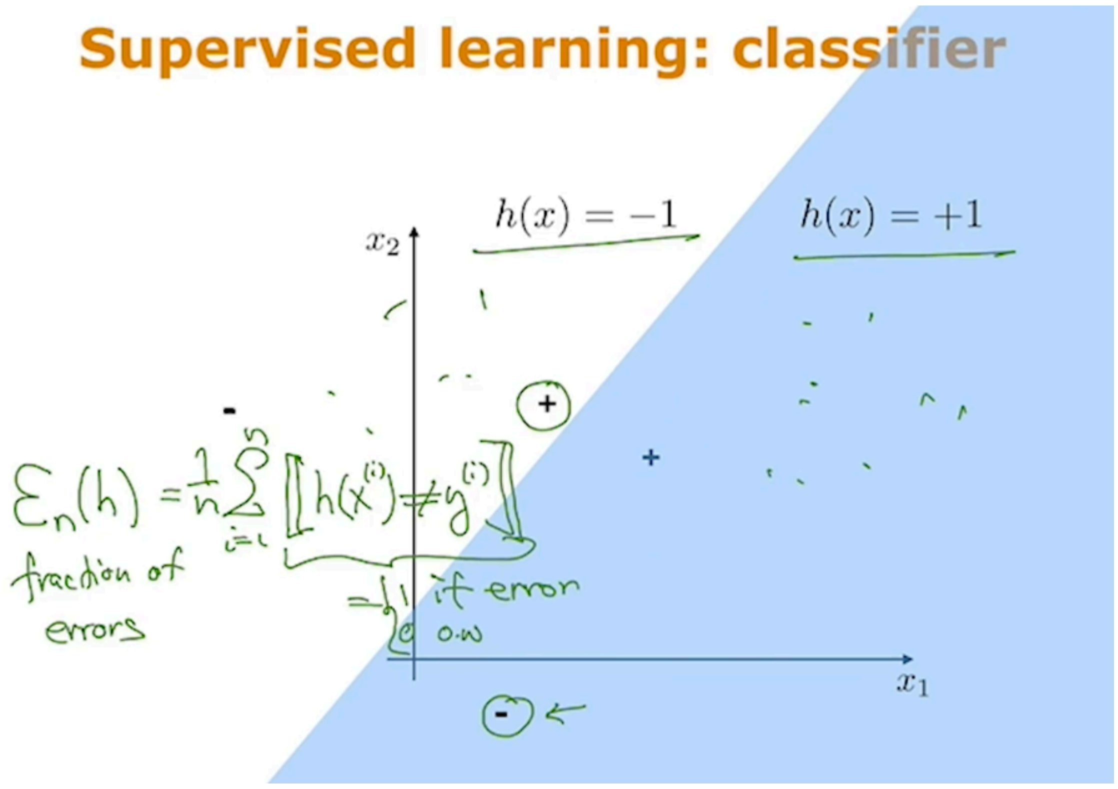
Week 1. Linear Classifiers and Generalization

Lecture 1. Intro to Machine Learning

1. ML is a discipline aims to design, understand, and apply computer programs that learn from experience for the purpose of modeling, prediction and control.
2. Supervised learning,
- we are given an example (e.g. an image) along with a target (e.g. what object is in the image), and the goal of the machine learning algorithm is to find out how to produce the target from the example.
- More specifically, in supervised learning, we hypothesize a collection of functions (or mappings) parametrized by a parameter, from the examples (e.g. the images) to the targets (e.g. the objects in the images). The machine learning algorithm then automates the process of finding the parameter of the function that fits with the example-target pairs the best.

3. Linear Classifiers

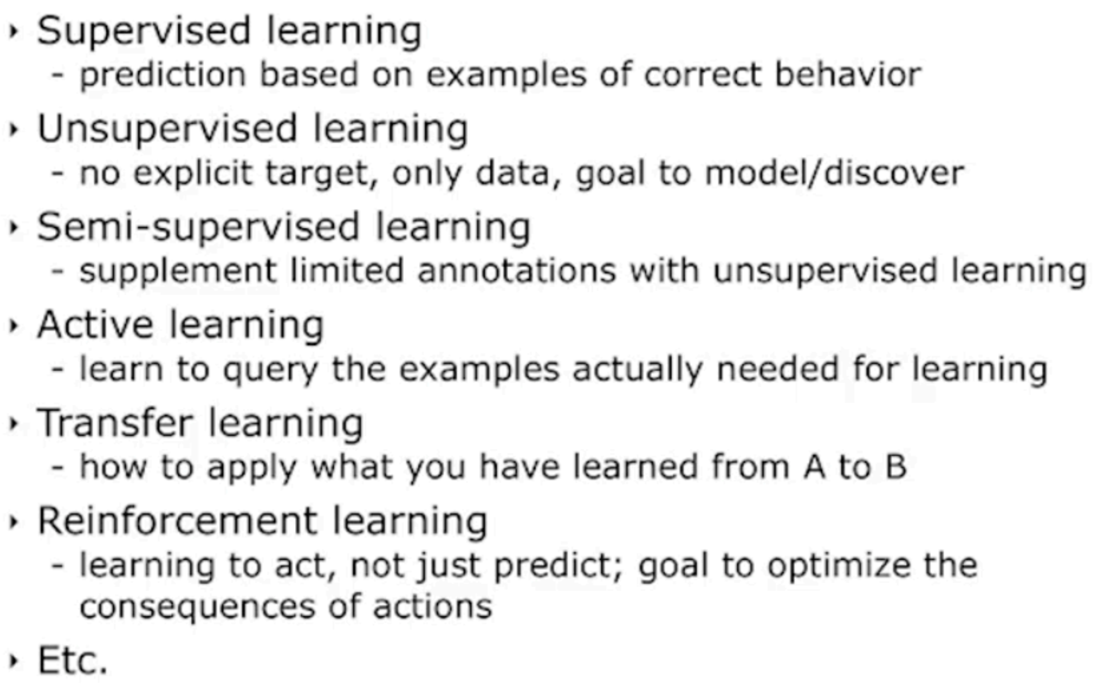

Supervised learning: classifier
$h(x) = -1$
$h(x) = +1$
x_2
x_1
$\mathcal{E}_n(h) = \frac{1}{n}\sum_{i=1}^{n} [\![h(x^{(i)}) \neq y^{(i)}]\!]$
fraction of errors
$= \{ 1$ if error, 0 o.w

- Training data can be graphically depicted on a (hyper)plane. **Classifiers** are **mappings** that take **feature vectors as input** and produce **labels as output**.
- Training Error
- Hypothesis space: the set of possible classifiers

- Classification and regression
- Classification maps **feature vectors** to **categories**

- Regression maps feature vectors to real numbers.

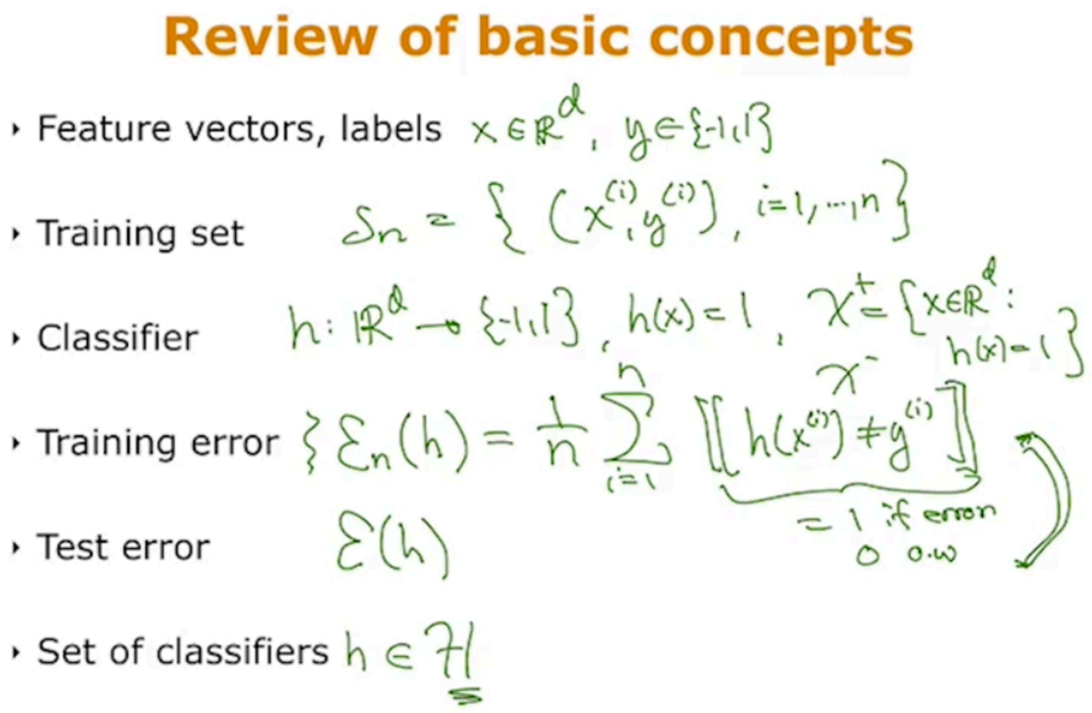
- Types of Machine Learning



- Supervised learning
  - prediction based on examples of correct behavior
- Unsupervised learning
  - no explicit target, only data, goal to model/discover
- Semi-supervised learning
  - supplement limited annotations with unsupervised learning
- Active learning
  - learn to query the examples actually needed for learning
- Transfer learning
  - how to apply what you have learned from A to B
- Reinforcement learning
  - learning to act, not just predict; goal to optimize the consequences of actions
- Etc.

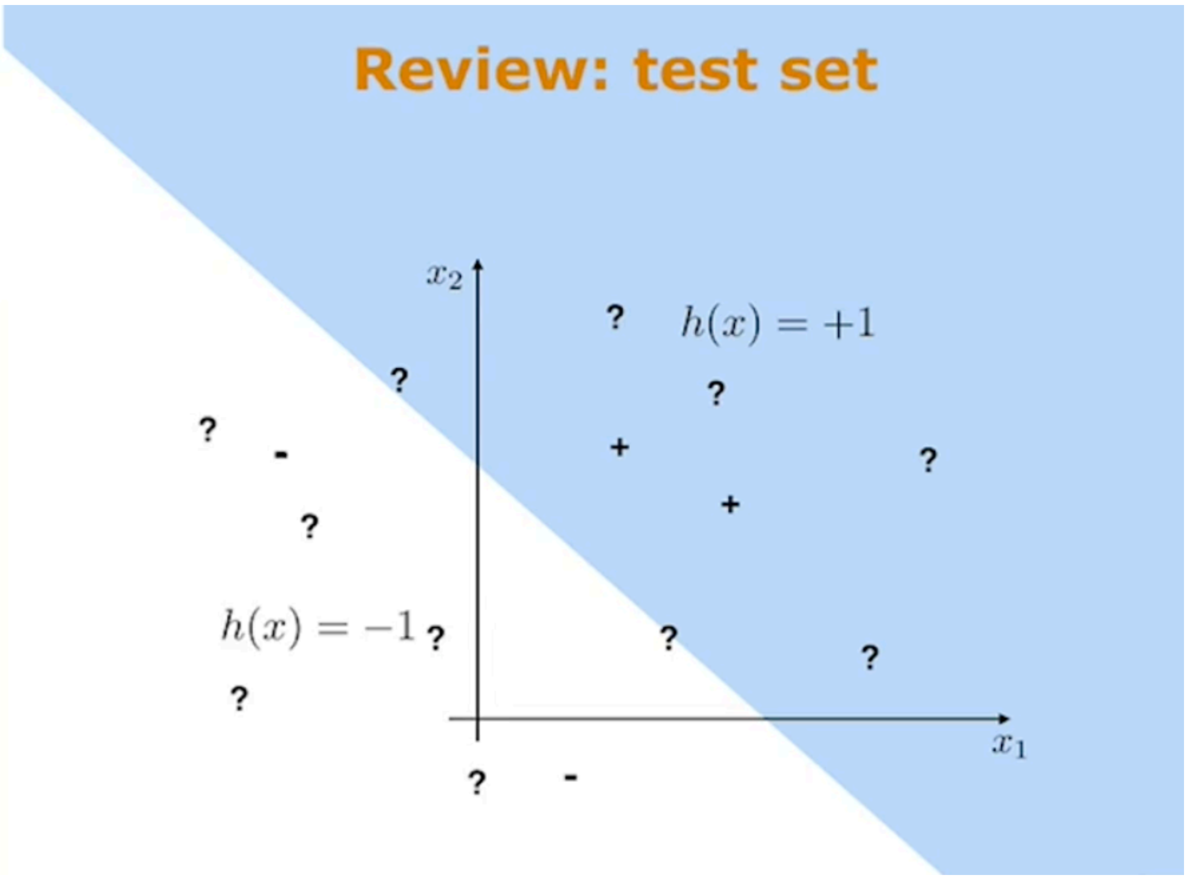
Lecture 2. Linear Classifier and Perceptron Algorithm

1. Review



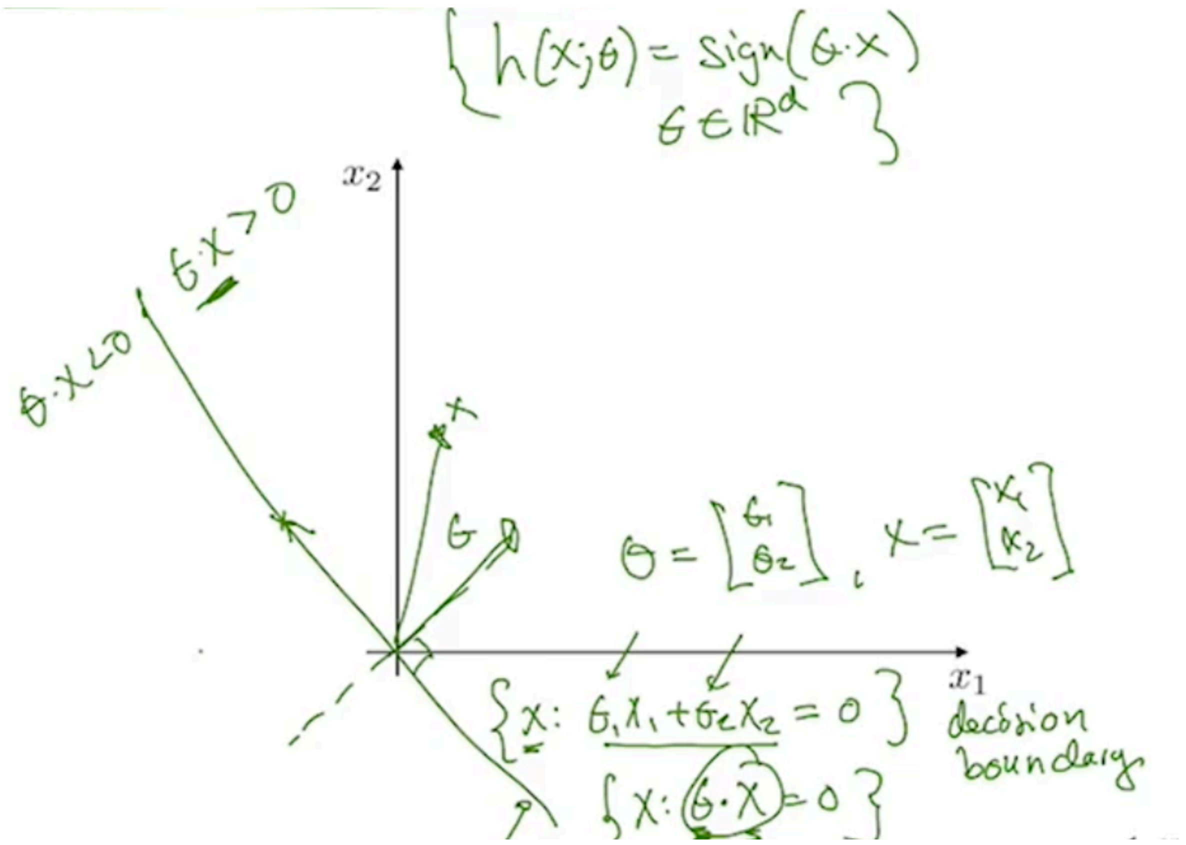
Review of basic concepts

- Feature vectors, labels $x \in \mathbb{R}^d$, $y \in \{-1,1\}$
- Training set $S_n = \{ (x^{(i)}, y^{(i)}), i=1,\dots,n \}$
- Classifier $h: \mathbb{R}^d \to \{-1,1\}$, $h(x)=1$, $\mathcal{X}^+ = \{x \in \mathbb{R}^d : h(x)=1\}$, $\mathcal{X}^-$
- Training error $\mathcal{E}_n(h) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{[[h(x^{(i)}) \neq y^{(i)}]]}_{\substack{=1 \text{ if error} \\ 0 \text{ o.w.}}}$
- Test error $\mathcal{E}(h)$
- Set of classifiers $h \in \mathcal{H}$

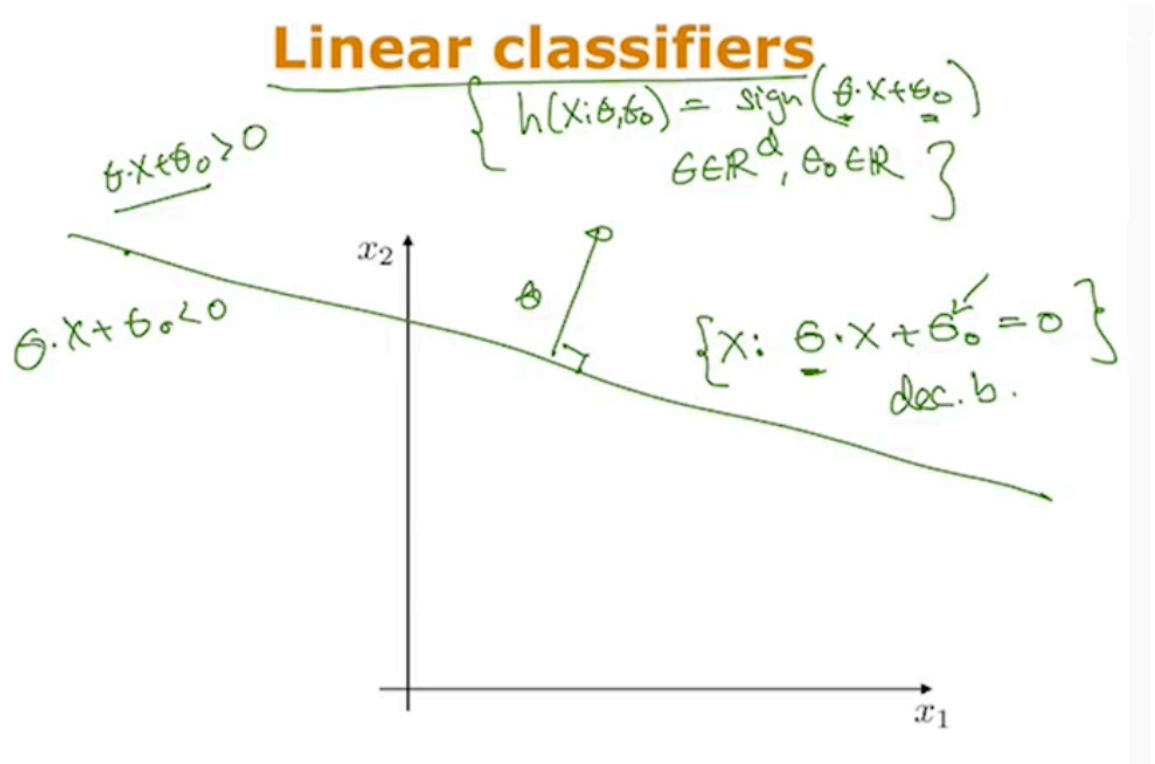


2. Linear Classifiers Mathematically revisits

- The simple case, linear classifiers through origin

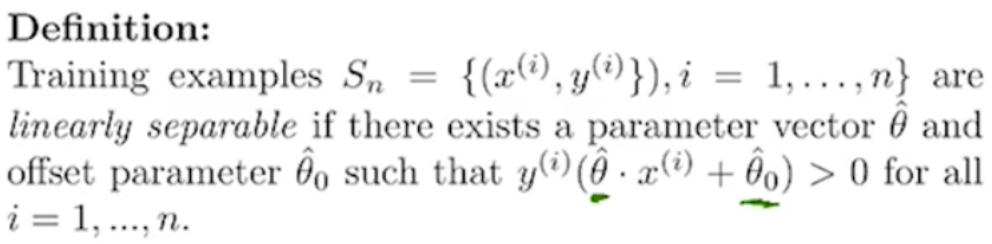

$\{h(x;\theta) = \text{sign}(\theta \cdot x)$
$\theta \in \mathbb{R}^d\}$
x_2
x_1
$\theta \cdot x > 0$
$\theta \cdot x < 0$
x
θ
$\theta = \begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix}, \; x = \begin{bmatrix}x_1 \\ x_2\end{bmatrix}$
$\{x: \theta_1 x_1 + \theta_2 x_2 = 0\}$
$\{x: \theta \cdot x = 0\}$
decision boundary

- The parameter vector \Theta is orthogonal to all the points that lie on the decision boundary
- Linear Classifiers


Linear classifiers
$\{h(x;\theta,\theta_0) = \text{sign}(\theta \cdot x + \theta_0)$
$\theta \in \mathbb{R}^d, \theta_0 \in \mathbb{R}\}$
$\theta \cdot x + \theta_0 > 0$
$\theta \cdot x + \theta_0 < 0$
θ
x_2
x_1
$\{x: \theta \cdot x + \theta_0 = 0\}$
dec. b.

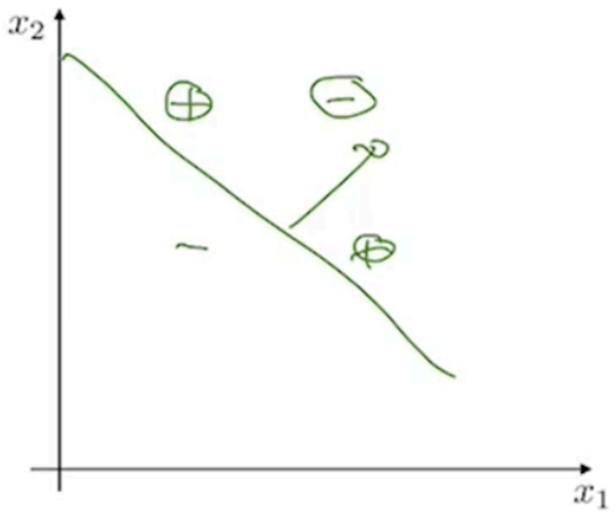
- Linear Separation

- Linear Separation Df

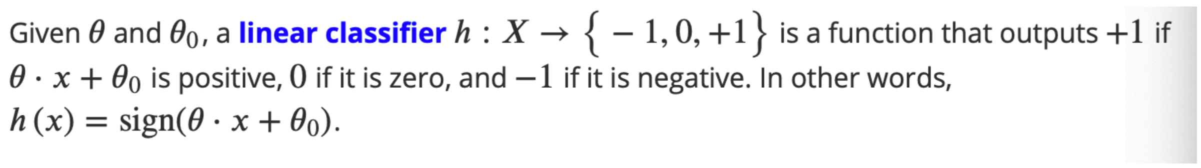


Definition:
Training examples $S_n = \{(x^{(i)}, y^{(i)}\}), i = 1, \ldots, n\}$ are *linearly separable* if there exists a parameter vector $\hat{\theta}$ and offset parameter $\hat{\theta}_0$ such that $y^{(i)}(\underline{\hat{\theta}} \cdot x^{(i)} + \underline{\hat{\theta}_0}) > 0$ for all $i = 1, ..., n$.

Linear separation: ex

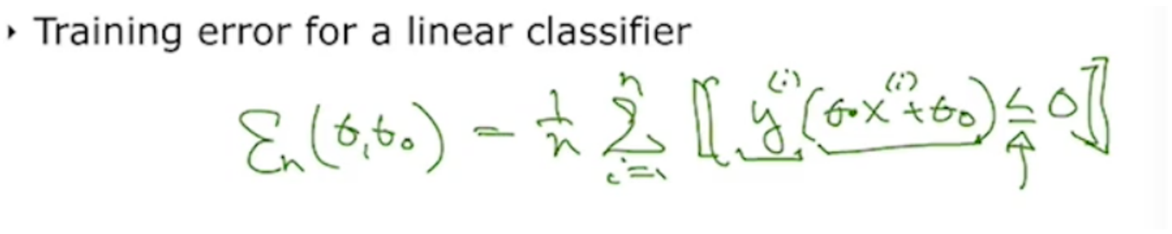


This is constrained since linear classifier can not classify training pts.



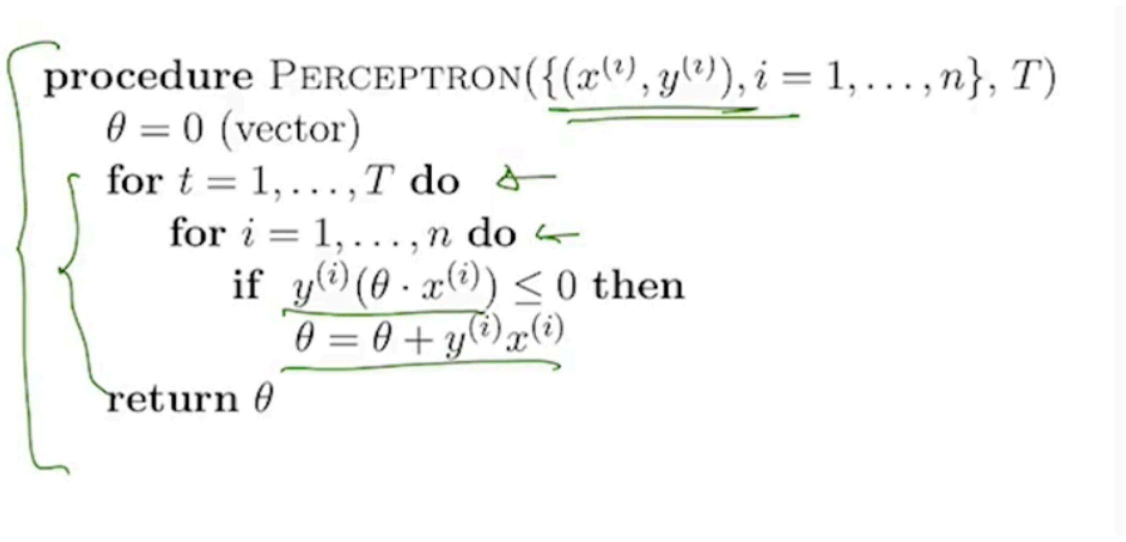
Given θ and θ_0, a **linear classifier** $h : X \rightarrow \{-1, 0, +1\}$ is a function that outputs $+1$ if $\theta \cdot x + \theta_0$ is positive, 0 if it is zero, and -1 if it is negative. In other words, $h(x) = \text{sign}(\theta \cdot x + \theta_0)$.

- The Perception Algorithm



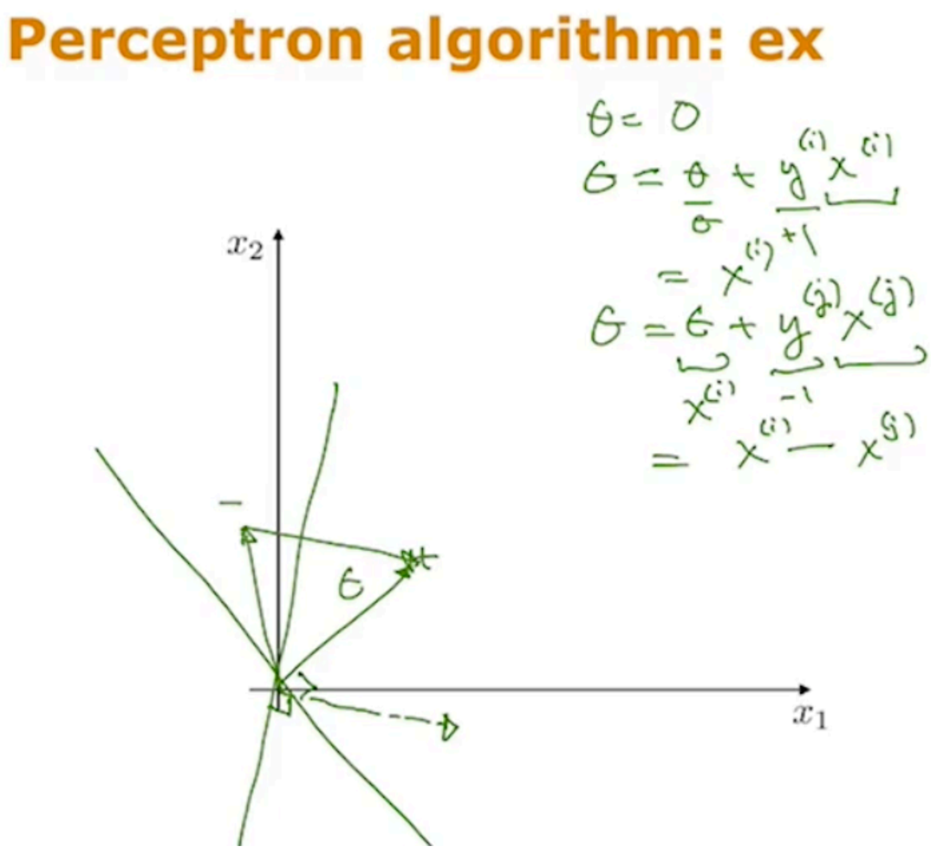
- Training error for a linear classifier

$$\mathcal{E}_n(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} [\![\underline{y^{(i)}} \underline{(\theta \cdot x^{(i)} + \theta_0)} \underset{\uparrow}{\leq} 0]\!]$$



```
procedure PERCEPTRON($\underline{\{(x^{(i)}, y^{(i)}), i = 1, \ldots, n\}}$, $T$)
    $\theta = 0$ (vector)
    for $t = 1, \ldots, T$ do
        for $i = 1, \ldots, n$ do
            if $\underline{y^{(i)}(\theta \cdot x^{(i)})} \le 0$ then
                $\underline{\theta = \theta + y^{(i)} x^{(i)}}$
    return $\theta$
```

- Perceptron Algorithm without offset



Perceptron algorithm: ex

$$\theta = 0$$

$$\theta = \underbrace{\theta}_{0} + \underbrace{y^{(i)}}_{+1} \underbrace{x^{(i)}}$$

$$= x^{(i)}$$

$$\theta = \underbrace{\theta}_{x^{(i)}} + \underbrace{y^{(j)}}_{-1} \underbrace{x^{(j)}}$$

$$= x^{(i)} - x^{(j)}$$

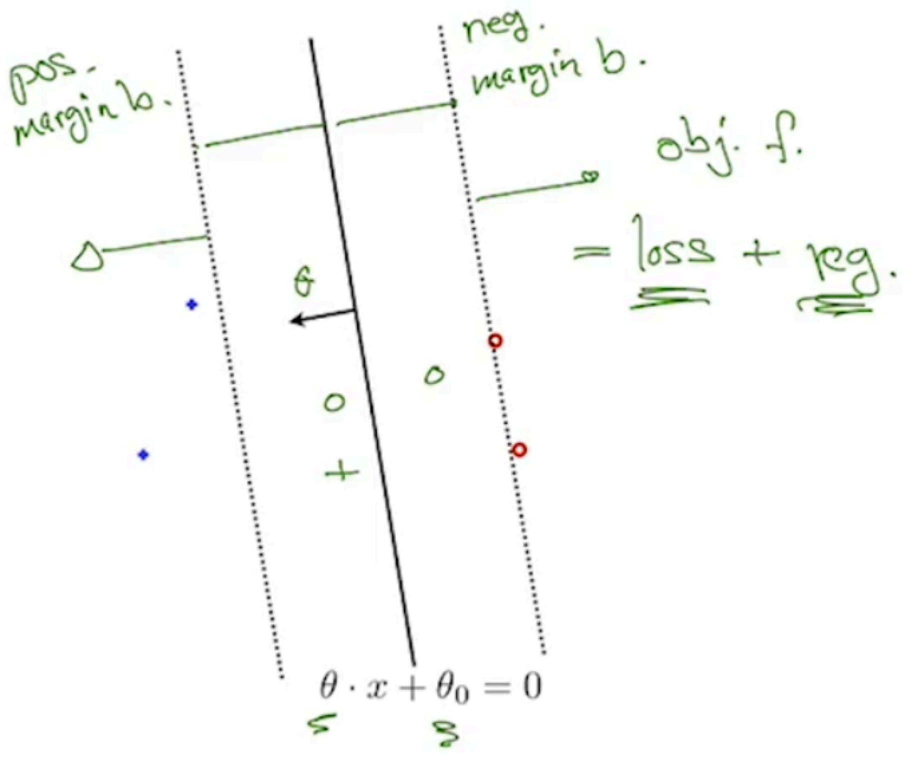
- Perceptron Algorithm with offset

```
Perceptron$\left(\left\{ (x^{(i)}, y^{(i)}), i = 1, \ldots, n\right\}, T\right)$ :
    initialize $\theta = 0$(vector); $\theta_0 = 0$(scalar)
        for $t = 1, \ldots, T$ do
            for $i = 1, \ldots, n$ do
                if $y^{(i)}(\theta \cdot x^{(i)} + \theta_0) \leq 0$ then
                update $\theta = \theta + y^{(i)}x^{(i)}$
                update $\theta_0 = \theta_0 + y^{(i)}$
```

This is always better than perceptron algorithm without offset

Lecture 3. Hinge Loss, Margin Boundaries and Regularization

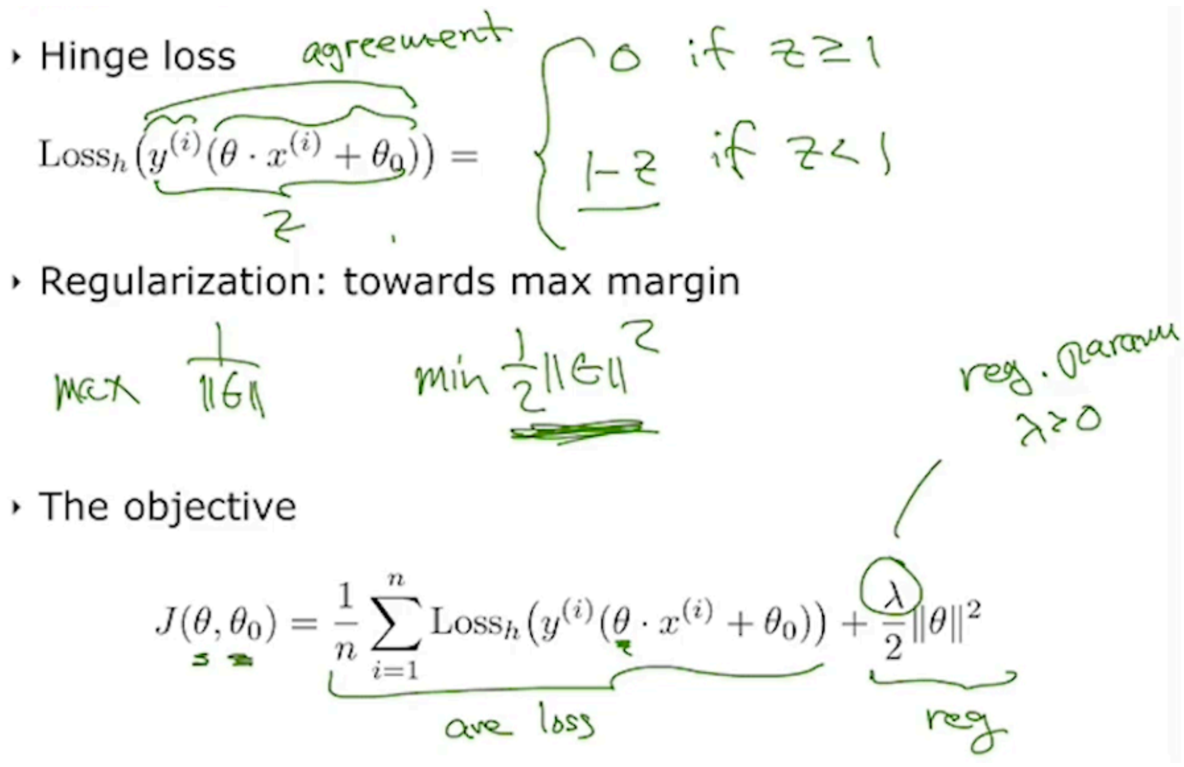
- Finding a large margin classifier - learning as optimization



Formalize the objective function and find theta and theta_0 to minimize the objective function

The **Decision Boundary** is the set of points x which satisfy
$\theta \cdot x + \theta_0 = 0$.
The **Margin Boundary** is the set of points x which satisfy
$\theta \cdot x + \theta_0 = \pm 1$.
So, the distance from the decision boundary to the margin boundary is $1/||\theta||$.

- Hinge Loss and Objective Function



- Hinge loss

$$\mathrm{Loss}_h(\underbrace{\overset{\text{agreement}}{y^{(i)}(\theta \cdot x^{(i)} + \theta_0)}}_{z}) = \begin{cases} 0 & \text{if } z \geq 1 \\ \underline{1-z} & \text{if } z < 1 \end{cases}$$

- Regularization: towards max margin

$$\max \frac{1}{\|\theta\|} \qquad \min \underline{\frac{1}{2}\|\theta\|^2}$$

- The objective

$$J(\theta, \theta_0) = \underbrace{\frac{1}{n}\sum_{i=1}^{n} \mathrm{Loss}_h(y^{(i)}(\theta \cdot x^{(i)} + \theta_0))}_{\text{ave loss}} + \underbrace{\frac{\overset{\text{reg. param } \lambda \geq 0}{\boxed{\lambda}}}{2}\|\theta\|^2}_{\text{reg}}$$

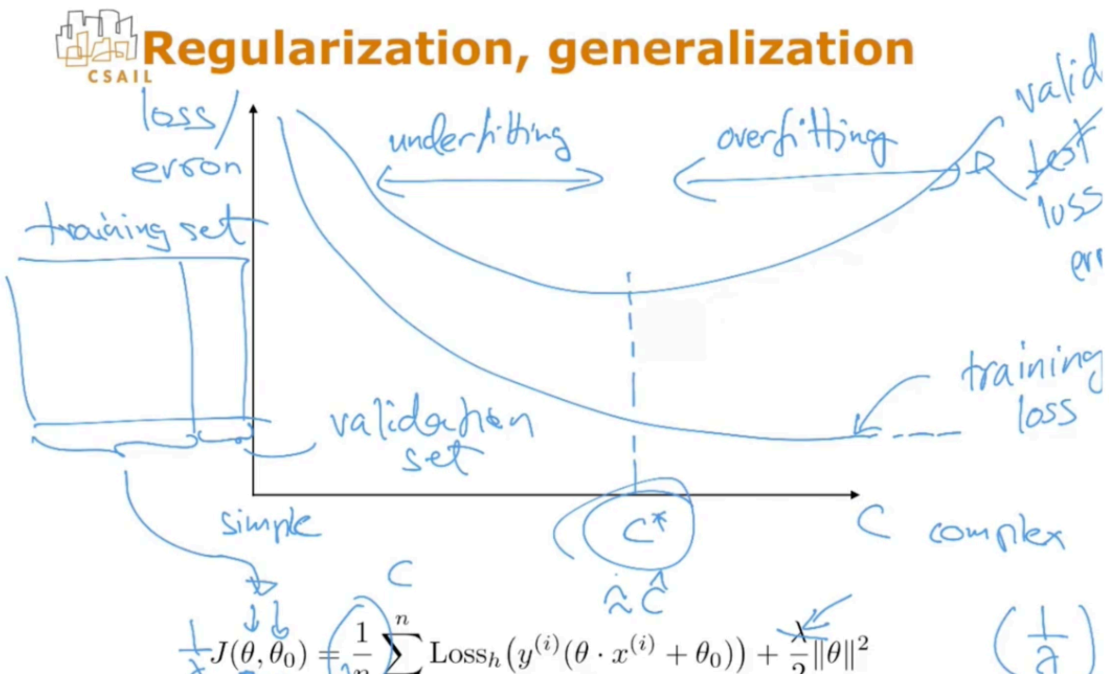
Objective function = average loss + regularization

- **Loss functions tell you in general how bad the prediction is.**

Lecture 4. Linear Classification and Generalization

- **Review**
- **Recall that Machine learning problems are often cast as optimization problems.**

- Small lambda is about correctly classify data points, using a large lambda means we care more about maximizing margin
  - Regularization and Generalization


Regularization, generalization
loss/error
training set
validation set
underfitting
overfitting
valid test loss err
training loss
simple
complex
C
C*
≈ C
$\frac{1}{\lambda}J(\theta,\theta_0) = \frac{1}{n}\sum^{n}\text{Loss}_h\left(y^{(i)}(\theta\cdot x^{(i)}+\theta_0)\right) + \frac{\lambda}{2}\|\theta\|^2$
$(\frac{1}{a})$

A c* is possible to achieve low loss error.

- Gradient Descent

Assume $\theta \in \mathbb{R}$. Our goal is to find θ that minimizes

$$J(\theta, \theta_0) = \frac{1}{n}\sum_{i=1}^{n} \text{Loss}_h\left(y^{(i)}\left(\theta \cdot x^{(i)} + \theta_0\right)\right) + \frac{\lambda}{2} \|\theta\|^2$$

through gradient descent. In other words, we will

1. Start θ at an arbitrary location: $\theta \leftarrow \theta_{start}$
2. Update θ repeatedly with $\theta \leftarrow \theta - \eta \frac{\partial J(\theta,\theta_0)}{\partial \theta}$ until θ does not change significantly

3. Stochastic Gradient Descent

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left[\text{Loss}_h(y^{(i)}\theta\cdot x^{(i)}) + \frac{\lambda}{2}\|\theta\|^2\right]$$

Select $i \in \{1, \ldots, n\}$ at random

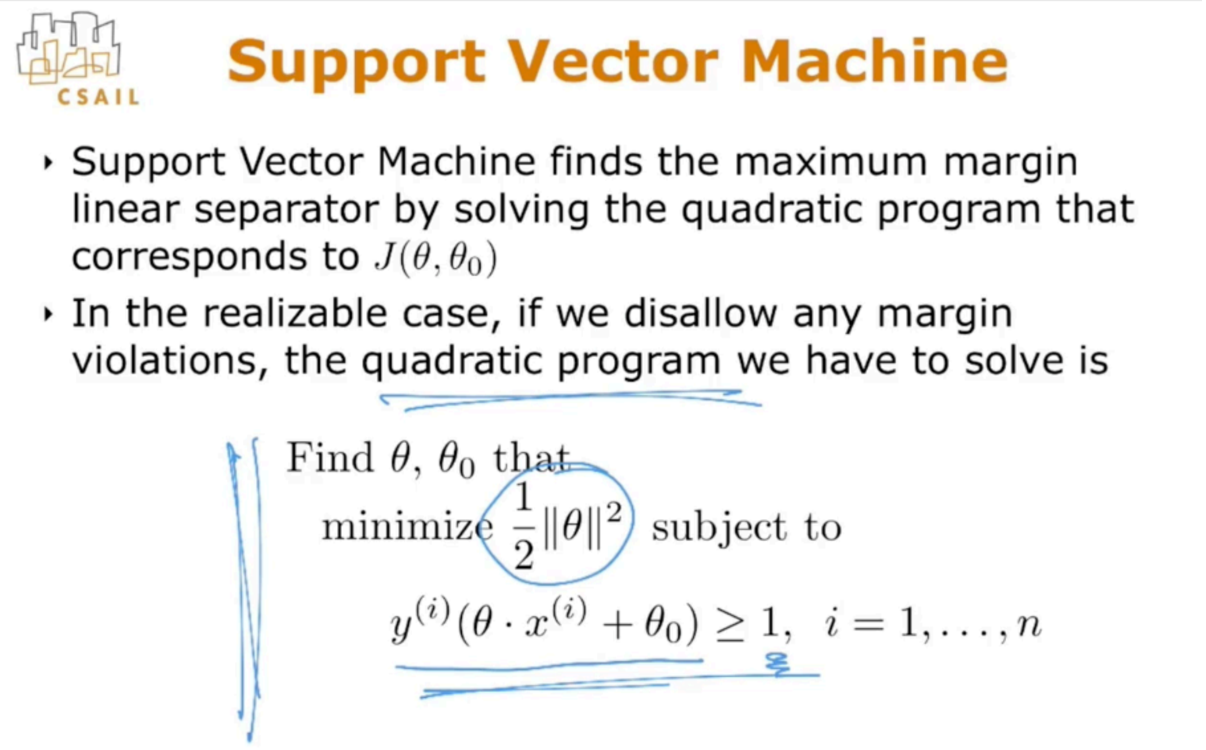
$$\theta \leftarrow \theta - \eta_t \nabla_\theta\left[\text{Loss}_h(y^{(i)}\theta\cdot x^{(i)}) + \frac{\lambda}{2}\|\theta\|^2\right]$$

This is different from perceptron as theta is updated even when there is no mistake

$$\theta \leftarrow \begin{cases} (1-\lambda\eta)\,\theta \text{ if Loss=0} \\ (1-\lambda\eta)\,\theta + \eta y^{(i)} x^{(i)} \text{ if Loss>0} \end{cases}$$

4. The Realized Case (linearly separable) - Quadratic Program

- Support Vector Machine



Support Vector Machine

- Support Vector Machine finds the maximum margin linear separator by solving the quadratic program that corresponds to $J(\theta, \theta_0)$
- In the realizable case, if we disallow any margin violations, the quadratic program we have to solve is

Find θ, θ_0 that

minimize $\frac{1}{2}\|\theta\|^2$ subject to

$$y^{(i)}(\theta \cdot x^{(i)} + \theta_0) \geq 1, \quad i = 1, \ldots, n$$

Large margin liner classifier can be also obtained via solving a quadratic program (SVM)