Introduction to Learning
Lecture 2

David Sontag
New York University

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

Supervised Learning: find f

- Given: Training set $\{(x_i, y_i) \mid i = 1 \ldots N\}$
- Find: A good approximation to $f : X \rightarrow Y$

Examples: what are X and Y?

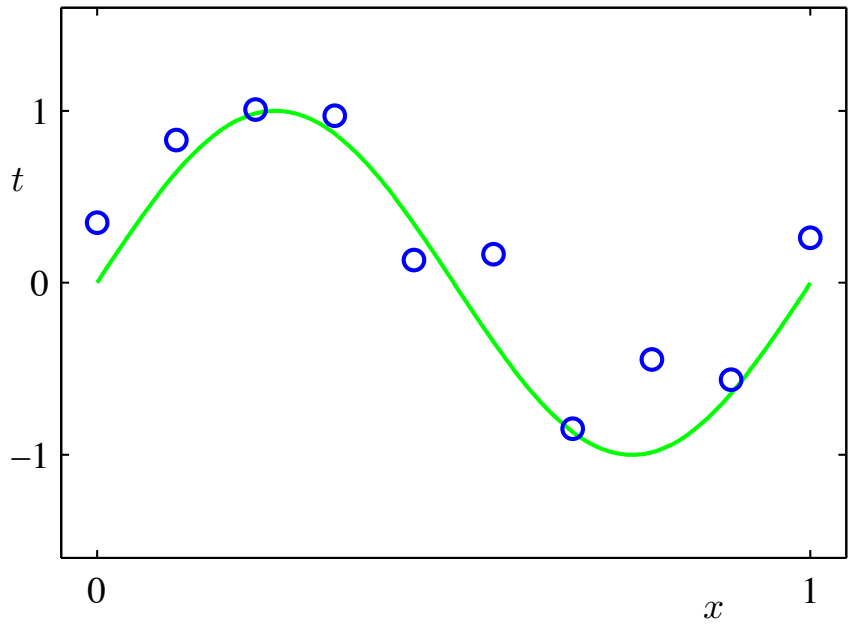
- Spam Detection
  - Map email to {Spam, Not Spam}
- Digit recognition
  - Map pixels to {0,1,2,3,4,5,6,7,8,9}
- Stock Prediction
  - Map new, historic prices, etc. to $\Re$ (the real numbers)

Example regression problem

- Consider a simple, regression dataset:
  - $f : X \rightarrow Y$
  - $X = \Re$
  - $Y = \Re$

- Question 1: How should we pick the *hypothesis space*, the set of possible functions f?

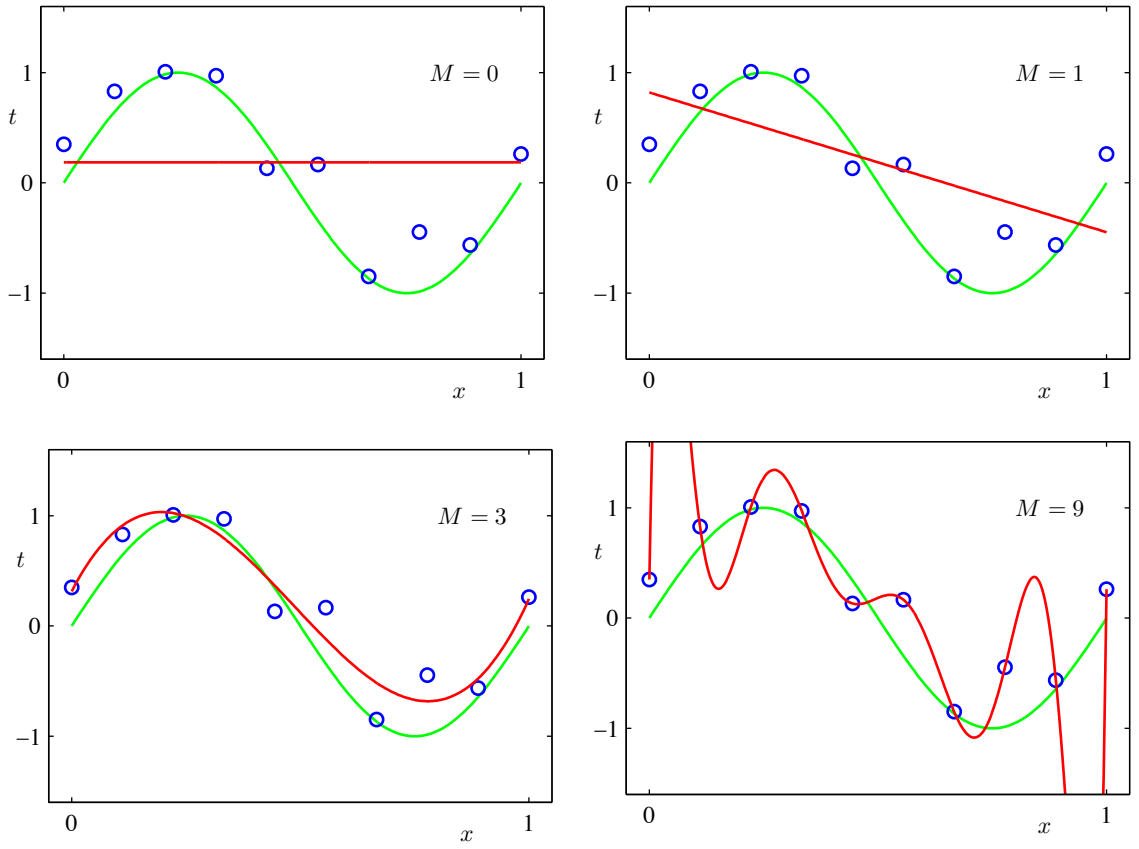
Dataset: 10 points generated from a sin function, with noise



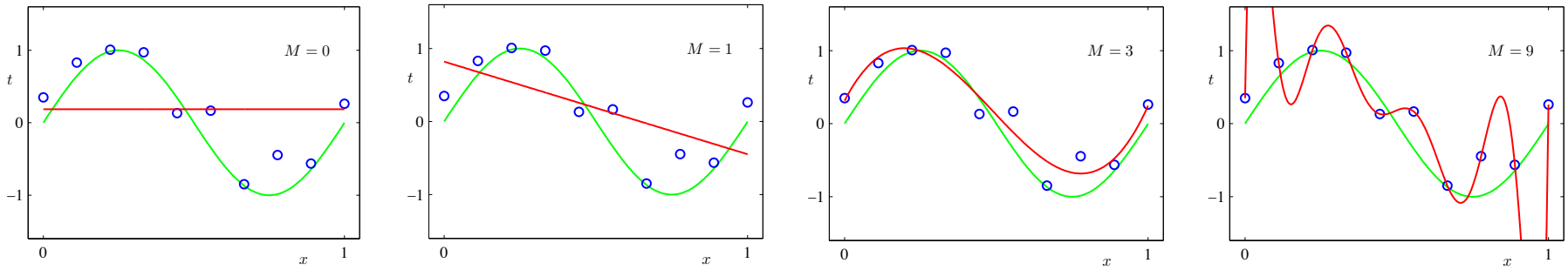
[Bishop]

Hypo. Space: Degree-N Polynomials

- Infinitely many hypotheses
- None / Infinitely many are consistent with our dataset
- Which one is **best**?



Hypo. Space: Degree-N Polynomials



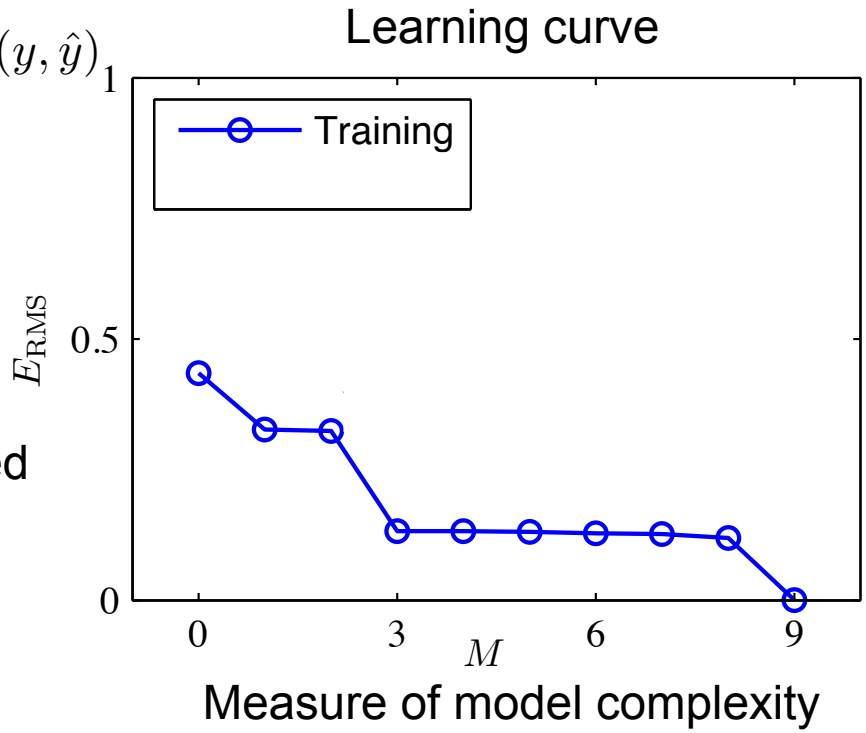
We measure error using a *loss function* $L(y, \hat{y})$

For regression, a common choice is squared loss:

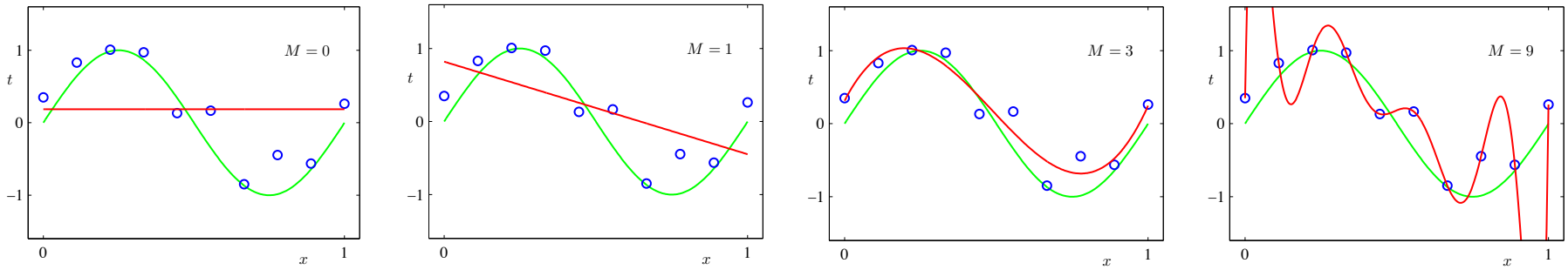
$$L(y_i, f(x_i)) = (y_i - f(x_i))^2$$

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

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



Hypo. Space: Degree-N Polynomials



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

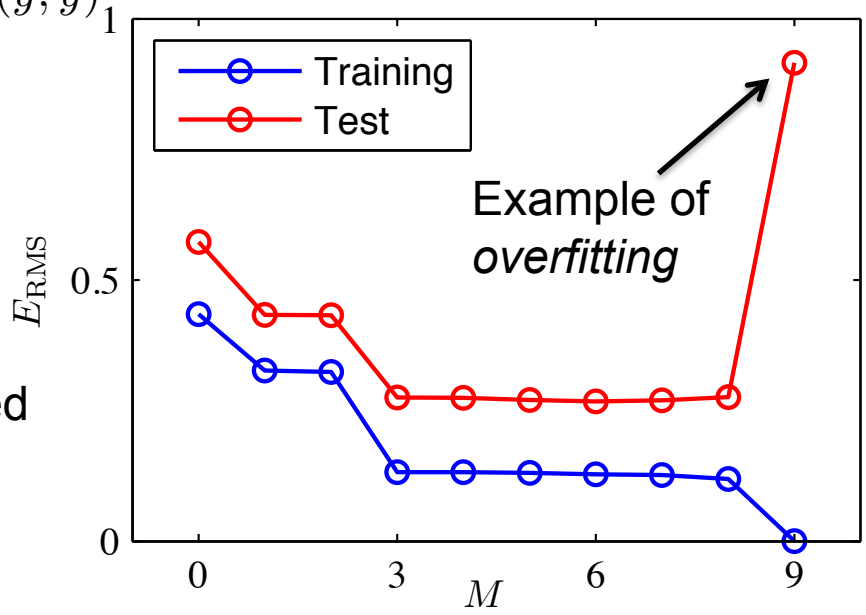
For regression, a common choice is squared loss:

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

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



Measure of model complexity

Training and test performance

- Assume that each training *and* test example-label pair (**x**, y) is drawn *independently at random* from the *same* (but unknown) population of examples and labels
- Represent this population as a probability distribution *p*(**x**, y), so that:

$$(\mathbf{x}_i, y_i) \sim p(\mathbf{x}, y)$$

- Empirical (training) loss = $\frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i))$ Also called *empirical risk,* $\hat{R}(f, D_N)$

- Expected (test) loss = $E_{(\mathbf{x},y)\sim p}\left\{L(y, f(\mathbf{x})\right\}$ Also called *risk,* $R(f)$

- Ideally, learning chooses the hypothesis that minimize the risk – but this is impossible to compute!
- Empirical risk is an unbiased estimate of the risk (by linearity of expectation)
- The principle of empirical risk minimization: $f^*(D_N) = \arg\min_{f\in\mathcal{F}} \hat{R}(f, D_N)$

Key Issues in Machine Learning

- How do we choose a hypothesis space?
  - Often we use **prior knowledge** to guide this choice
  - The ability to answer to the next two questions also affects choice

- How can we gauge the accuracy of a hypothesis on unseen testing data?
  - The previous example showed that choosing the hypothesis which simply minimizes training set error (i.e., empirical risk minimization) **is not optimal**
  - This question is the main topic of **learning theory**

- How do we find the best hypothesis?
  - This is an **algorithmic** question, at the intersection of computer science and optimization research

Occam's Razor Principle

- William of Occam: Monk living in the 14th century
- Principle of parsimony:

"One should not increase, beyond what is necessary, the number of entities required to explain anything"

- When many solutions are available for a given problem, we should select the simplest one
- But what do we mean by simple?
- We will use prior knowledge of the problem to solve to define what is a simple solution

Example of a prior: smoothness

[Samy Bengio]

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.

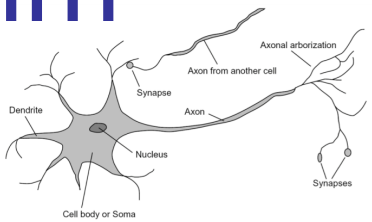
First, I must solicit your confidence in this transaction, this is by virture of its nature as being utterly confidencial and top secret. …

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

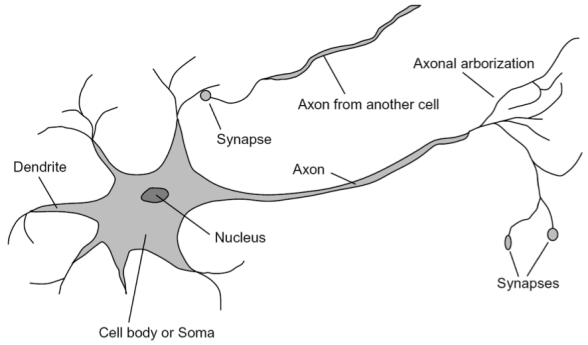
- 1969: Minksky & Papert book shows perceptrons limited to *linearly separable* data, and Rosenblatt dies in boating accident

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

[William Cohen]

Linear Classifiers

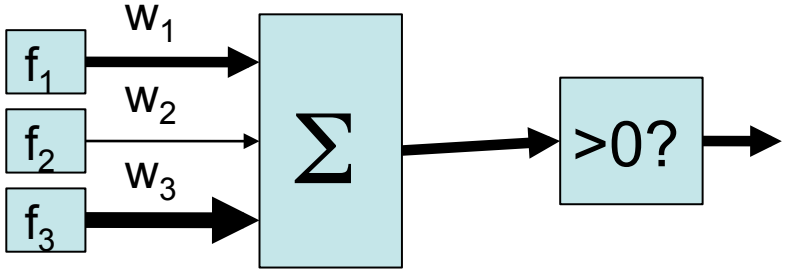
- Inputs are feature values
- Each feature has a weight
- Sum is the activation



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
...
```

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

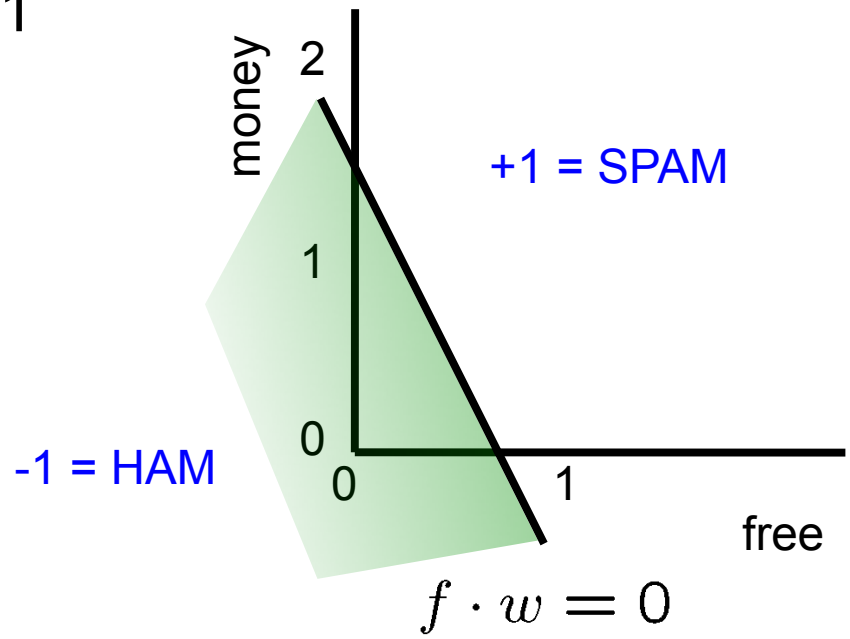
w.f(x) > 0 ➔ SPAM!!!

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
...
```



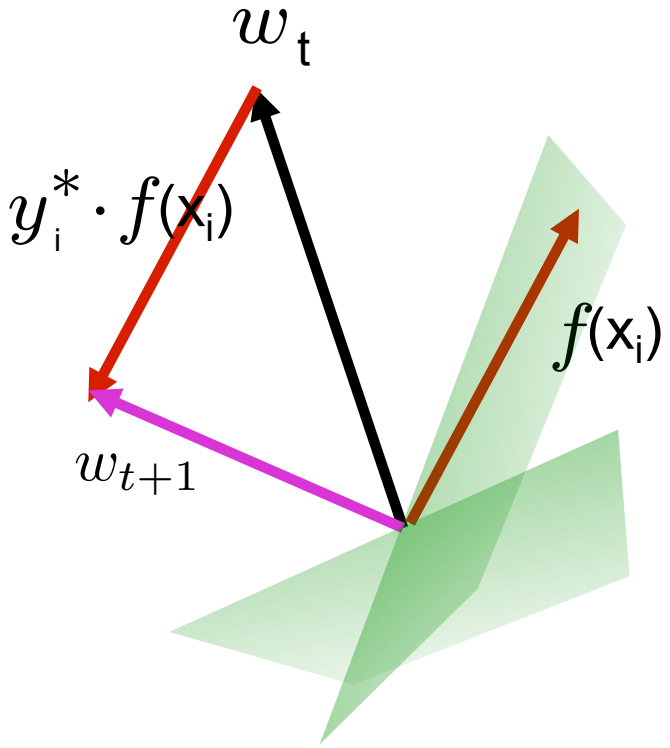
The perceptron algorithm

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

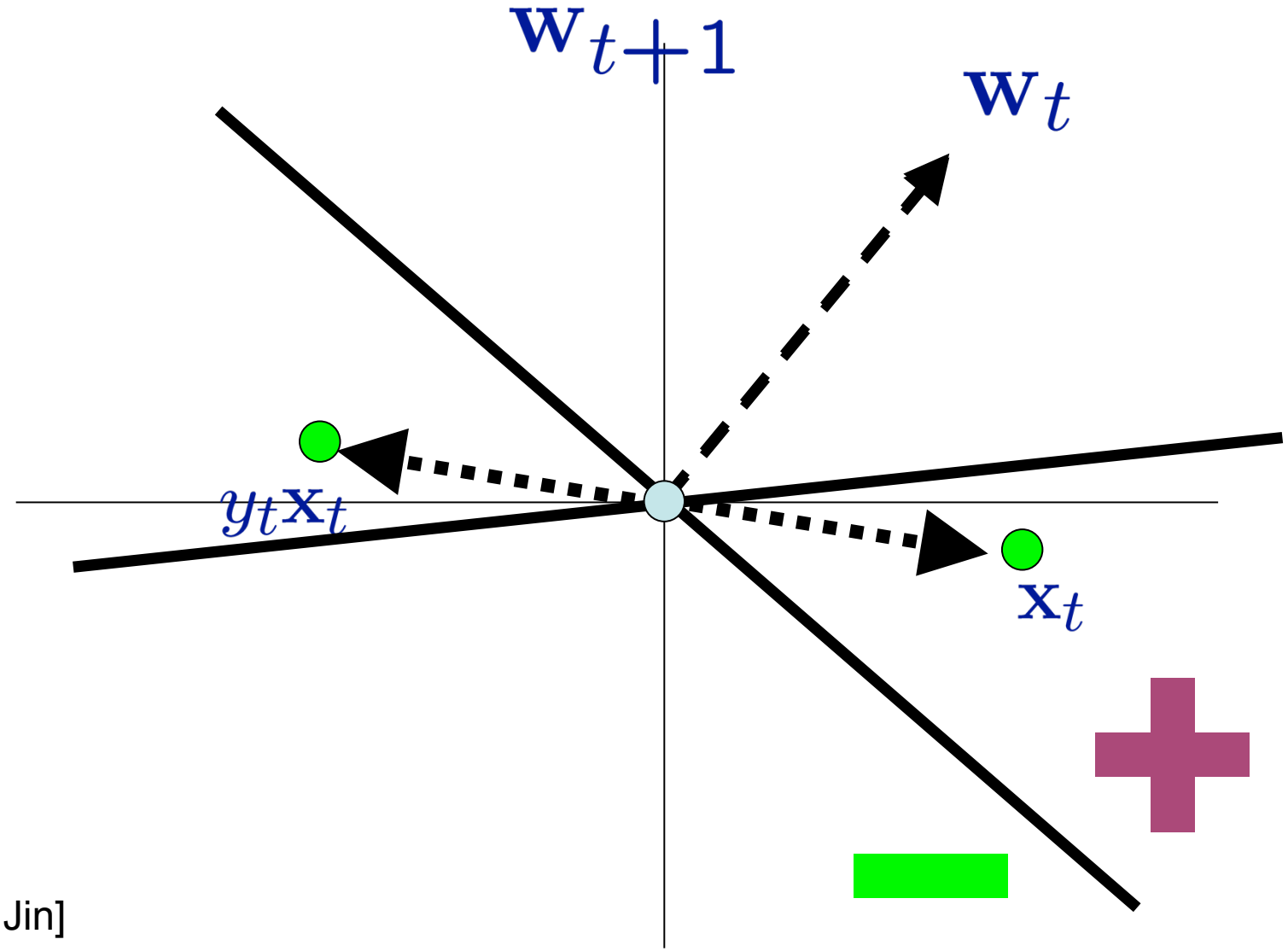
$$y_i = \begin{cases} +1 & \text{if } w \cdot f(x_i) \geq 0 \\ -1 & \text{if } w \cdot f(x_i) < 0 \end{cases}$$

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

$$w = w + y_i^* f(x_i)$$



Geometrical Interpretation

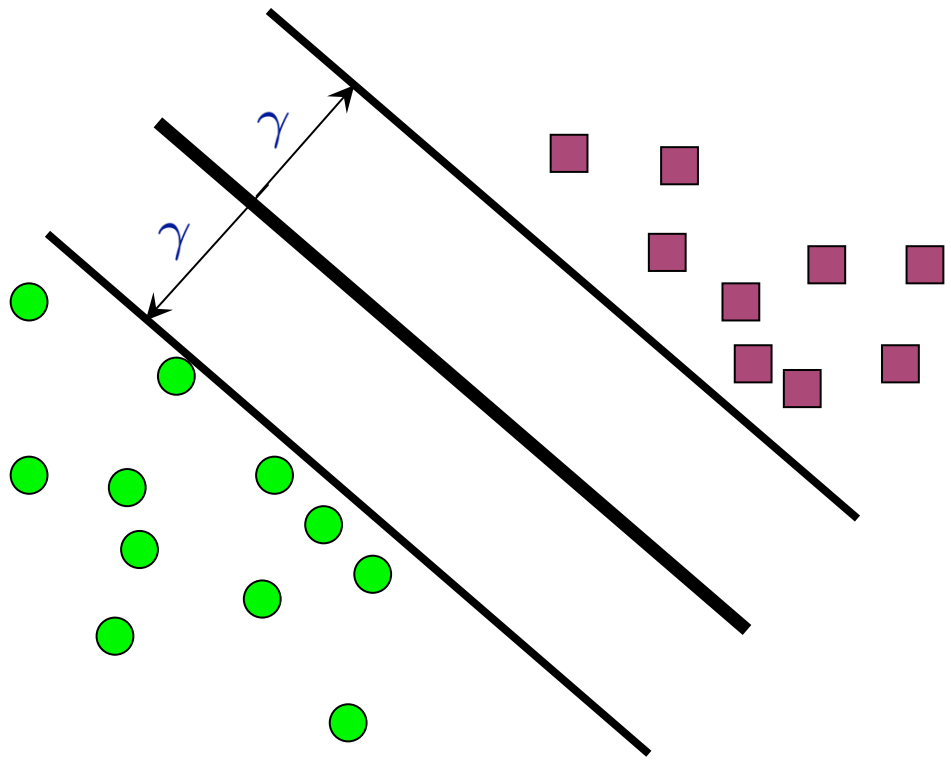


[Rong Jin]

Def: Linearly separable data

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

Called the *margin*



[Rong Jin]

Mistake Bound: Separable Case

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

$$\exists \mathbf{w}^*, |\mathbf{w}^*|_2 = 1,\ \forall t, y_t \mathbf{x}_t^\top \mathbf{w}^* \geq \gamma$$

- Assume $|\mathbf{x}_t|_2 \leq R, \forall t$
- Theorem: The maximum number of mistakes made by the perceptron algorithm is bounded by R^2/γ^2

[Rong Jin]

Proof by induction

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

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

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

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

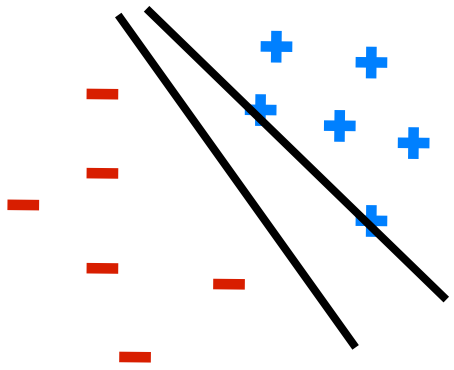
$$M_t \leq \frac{R^2}{\gamma^2}$$

[Rong Jin]

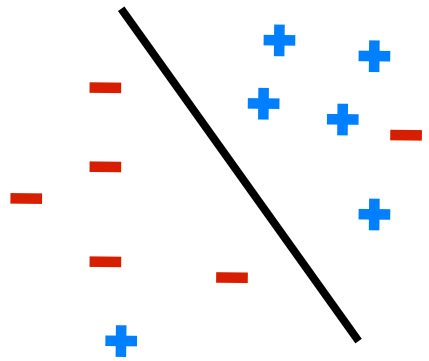
Properties of the perceptron algortihm

- Separability: some parameters get the training set perfectly correct
- Convergence: if the training is **linearly separable**, perceptron will eventually converge

Separable

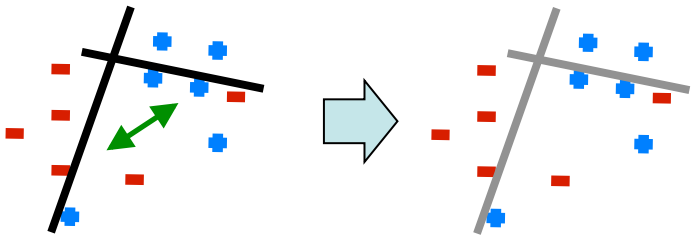


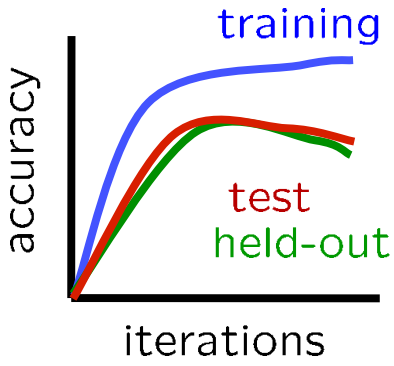
Non-Separable

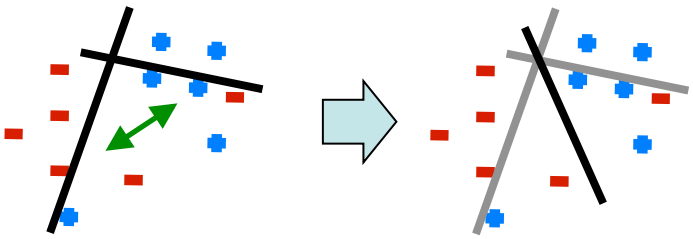


Problems with the perceptron algorithm

- **Noise**: if the data isn't linearly separable, no guarantees of convergence or accuracy
- Frequently the training data ***is*** linearly separable! Why?
  - For example, when the number of examples is much smaller than the number of features
  - Perceptron can significantly **overfit** the data
- **Averaged** perceptron is an algorithmic modification that helps with both issues
  - Averages the weight vectors across all iterations

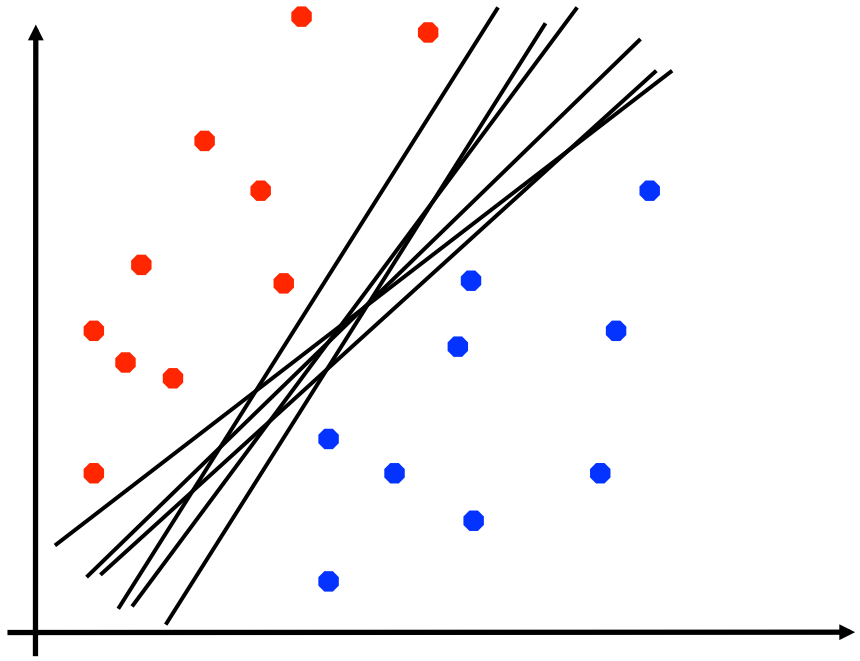






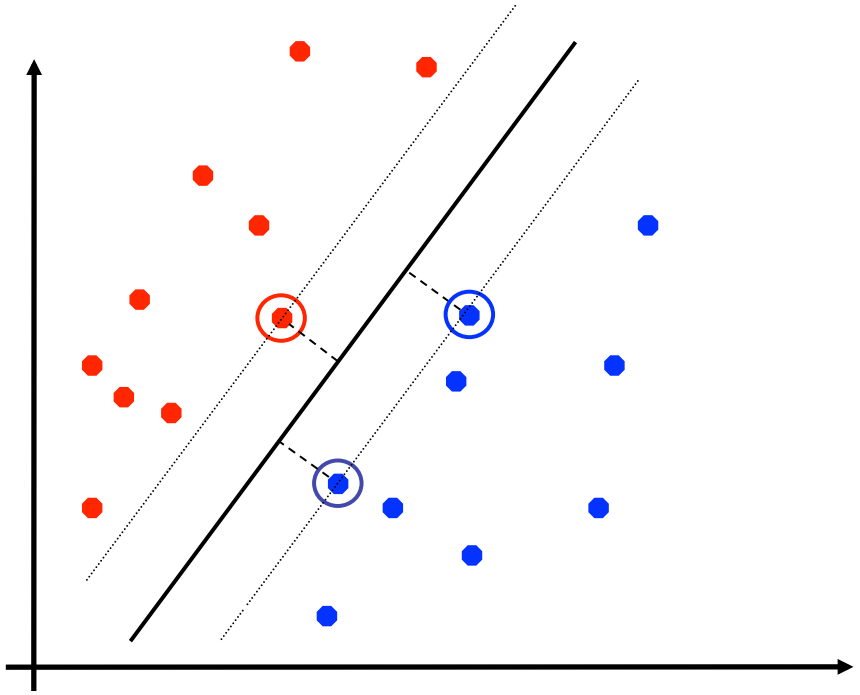
Linear Separators

- 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