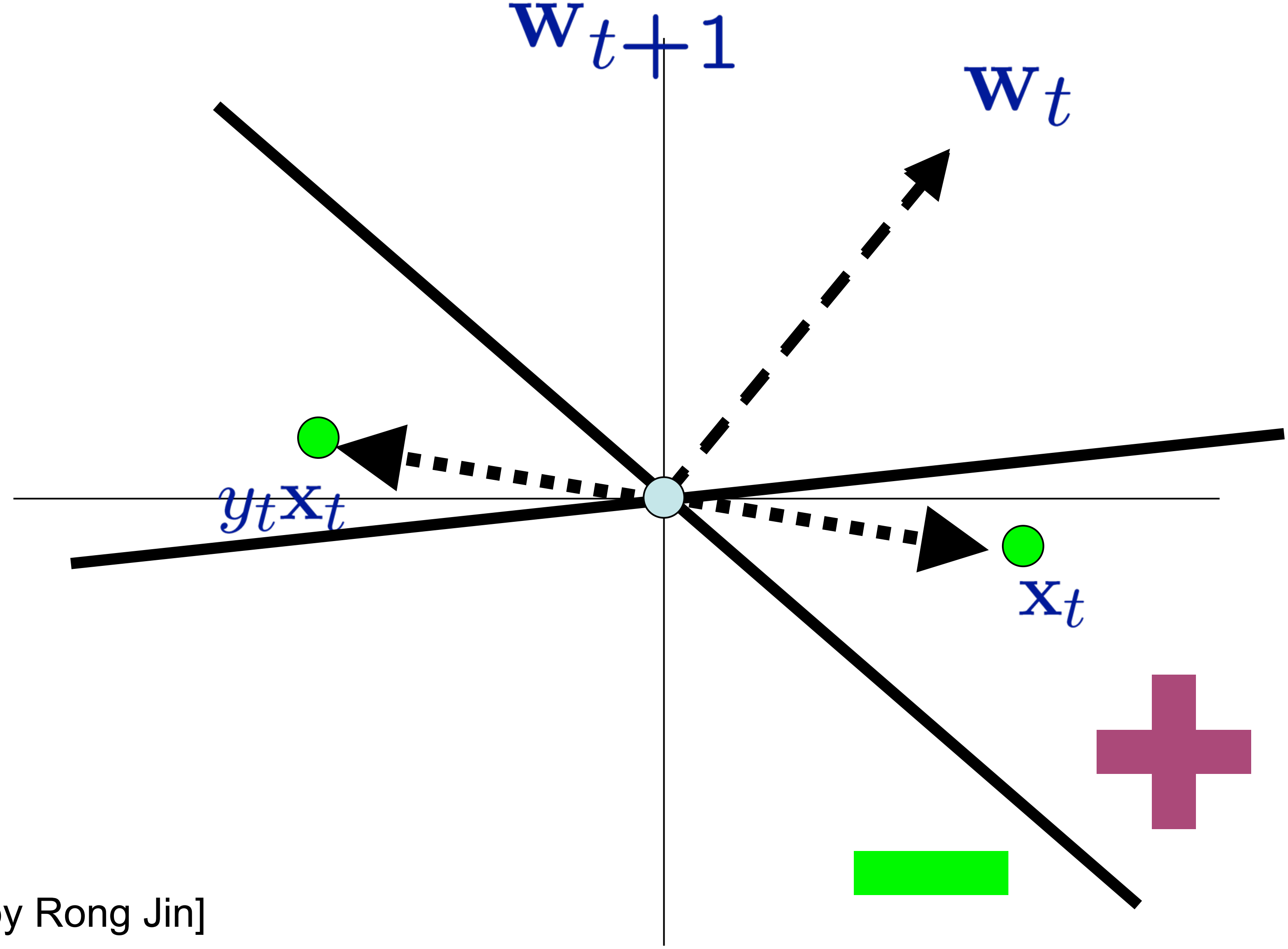
$$y = \begin{cases} +1 & \text{if} \;\; w \cdot f(x_i) \geq 0 \\ -1 & \text{if} \;\; w \cdot f(x_i) < 0 \end{cases}$$

  - If correct (i.e., $y=y_i$), no change!
  - If wrong: update

$$w = w + y_i \, f(x_i)$$



Geometrical Interpretation



[Slide by Rong Jin]

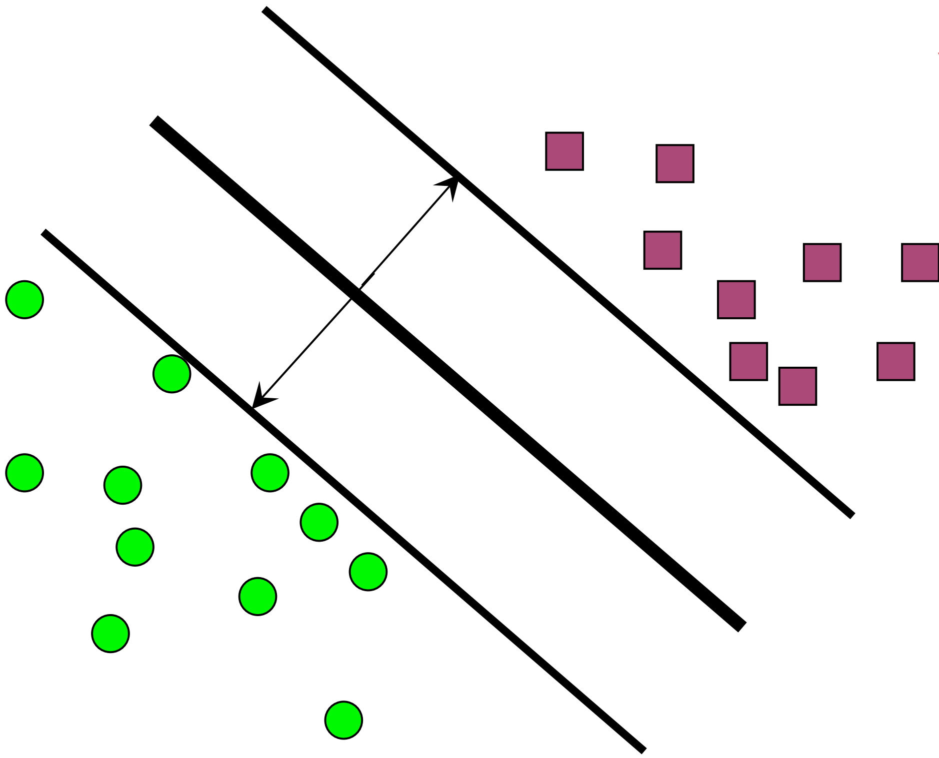
What questions should we ask about a learning algorithm?

- What is the perceptron algorithm’s running time?
- If a weight vector with small training error exists, will perceptron find it?
- How well does the resulting classifier generalize to unseen data?

Linearly Separable

$\exists \mathbf{w}$ such that $\forall t$ $\quad y_t(\mathbf{w} \cdot \mathbf{x}_t) \geq \gamma > 0$

Called the *functional margin* with respect to the training set



Equivalently, for $y_t = +1$,

$$w \cdot x_t \geq \gamma$$

and for $y_t = -1$,

$$w \cdot x_t \leq -\gamma$$

Mistake Bound for Perceptron

- Assume the data set *D* is linearly separable with *geometric* margin γ, i.e.,

$$\exists w^* \text{ s.t. } \|w^*\|_2 = 1 \text{ and } \forall t,\ y_t(w^* \cdot x_t) \geq \gamma$$

- Assume $\|x_t\|_2 \leq R, \forall t$
- <u>Theorem</u>: The maximum number of mistakes made by the perceptron algorithm is bounded by R^2/γ^2

Proof by induction

Assume we make a mistake for $(\mathbf{x}_t, y_t)$

$$\|\mathbf{w}_{t+1}\|_2^2 = \|\mathbf{w}_t + y_t\mathbf{x}_t\|^2 \leq \|\mathbf{w}_t\|_2^2 + R^2$$

$$\mathbf{w}_{t+1}^\top\mathbf{w}^* = \mathbf{w}_t^\top\mathbf{w}^* + y_t\mathbf{x}_t^\top\mathbf{w}^* \geq \mathbf{w}_t^\top\mathbf{w}^* + \gamma$$

$$\|\mathbf{w}_t\|_2^2 \leq M_t \cdot R^2 \qquad \mathbf{w}_t^\top\mathbf{w}^* \geq M_t \cdot \gamma$$

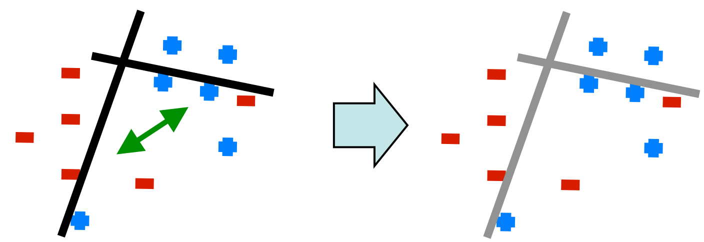
$$M_t \leq \frac{R^2}{\gamma^2}$$

(full proof given on board)

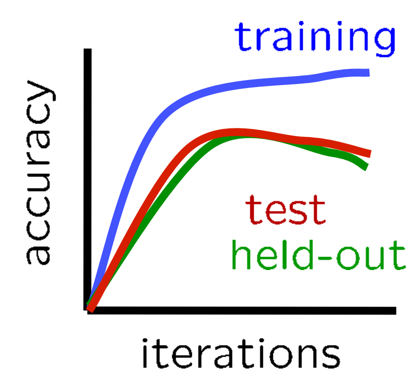
[Slide by Rong Jin]

Problems with the perceptron algorithm

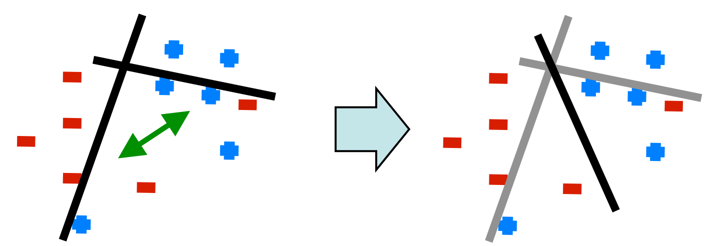
- If the data isn't linearly separable, no guarantees of convergence or training accuracy



- Even if the training data is linearly separable, perceptron can overfit



- **Averaged** perceptron is an algorithmic modification that helps with both issues
  - Averages the weight vectors across all iterations



ML Methodology

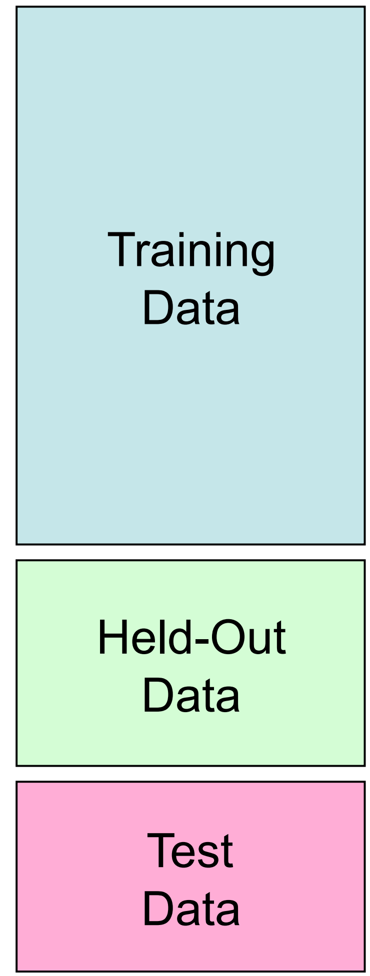
- Data: labeled instances, e.g. emails marked spam/ham
  - Training set
  - Held out set (sometimes call Validation set)
  - Test set

  Randomly allocate to these three, e.g. 60/20/20

- Features: attribute-value pairs which characterize each x

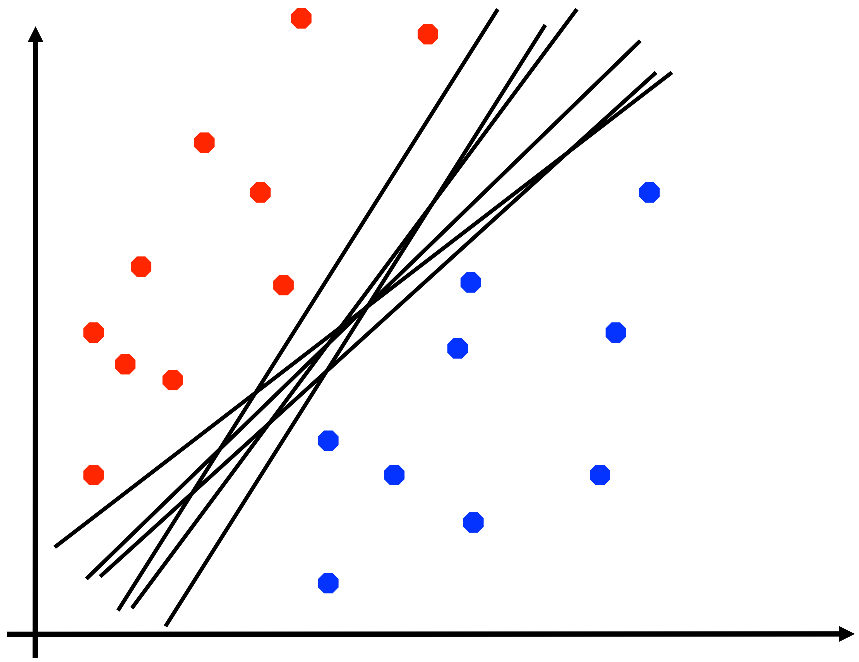
- Experimentation cycle
  - Select a hypothesis f
    (Tune hyperparameters on held-out or *validation* set)
  - Compute accuracy of test set
  - Very important: never "peek" at the test set!

- Evaluation
  - Accuracy: fraction of instances predicted correctly



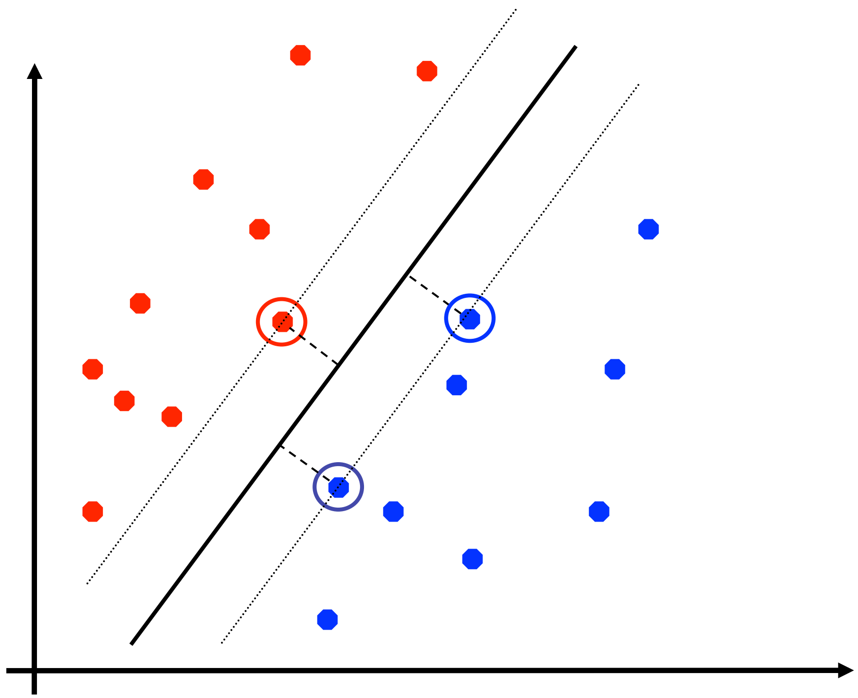
Linear Separators

- Which of these linear separators is optimal?



Next week: Support Vector Machines

- SVMs (Vapnik, 1990’s) choose the linear separator with the **largest margin**



- Good according to intuition, theory, practice