Machine Learning

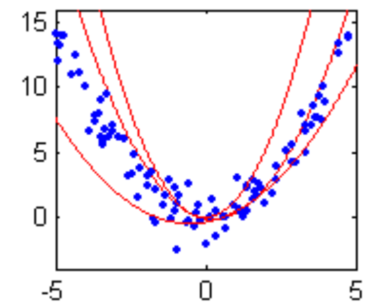
4771

Instructors:
Adrian Weller and Ilia Vovsha

Lecture 5+6: Perceptron & Neural Networks

- Cross-Validation
- Linear decision surface
- Perceptron (Duda 5.1-5.5)
- Convergence proof
- Neural Networks (Bishop 5.1-5.3.2)
- Network Learning, Lagrange multipliers
- Back-Propagation

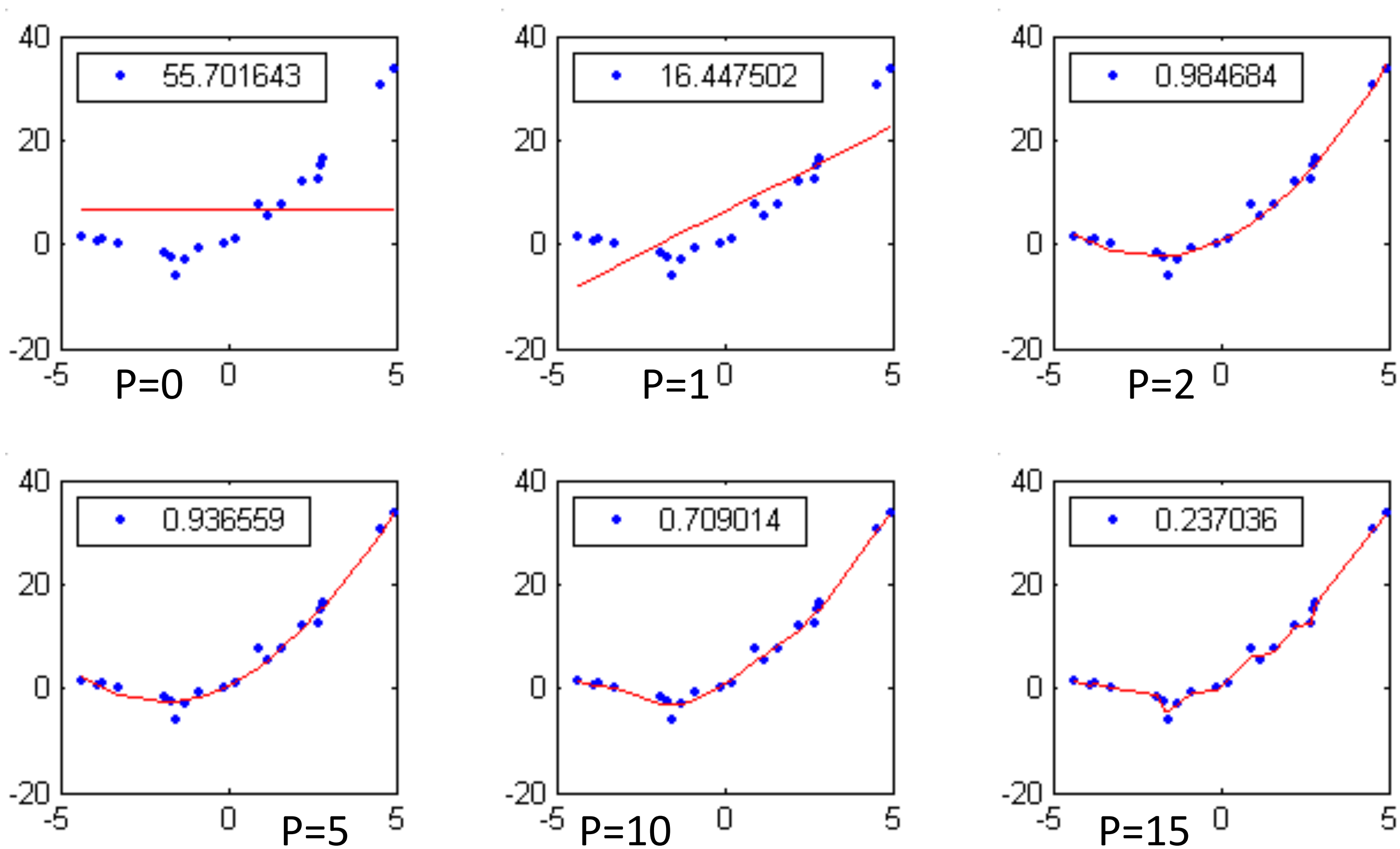
Polynomial Function Classes



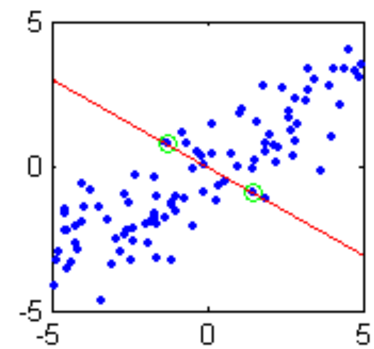
- Back to 1-dim x (D=1) BUT Nonlinear Function Classes
- Polynomial: $f(x;w) = \sum_{p=1}^{P} w_p x^p + w_0$
- Writing Risk: $R(w) = \frac{1}{2N}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2, \quad X = \begin{bmatrix} 1 & x_1^1 & \cdots & x_1^P \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_N^1 & \cdots & x_N^P \end{bmatrix}$
- Order-P polynomial regression fitting for 1D variable is the same as P-dimensional linear regression!
- Construct a multidim x-vector from x scalar: $x_i = [x_i^0, x_i^1, x_i^2]^T$
- More generally any function: $x_i = [\phi_0(x_i), \phi_1(x_i), \phi_2(x_i)]^T$

Underfitting/Overfitting

- Try varying P. Higher P fits a more complex function class
- Observe R(w*) drops with bigger P



Evaluating the Model



- Unfair to use training error to find best order P
- High P (vs. N) can overfit, even linear case!
- min R(w*) not on training but on future data
- Want model to *Generalize* to future data

Expected (true) loss: $R_{\text{expected}}(w) = \int L(y, f(x;w))P(x,y)\,dxdy$

- One approach: split data into training / testing portion

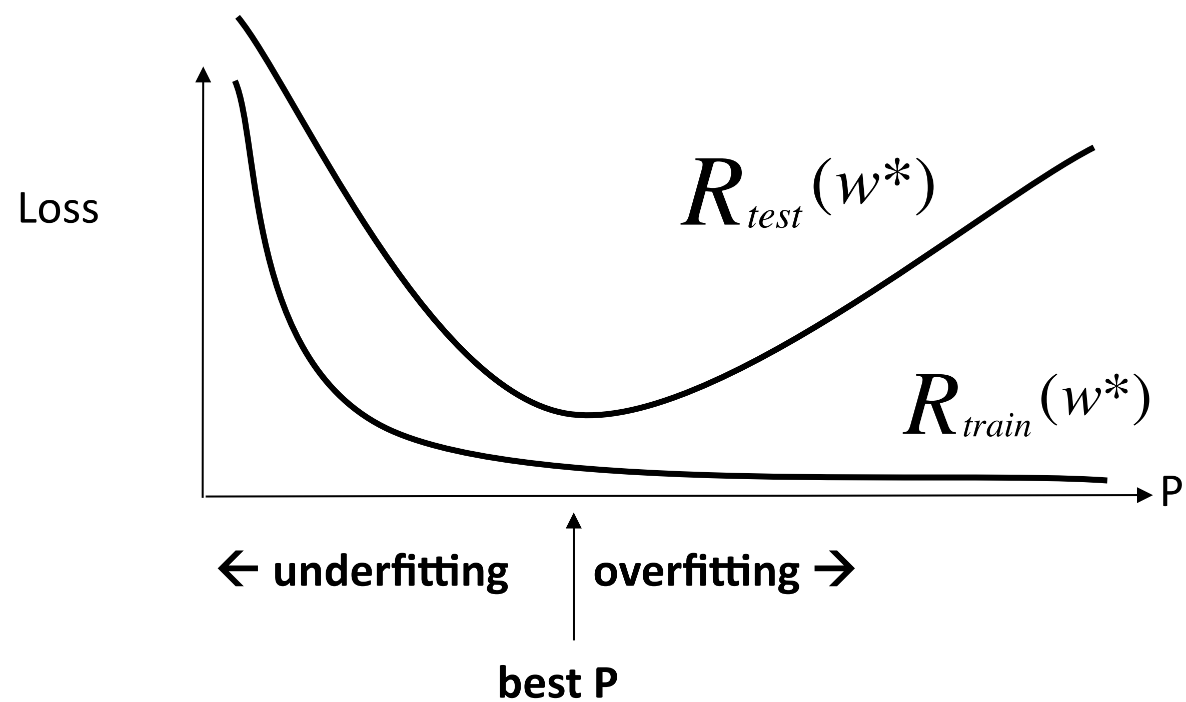
$\{(x_1, y_1), \ldots, (x_v, y_v)\}$ $\{(x_{v+1}, y_{v+1}), \ldots, (x_N, y_N)\}$

- Estimate ω* with training (empirical) loss: $R_{train}(w) = \frac{1}{2v}\sum_{i=1}^{v}(y_i - w^T x_i)^2$

- Evaluate P with testing loss: $R_{test}(w) = \frac{1}{2(N-v)}\sum_{i=v+1}^{N}(y_i - w^T x_i)^2$

Validation

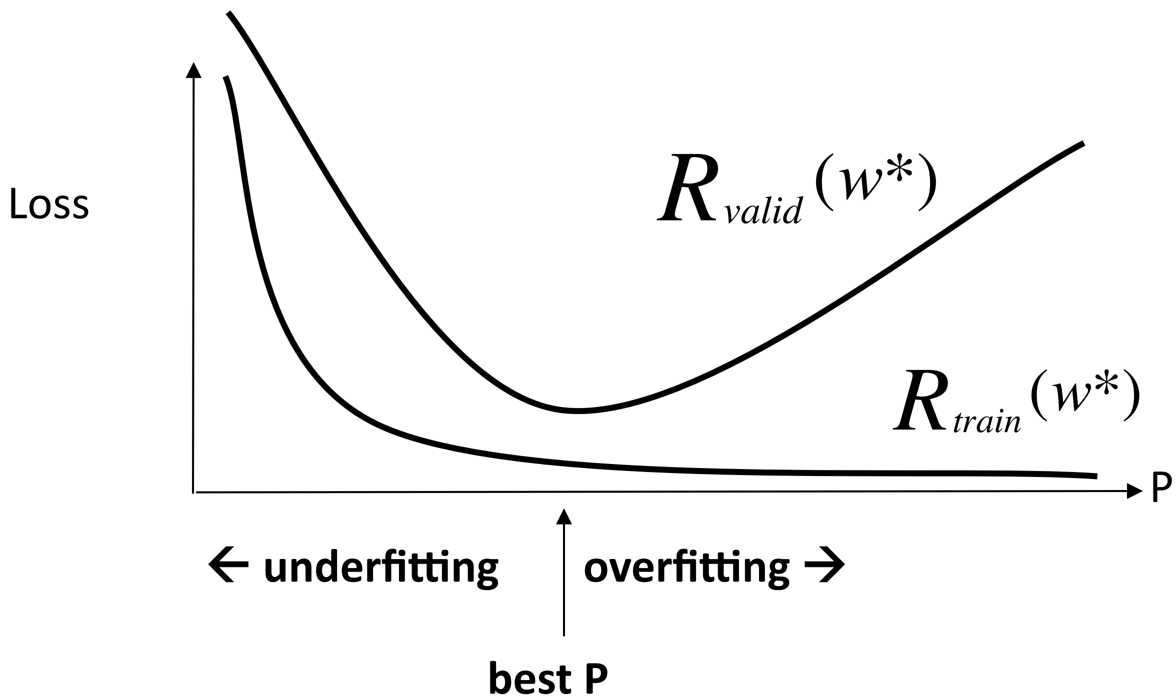
- Try fitting with different polynomial order P
- Select P which gives lowest $R_{test}(w^*)$



- Think of P as a measure of the complexity of the model
- Higher order polynomials are more flexible and complex

Cross-Validation

- Better idea: split data into three sets (training / validation / test)
- Even better idea: split data into two sets (training / test) but do *K-fold cross-validation* on the training set
  - K folds, K-1 for training, 1 for testing; repeat process K times; average error
- Best idea (sometimes): *leave-one-out cross-validation* on the training set
  - N examples, N-1 for training 1 for testing; repeat process N times; average error



The Weierstrass Approximation Theorem

- Theorem (1885):

  Suppose $f(x)$ is a continuous real-valued function defined on the real interval $[a,b]$. For every $\epsilon > 0$, there exists a polynomial function p over $\mathbb{R}$ such that $\forall x \in [a,b]$, we have $|f(x) - p(x)| < \epsilon$, or equivalently, $\|f(x) - p(x)\|_\infty < \epsilon$.

- Definition of supremum (infinity) norm:

  $$\|f(x) - p(x)\|_\infty = \sup\{|f(x) - p(x)|\}$$

Parametric Paradigm (Philosophy)

- Heyday: 1930 – 1960's
- Standard assumptions: familiar problem & underlying physical process
- Problem: set of parameters that needs to be estimated
- Approach: adopt the Maximum-Likelihood / MAP / Bayesian method
- Strength:
    1. If assumptions are correct, we obtain more accurate estimates
    2. Math is simpler & faster to compute.
- Principle: if it works for the *asymptotic* case, should work for a small sample too.

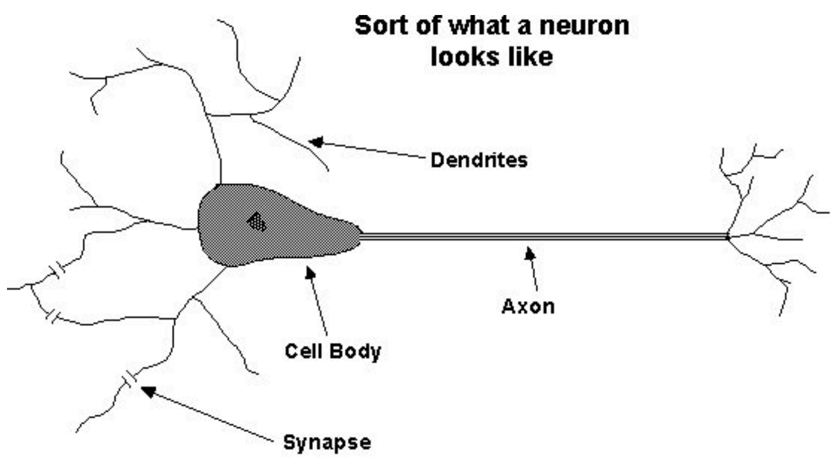
Parametric Paradigm (Beliefs?)

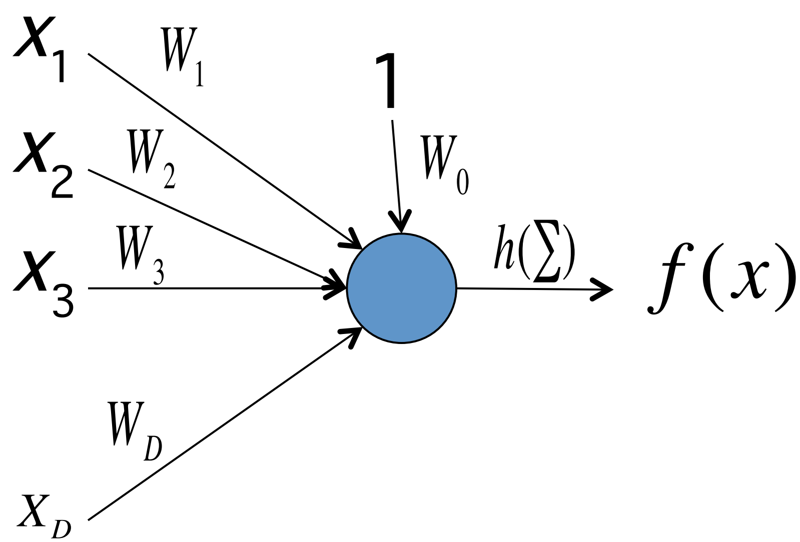
A. It is possible to find a good approximation to any function with few parameters
   - Evidence (?): Weierstrass Approximation Theorem
   - Strength: computationally simple
B. The underlying law behind many real-life problems is the normal law
   - Evidence: Central Limit Theorem
C. MLE / MAP / Bayesian are good approaches for estimating the parameters
   - Evidence: conditional optimality (restricted set or asymptotic case)

The Neuron as Regression

- The McCullough-Pitts Neuron is a graphical representation of linear regression:
  - Edges (synapses): multiply signal by scalar weight
  - Nodes: sum the inputs
  - Parameters: $w_1 \ldots w_D$ = weights w_0 = bias
  - Activation function: linear

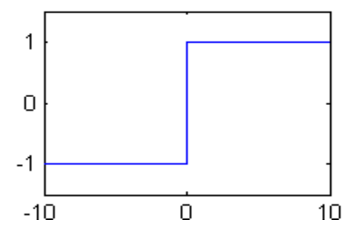
$$f(\mathrm{x};\mathrm{w}) = \sum_{i=1}^{D} w_i x_i + w_0$$



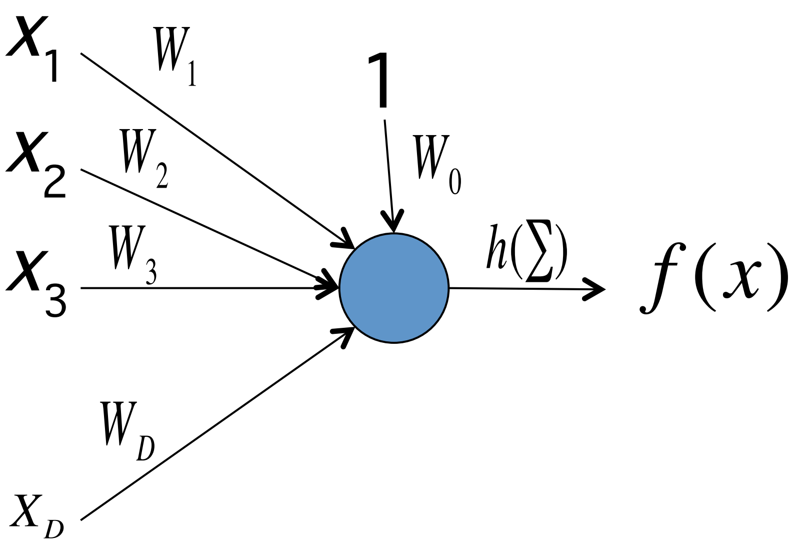


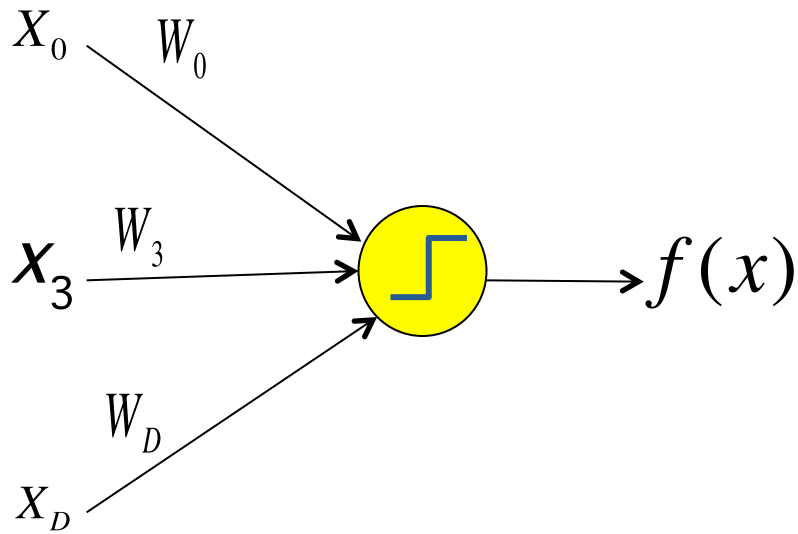
The Neuron as Classifier

- The Neuron as a graphical representation of a linear classifier:
  - Edges (synapses): multiply signal by scalar weight
  - Nodes: sum the inputs
  - Parameters (augmented vector): w_0 … w_D = weights
  - Activation function: step function



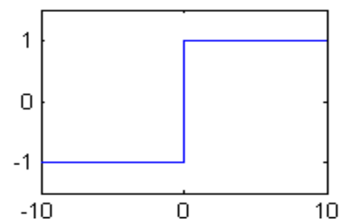
$$x_A = [1, x]; \quad f(x;w) = \sum_{i=0}^{D} w_i x_i = w^T x$$





Step Function

$$h(z) = \begin{cases} -1 & when\ z < 0 \\ +1 & when\ z \geq 0 \end{cases}$$

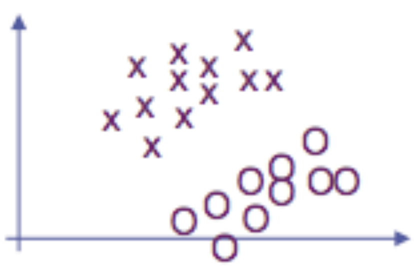


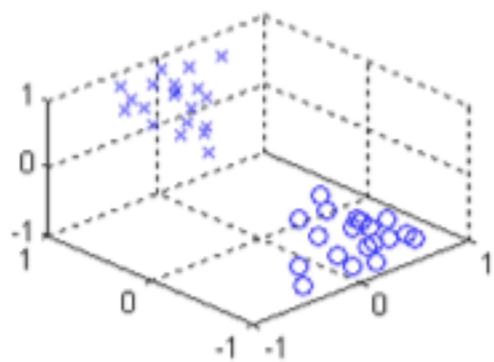
$$f(\mathrm{x};\mathrm{w}) = \sum_{i=0}^{D} w_i x_i = w^T x$$

$$h\left[f(\mathrm{x};\mathrm{w})\right] \Rightarrow \frac{w^T x \geq 0: \quad assign\ 1}{w^T x < 0: \quad assign\ -1}$$

Linear Decision Surface

- Previously: form of probability densities is known
- Now: form of discriminant/decision surface is known
- The equation $f(\mathrm{x}) = w^T x = 0$ defines the decision surface
- Linear surface = hyperplane





Geometry of Linear Surface

- **W** is normal to any vector lying in the hyperplane ***H***
  - Proof: $x_1, x_2 \ on \ H \Rightarrow w^T x_1 = 0, w^T x_2 = 0$

    $\Rightarrow w^T x_1 - w^T x_2 = 0$

    $\Rightarrow w^T (x_1 - x_2) = 0$
- ***H*** divides the space into two half spaces.
- Normal vector (**w**) points to the positive side of ***H*** (why?)
- Discriminant function ***f(x)*** is proportional to the distance from **x** to ***H***
  - Proof: $x = x_{pr} + r\frac{w}{\|w\|} \Rightarrow f(x) = f(x_{pr} + r\frac{w}{\|w\|})$

    $\Rightarrow f(x) = w^T x_{pr} + r(\frac{w^T w}{\|w\|})$

    $\Rightarrow f(x) = r(\frac{\|w\|^2}{\|w\|})$

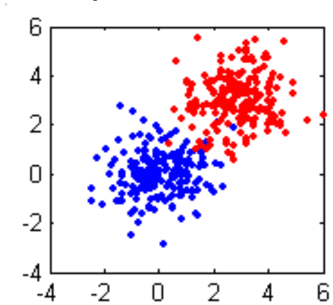
    $$\Rightarrow r = \frac{f(x)}{\|w\|}$$

Linearly Separable 2-Class Problem

- Start with training dataset

$$\mathcal{X} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\} \quad x \in \mathbb{R}^D \quad y \in \{-1, 1\}$$

- Have N (vector, label {-1,1}) pairs
- Find a discriminant function f(x) to predict class (label) from x
- Assume there exists a weight vector **w** that classifies all samples correctly
  - ➢ Such **w** is called a solution vector
  - ➢ More than one (infinite #) **w**: solution region
  - ➢ We say the data is "linearly seperable"
  - ➢ Otherwise "non-separable" (example on the right)



- Symmetry:

$$\frac{w^T x_i \geq 0 : \quad assign\ 1}{w^T x_i < 0 : \quad assign\ -1} \Rightarrow y_i(w^T x_i) > 0$$