COMP4901K/Math4824B
Machine Learning for Natural Language Processing

Lecture 8: Perceptron, Error-Driven Classification

Instructor: Yangqiu Song

Slides credits: Dan Roth, Barbara Rosario

Today

- Algorithms for Discriminative Classification
- Binary classification
  - Perceptron

Binary Classification: examples

- Spam filtering (spam, not spam)
- Customer service message classification (urgent vs. not urgent)
- Information retrieval (relevant, not relevant)
- Sentiment classification (positive, negative)

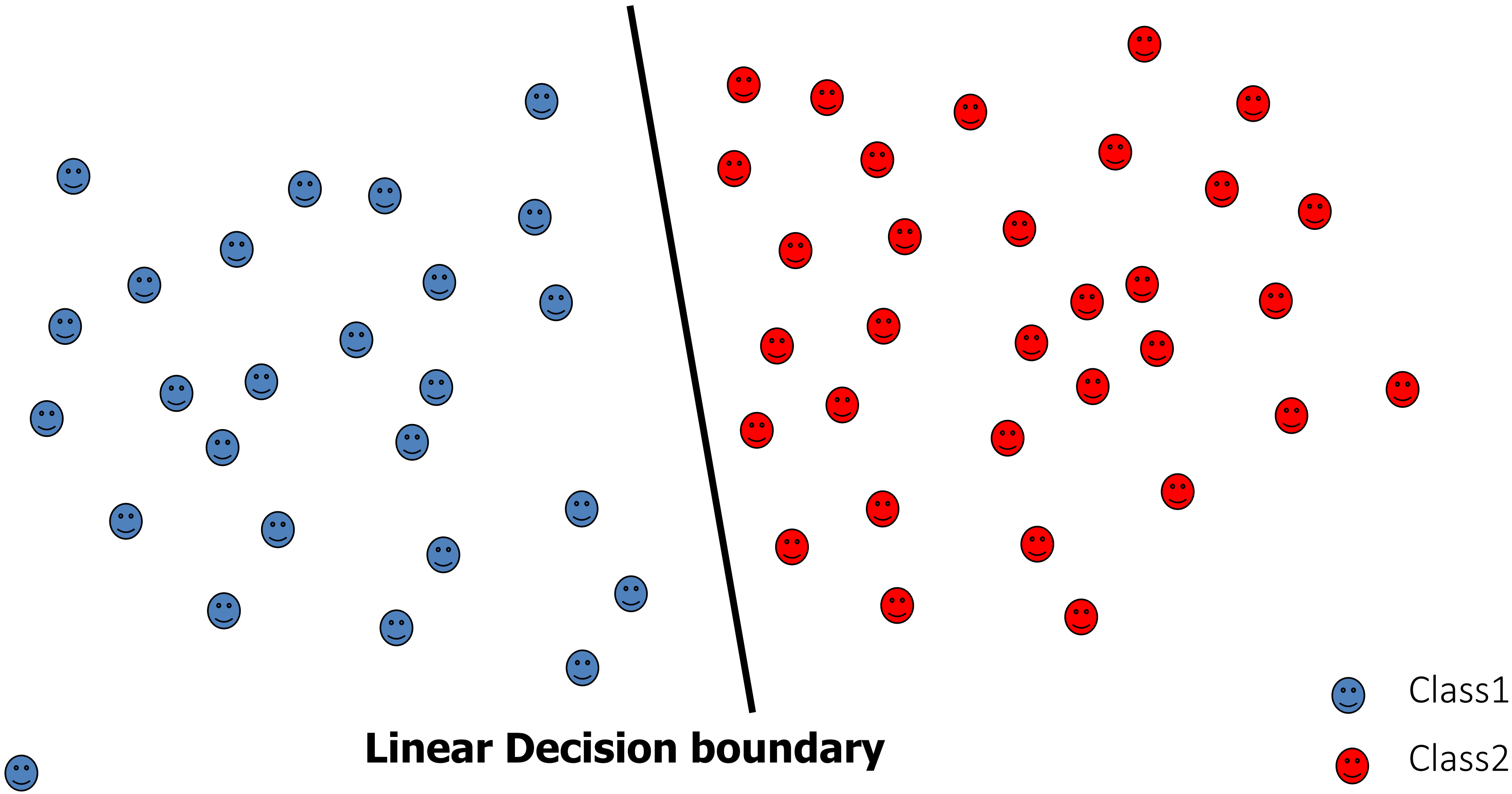
Binary Classification

- Given: some data items that belong to a positive (+1) or a negative (-1) class
- Task: Train the classifier and predict the class for a new data item
- Geometrically: find a separator

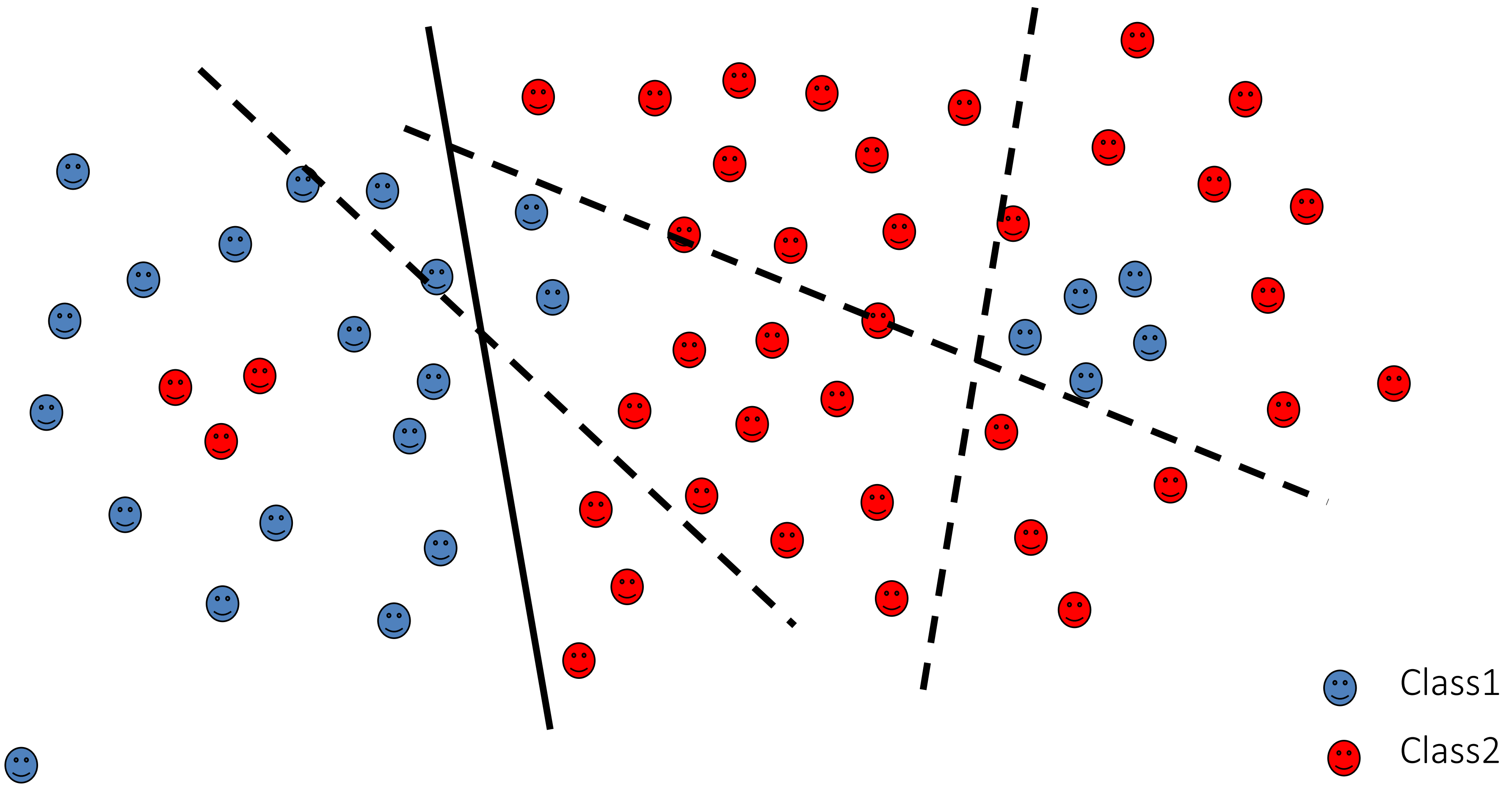
Linear versus Non Linear algorithms

- **Linearly separable data**: if all the data points can be correctly classified by a linear (hyperplanar) decision boundary

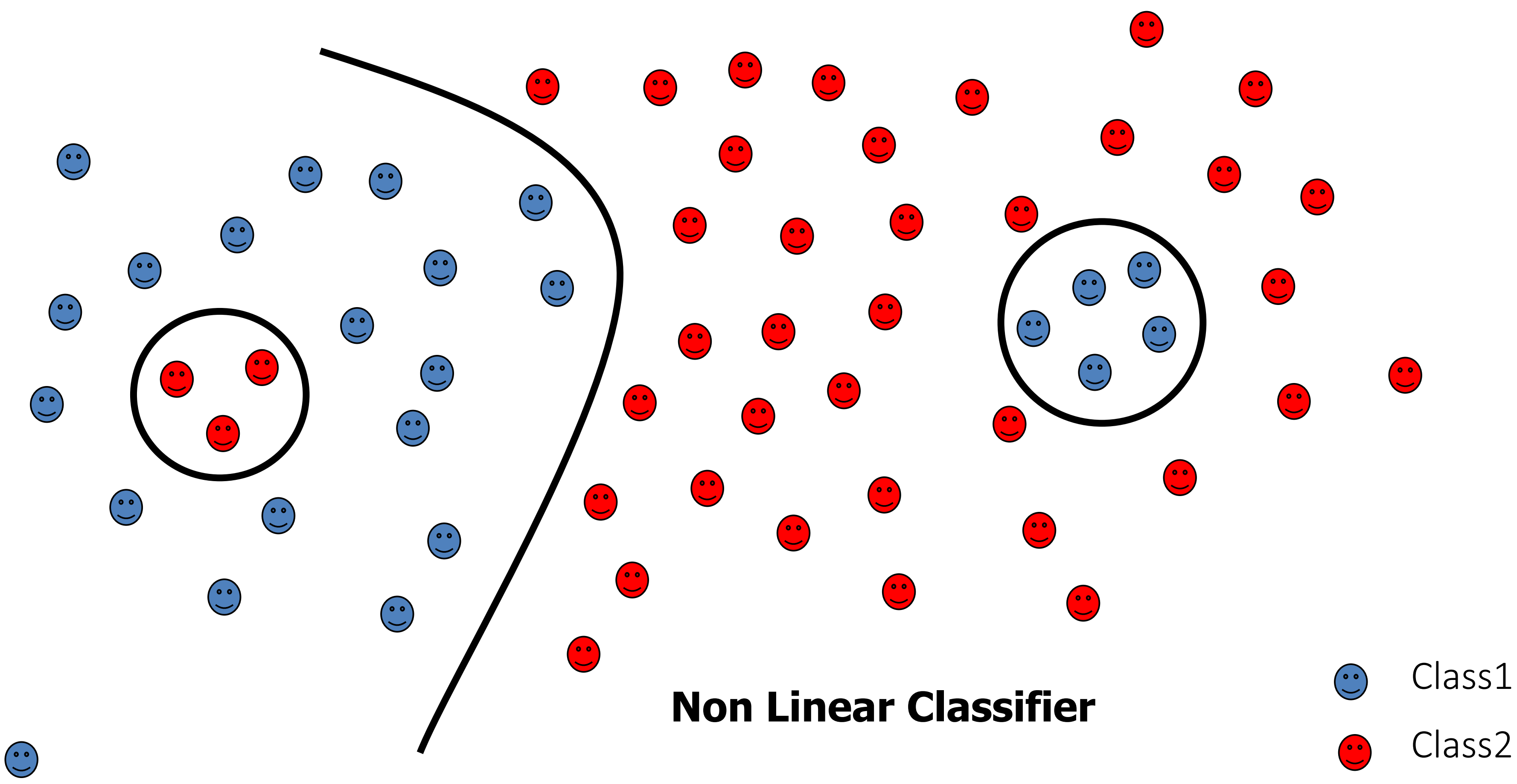
Linearly separable data



Non linearly separable data



Non linearly separable data



Linear versus Non Linear algorithms

- Linear or Non linear separable data?
  - We can find out only empirically
- Linear algorithms (algorithms that find a linear decision boundary)
  - When we think the data is linearly separable
  - Advantages
    - Simpler, less parameters
  - Disadvantages
    - Data (like for texts) is usually not linearly separable
  - Examples: Perceptron, Winnow, SVM

Linear versus Non Linear algorithms

- Non Linear
  - When the data is non linearly separable
  - Advantages
    - More accurate
  - Disadvantages
    - More complicated, more parameters
  - Example: Deep learning

Simple linear algorithms

- Perceptron algorithm
  - Linear
  - Binary classification
  - Online (process data sequentially, one data point at the time)
  - Mistake driven
  - Simple single layer Neural Networks

Linear binary classification

- Data: $\{(\boldsymbol{x}_i, y_i)\}_{i=1\ldots n}$
  - $\boldsymbol{x}$ in R^d (x is a vector in d-dimensional space)

    → feature vector
  - y in $\{-1,+1\}$

    → label (class, category)

- Question:
  - Design a linear decision boundary: $\boldsymbol{w}^T\boldsymbol{x} + b$ (equation of hyperplane) such that the classification rule associated with it has minimal probability of error
  - classification rule
    - $y = \text{sign}(\ \boldsymbol{w}^T\boldsymbol{x} + b\ )$ which means:
    - if $\boldsymbol{w}^T\boldsymbol{x} + b > 0$ then $y = +1$
    - if $\boldsymbol{w}^T\boldsymbol{x} + b < 0$ then $y = -1$

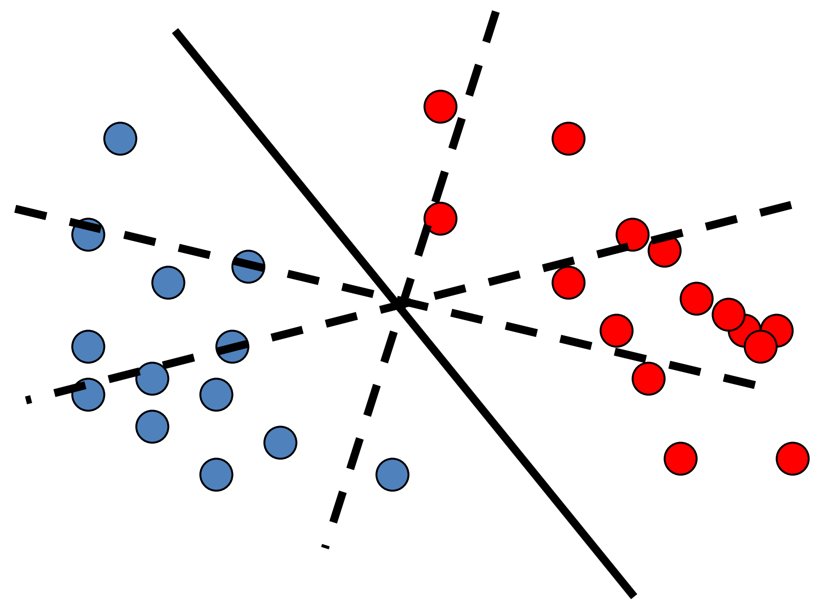
Linear binary classification

- Find a good hyperplane

  ($\boldsymbol{w}$,b) in R^{d+1}

  that correctly classifies data points as much as possible

- In online fashion: one data point at the time, update weights as necessary



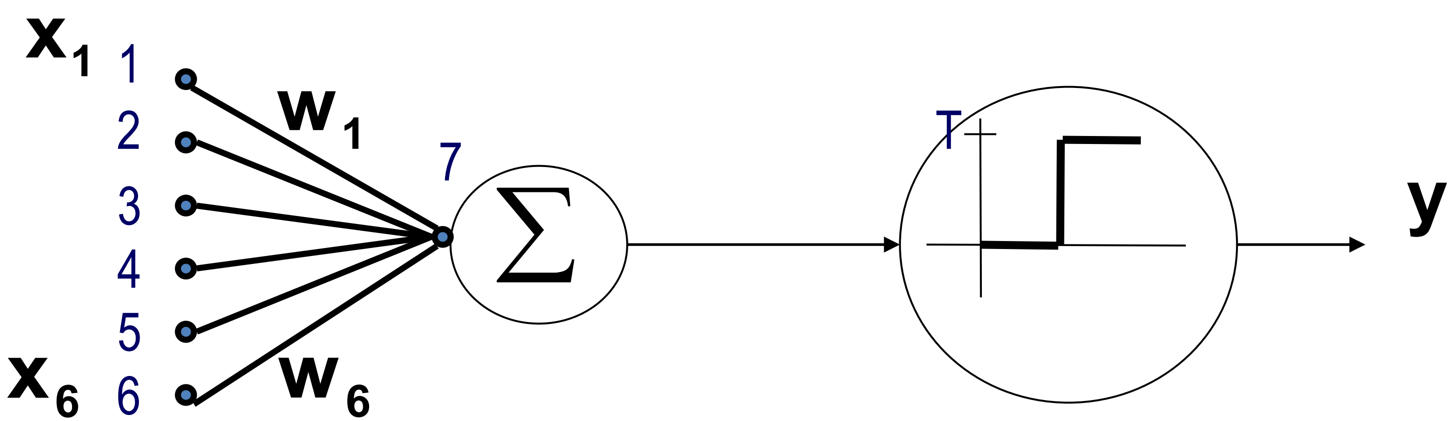
$\boldsymbol{w}^T\boldsymbol{x} + b=0$

Classification Rule:

$y = \text{sign}(\boldsymbol{w}^T\boldsymbol{x} + b)$

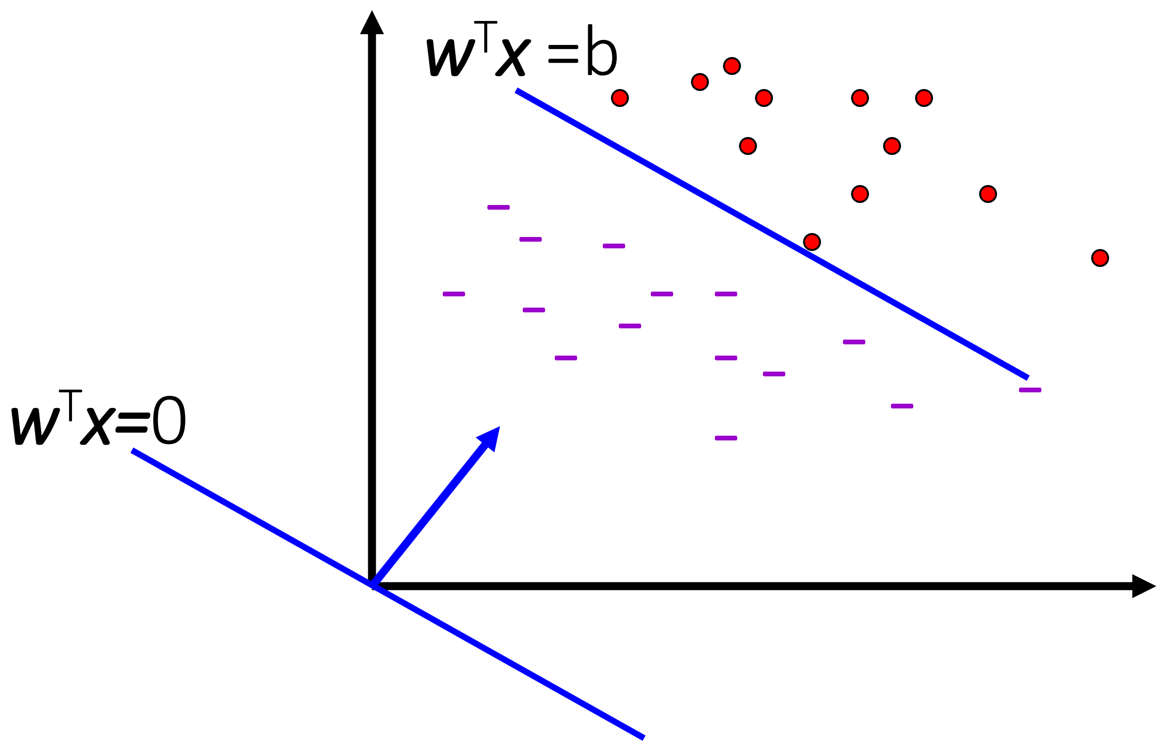
Perceptron learning rule

- On-line, mistake driven algorithm.
- Rosenblatt (1959) suggested that when a target output value is provided for a single neuron with fixed input, it can incrementally change weights and learn to produce the output using the Perceptron learning rule
- (Perceptron == Linear Threshold Unit)



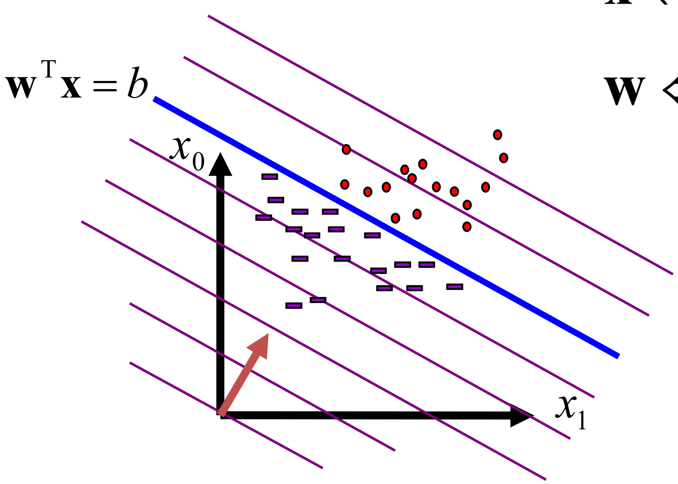
Perceptron learning rule

- We learn f:$\mathbf{X} \rightarrow \{-1,+1\}$ represented as $f = \mathrm{sgn}\{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}\}$
- Where $\mathbf{X} = \{0,1\}^n$ or $\mathbf{X} = R^n$ and $\boldsymbol{w} \in R^n$
- Given Labeled examples: $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots (\boldsymbol{x}_N, y_N)\}$



Footnote About the Threshold

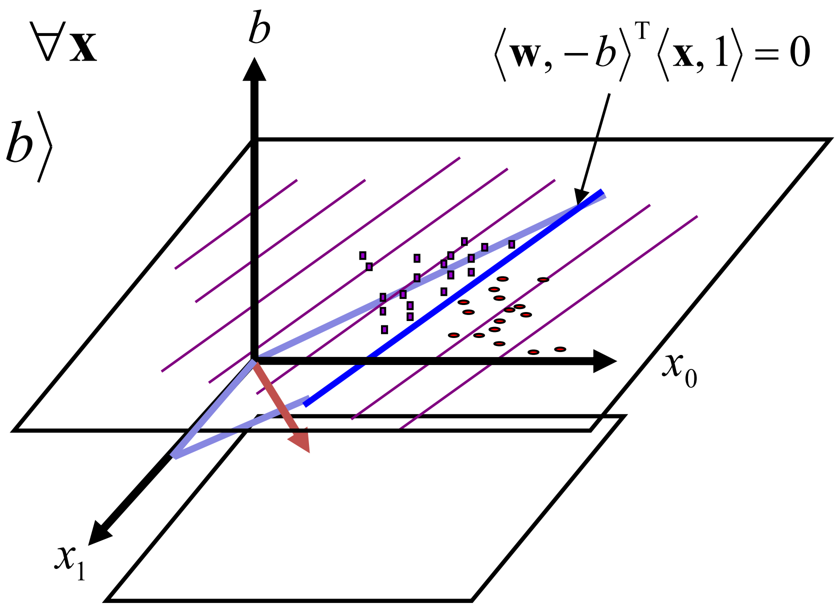
- On previous slide, Perceptron has no threshold
- But we don’t lose generality:



$$\mathbf{x} \Leftrightarrow \langle \mathbf{x}, 1 \rangle \quad \forall \mathbf{x}$$

$$\mathbf{w} \Leftrightarrow \langle \mathbf{w}, -b \rangle$$

$$\Leftrightarrow$$



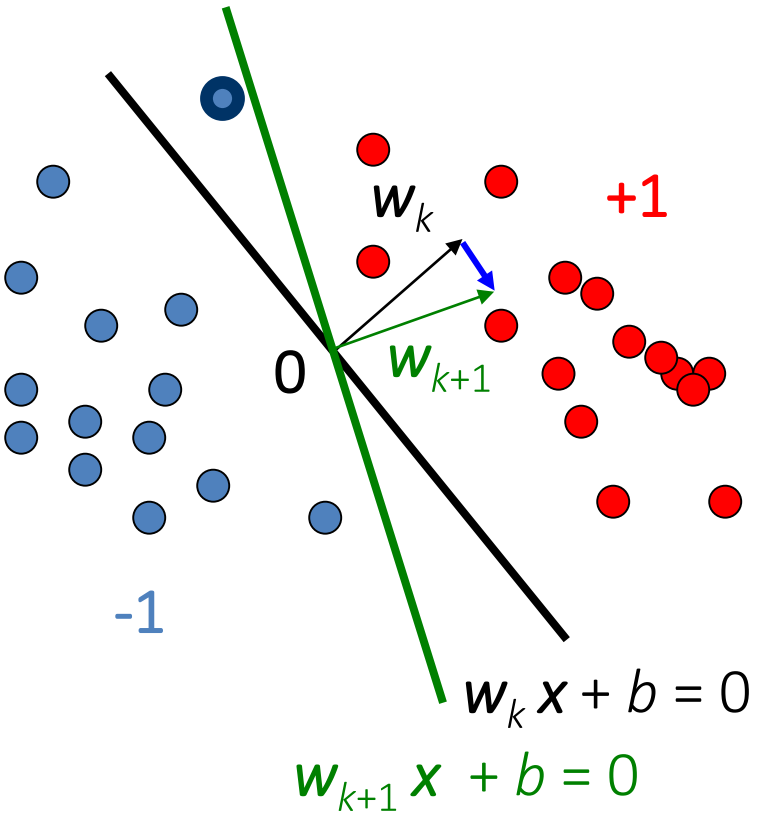
Perceptron algorithm

- Initialize: $\boldsymbol{w}_1 = \mathbf{0}$ in R^n
- Updating rule
- For each data point $\boldsymbol{x}$
  - Predict the label $y' = \text{sgn}\{\boldsymbol{w}^T\boldsymbol{x}\}$
  - if $y' \mathrel{!}= y$, update the weight vector

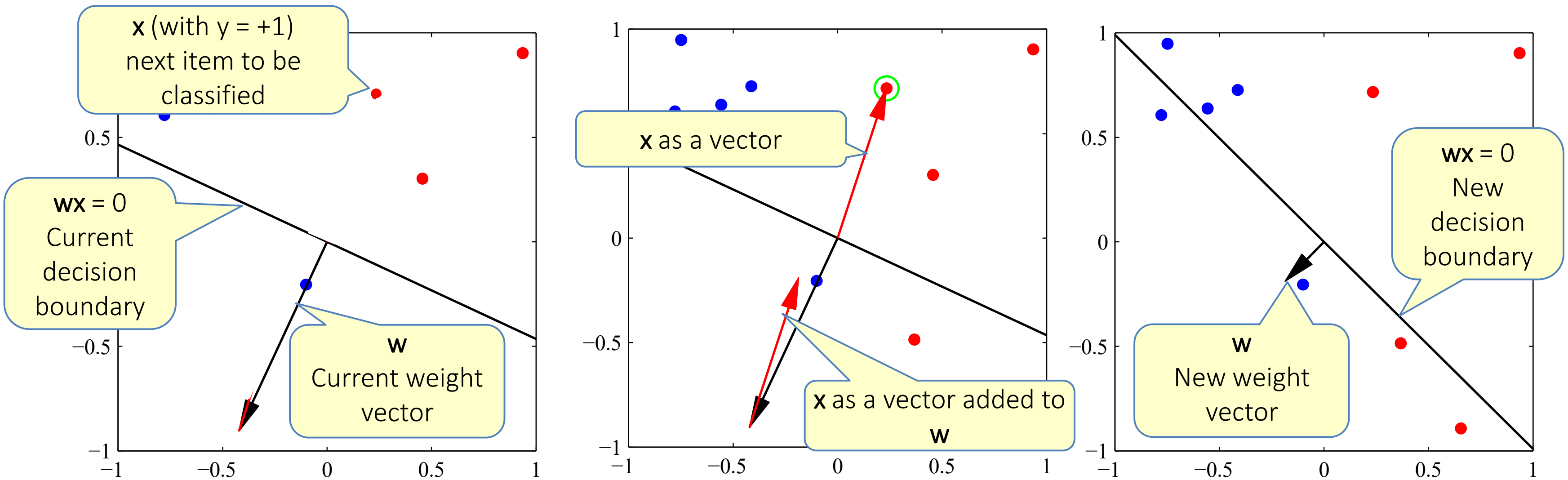
    $\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k + a\, y_i \boldsymbol{x}_i$

    (a - a constant, learning rate)
  - else

    $\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k$
- Function: $y' = \text{sgn}\{\boldsymbol{w}^T\boldsymbol{x}\}$
  - if $\boldsymbol{w}^T\boldsymbol{x} > 0$ return +1
  - else return -1

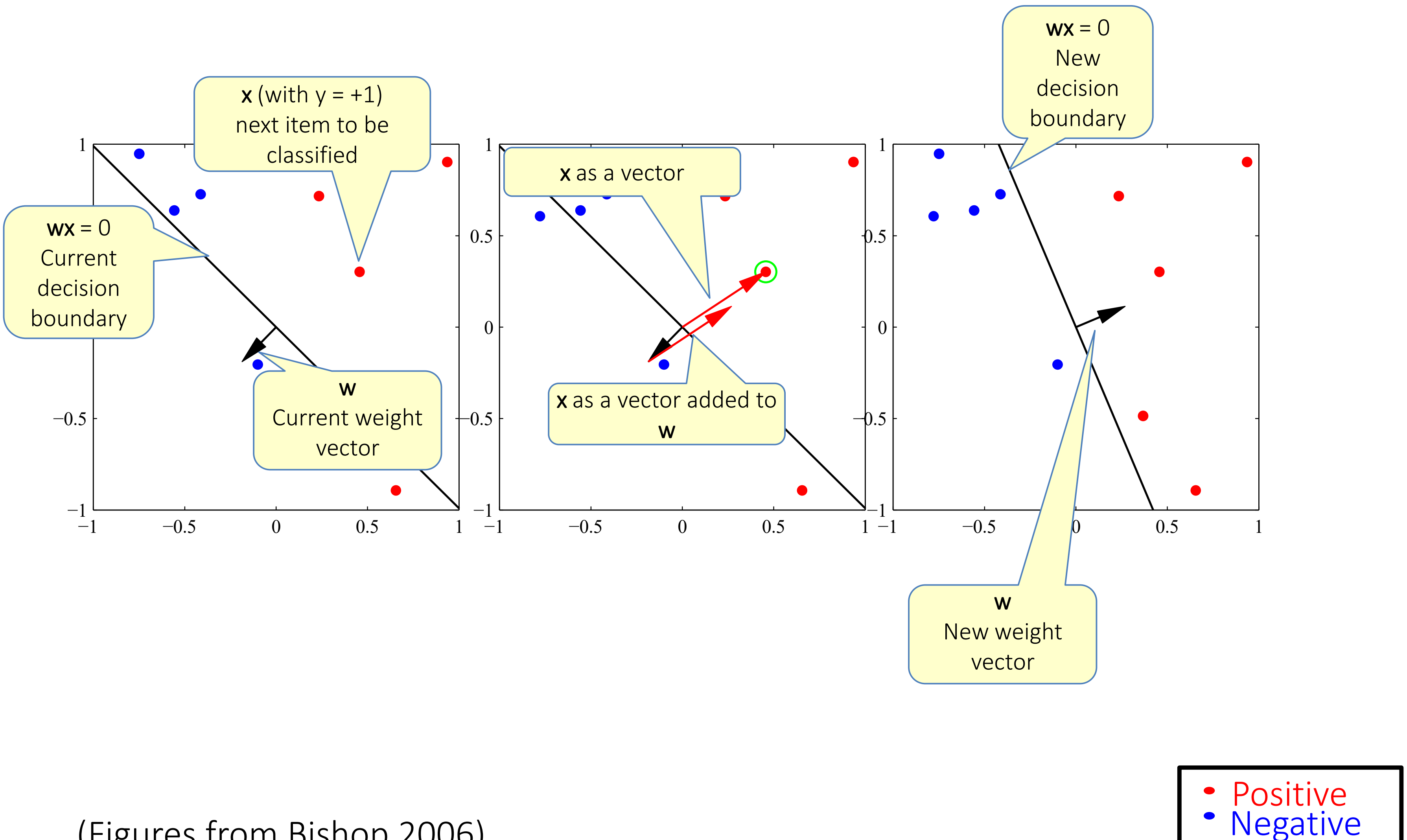


Perceptron in action



(Figures from Bishop 2006)

Perceptron in action



(Figures from Bishop 2006)

Perceptron learning rule

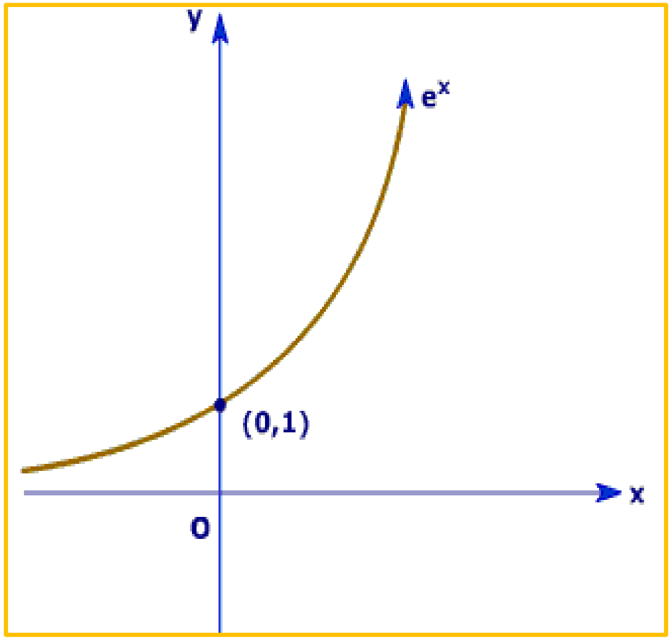
- If x is Boolean, only weights of active features are updated

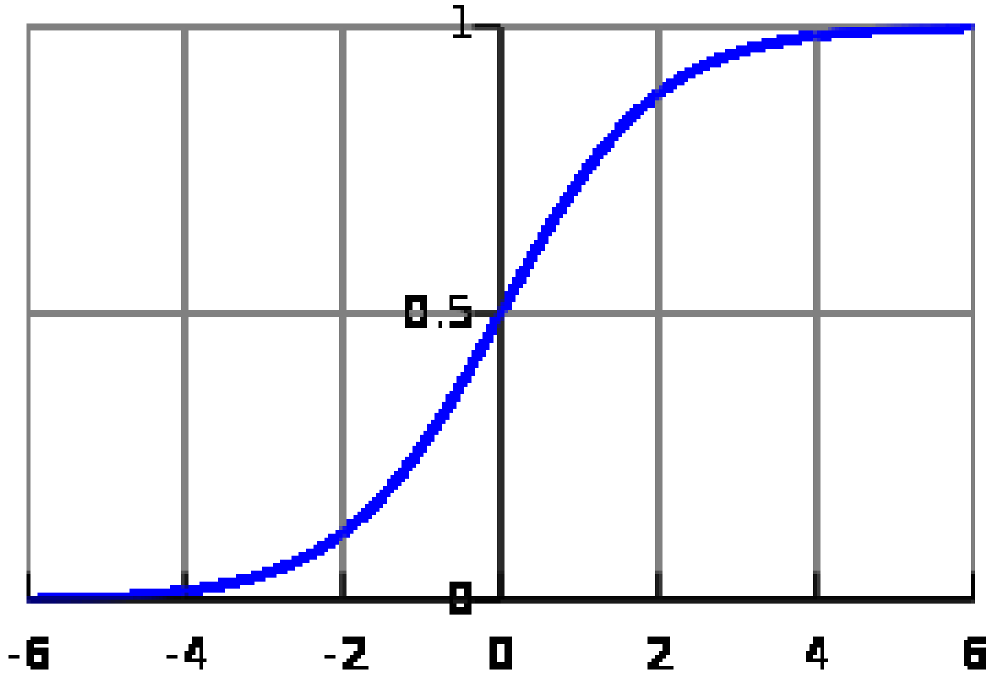
$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{x}$$

$$\begin{pmatrix} w^{(1)} + 1 \\ w^{(2)} \\ w^{(3)} - 1 \end{pmatrix} = \begin{pmatrix} w^{(1)} \\ w^{(2)} \\ w^{(3)} \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$$

Perceptron learning rule

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} > \mathbf{0} \text{ is equivalent to } \frac{\mathbf{1}}{\mathbf{1} + \mathbf{exp\{-(w}^{\mathrm{T}}\mathbf{x)\}}} > \mathbf{1/2}$$





Perceptron Learnability

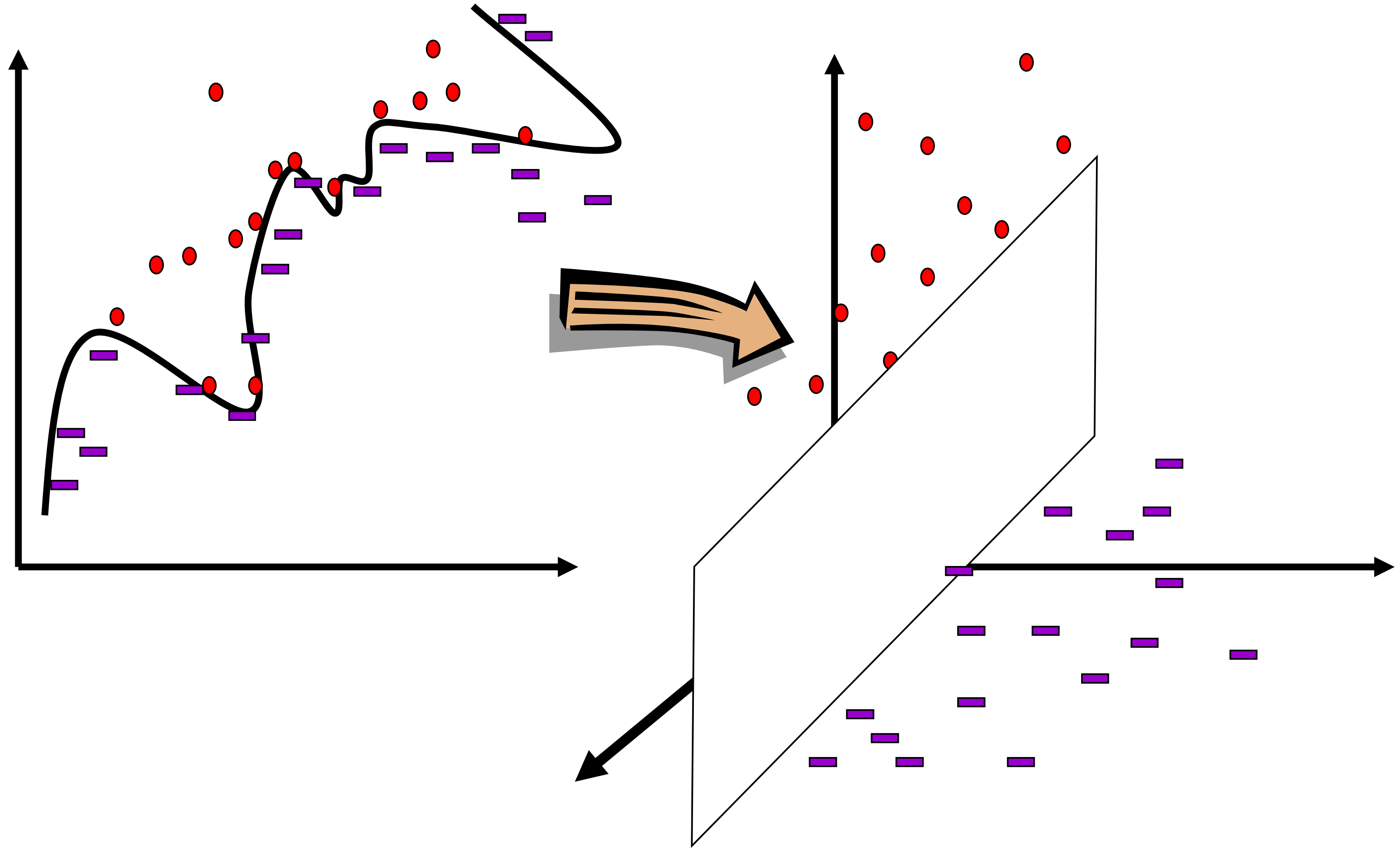
- Obviously can’t learn what it can’t represent Only linearly separable functions

- Minsky and Papert (1969) wrote an influential book demonstrating Perceptron’s representational limitations
  - Parity functions can’t be learned (XOR)
  - In vision, if patterns are represented with local features, can’t represent symmetry, connectivity

- Research on Neural Networks stopped for years

- Rosenblatt himself (1959) asked,

  - *“What pattern recognition problems can be transformed so as to become linearly separable?”*



Perceptron Convergence

- Perceptron Convergence Theorem:
  - If there exist a set of weights that are consistent with the data (i.e., the data is linearly separable), the perceptron learning algorithm will converge
    - How long would it take to converge ?

- Perceptron Cycling Theorem:
  - If the training data is not linearly separable the perceptron learning algorithm will eventually repeat the same set of weights and therefore enter an infinite loop.
    - How to provide robustness, more expressivity ?

Perceptron-Mistake Bound

- Assume that a weight vector $\boldsymbol{w} \in \mathrm{R}^d$. Upon receiving an example $\boldsymbol{x} \in \mathrm{R}^d$, we predict according to a linear threshold function.
- **Theorem [Novikoff,1963]**
  - Let $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ be a sequence of labeled examples with $\boldsymbol{x}_i \in R^d$, $||\boldsymbol{x}|| \leq r$, and $y_i \in \{-1, +1\}$ for all i.
  - Let $\boldsymbol{u} \in \mathrm{R}^d$, $\gamma > 0$ be such that $||\boldsymbol{u}|| = 1$ and $y_i \cdot \boldsymbol{u}^{\mathrm{T}} \boldsymbol{x}_{\boldsymbol{i}} \geq \gamma$ for all i.
  - Perceptron makes at most $\frac{r^2}{\gamma^2}$ mistakes on this example sequence.
- Assumptions:
  - This theorem assumes that all examples are bounded by some r; for all $\boldsymbol{x}_{\boldsymbol{i}}$, find the largest one, and r is at least this size.
  - The theorem further assumes that there exists some $\boldsymbol{u}$ that separates the data.
  - Requiring that $||\boldsymbol{u}|| = 1$ is simply a constant that could be arbitrarily scaled.
  - Finally, the theorem assumes that there exists some γ such that the inequality is satisfied.
    - We refer to as the complexity parameter: γ is very large, finding a hyperplane is much easier

Proof

- Let $\boldsymbol{w_k}$ be the hypothesis before the kth mistake.
- Assume that the kth mistake occurs on the input example $(\boldsymbol{x}_i, y_i)$

$$\therefore y_i\left(\boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}_i\right) \le 0$$

$$\boldsymbol{w_{k+1}} = \boldsymbol{w_k} + y_i\boldsymbol{x}_i$$

$$\begin{aligned}\boldsymbol{w}_{k+1}^{\mathrm{T}}\boldsymbol{u} = \boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{u} + y_i\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{u} &\ge \boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{u} + \gamma \quad (\because y_i\boldsymbol{u}^{\mathrm{T}}\boldsymbol{x_i} \ge \gamma) \\ &\ge \boldsymbol{w}_{k-1}^{\mathrm{T}}\boldsymbol{u} + 2\gamma \\ &\cdots \\ &\ge k\gamma\end{aligned}$$

$$\begin{aligned}\left|\left|\boldsymbol{w}_{k+1}\right|\right|^2 &= \left|\left|\boldsymbol{w}_k\right|\right|^2 + 2y_i\boldsymbol{w}_k^T\boldsymbol{x}_i + \left|\left|\boldsymbol{x}_2\right|\right|^2 \\ &\le \left|\left|\boldsymbol{w}_k\right|\right|^2 + r^2 \quad (\because y_i\left(\boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{x}_i\right) \le 0) \\ &\cdots \\ &\le kr^2\end{aligned}$$

Therefore, $\sqrt{k}r \ge \left|\left|\boldsymbol{w}_{k+1}\right|\right| \ge \boldsymbol{w}_{k+1}^{\mathrm{T}}\boldsymbol{u} \ge k\gamma$

(Second inequality: $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{v} = \left|\left|\boldsymbol{u}\right|\right| \cdot \left|\left|\boldsymbol{v}\right|\right| \cdot cos(\boldsymbol{u}, \boldsymbol{v}) \le \left|\left|\boldsymbol{u}\right|\right| \cdot \left|\left|\boldsymbol{v}\right|\right| = ||\boldsymbol{v}||$ and $||\boldsymbol{u}|| = 1$)

$$\therefore k \le \frac{r^2}{\gamma^2}$$

- Further reading
  - https://www.cse.iitb.ac.in/~shivaram/teaching/old/cs344+386-s2017/resources/classnote-1.pdf

Practical Issues and Extensions

- There are many extensions that can be made to these basic algorithms.
- Some are necessary for them to perform well
  - Regularization (later soon)
- Some are for ease of use and tuning
  - Converting the output of a Perceptron to a conditional probability

$$P(y = +1 \mid x) = [1 + \exp(-\,a\,wx)]^{-1}$$

  - Can tune the parameter a
- Multiclass classification

Regularization Via Averaged Perceptron

- An Averaged Perceptron Algorithm is motivated by the following considerations:
  - In the mistake bound model: We don't know when we will make the mistakes.

- Averaged Perceptron returns a weighted average of a number of earlier hypotheses; the weights are a function of the length of no-mistakes stretch.

Regularization Via Averaged Perceptron

- **Training:**

[m: #(examples); k: #(mistakes) = #(hypotheses); c_i: consistency count for $\mathbf{w}_i$]

- Input: a labeled training set $\{(\mathbf{x}_1, y_1),\ldots(\mathbf{x}_m, y_m)\}$
- Number of epochs T
- Output: a list of weighted perceptrons $\{(\mathbf{w}_1, c_1),\ldots,(\mathbf{w}_k, c_k)\}$
- Initialize: k=0; $\mathbf{w}_1 = 0$, $\mathbf{c}_1 = 0$
- Repeat T times:
  - For i =1,…m:
  - Compute prediction y' = $\mathrm{sign}(\mathbf{w}_k{}^T x_i)$
  - If y' = y, then $\mathbf{c}_k = \mathbf{c}_k + 1$

    else: $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{w}_i\, \mathbf{x}$; $\mathbf{c}_{k+1} = 1$; $k = k+1$

- **Prediction:**
- Given: a list of weighted perceptrons $\{(\mathbf{w}_1, c_1),\ldots(\mathbf{w}_k, c_k)\}$; a new example $\mathbf{x}$

  Predict the label($\mathbf{x}$) as follows:

  $$y(\mathbf{x}) = \mathrm{sign}\left[\sum_{1,k} c_i\, \mathrm{sign}(w_i{}^T \mathbf{x})\right]$$

Perceptron algorithm

- Online: can adjust to changing target, over time
- Advantages
  - Simple and computationally efficient
  - Guaranteed to learn a linearly separable problem (convergence, global optimum)
- Limitations
  - Only linear separations
  - Only converges for linearly separable data
  - Not really "efficient with many features"