Machine Learning 4771

Instructor: Itsik Pe’er

Administration

- Quiz:
  - 30 minutes
  - Multiple choice
  - Mudd 833/PUP428/CVN proctorship/IDS

- 2nd Quiz: April 10th

Piazza

- A professional, not social forum
- Avoid offensive language
- Report issues to me/head TA
- Avoid staff feedback
  - Send to me, iachair, CULPA, class evaluation
- No bullying

- Violators will be kicked off the forum

Gradient Descent Algorithm

Initialize

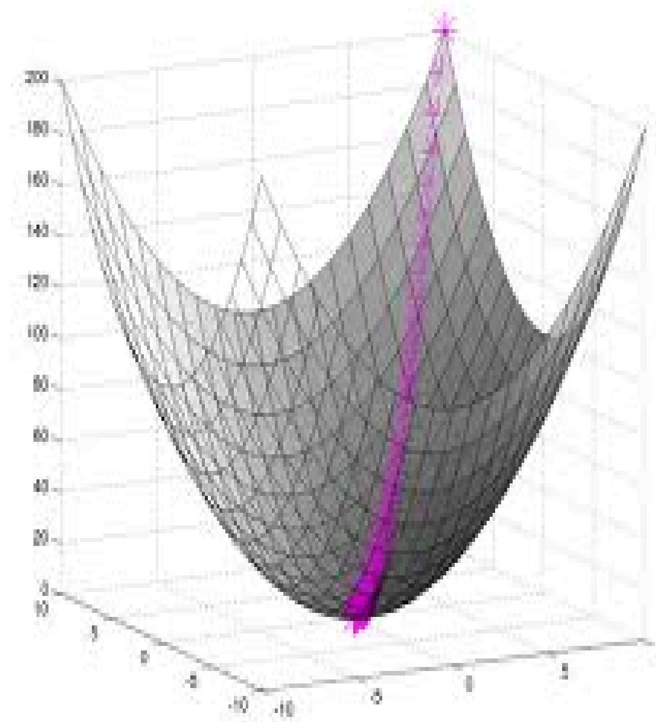
$t \leftarrow 0$; choose θ^0

$\theta^{t+1} \leftarrow \theta^t - \eta_t \nabla_\theta R_{emp}|_{\theta^t}$

Too small: slow
too big: unstable

$t \leftarrow t+1$

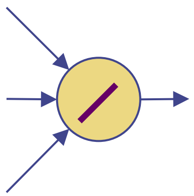
while
$\|\theta^t - \theta^{t-1}\| \geq \epsilon$

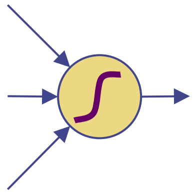


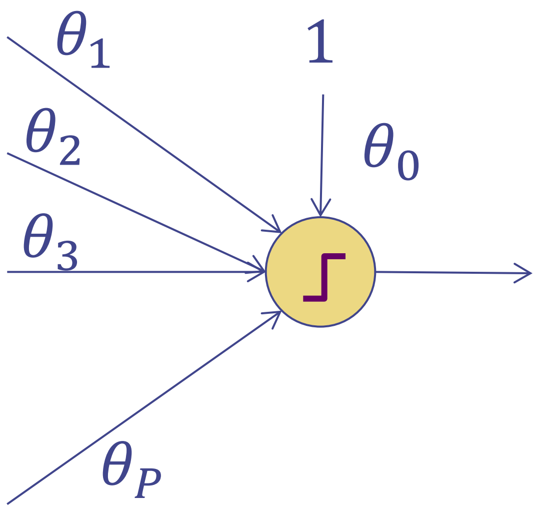
Class 7

- Perceptrons
- Online & Stochastic Gradient Descent
- Convergence Guarantee
- Gap tolerance

Perceptron (another Neuron)







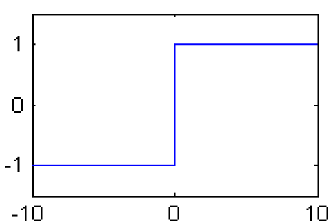
Perceptron (another Neuron)

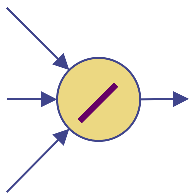
- Classification scenario once again but consider +1, -1 labels

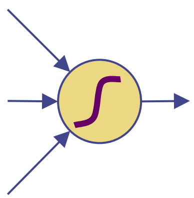
$$X = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\}, \boldsymbol{x} \in \boldsymbol{R}^D, y \in \{-1,1\}$$

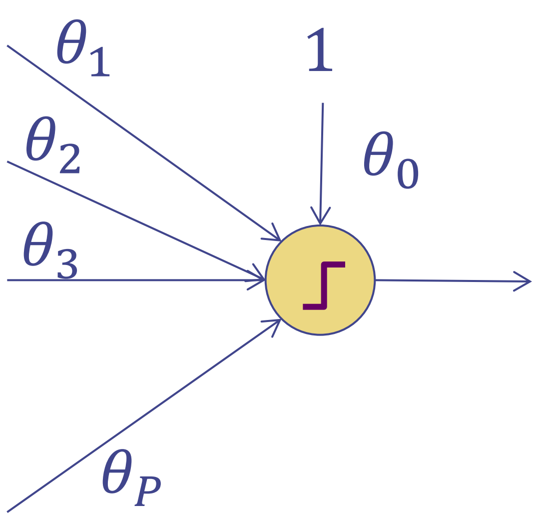
- A better choice for a classification squashing function is

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





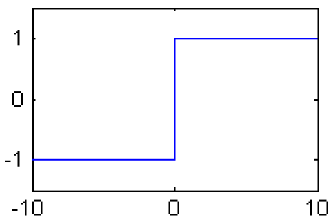


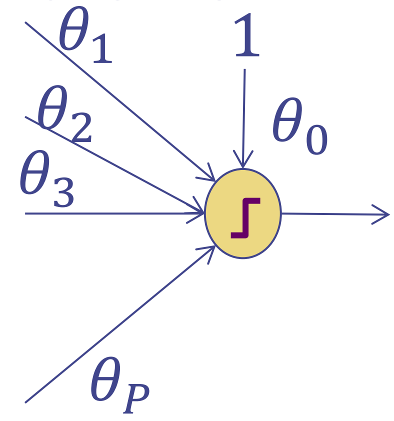


Perceptron (another Neuron)

- A better choice for a classification squashing function is

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



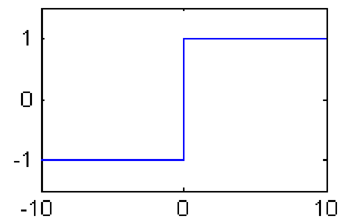


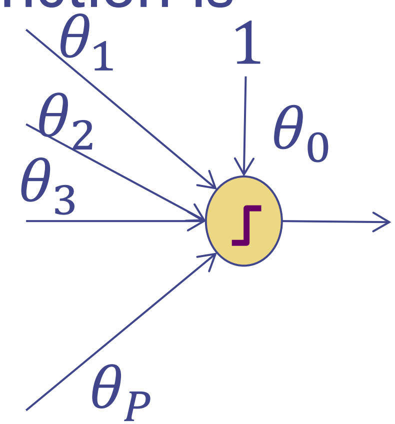
- And a better choice is classification loss

Perceptron (another Neuron)

- A better choice for a classification squashing function is

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





- And a better choice is classification loss

$$Loss^{class}\big(y, f(\boldsymbol{x};\theta)\big) = step\big(-yf(\boldsymbol{x};\theta)\big)$$

$$step(z) = \begin{cases} 1 & z > 0 \\ 0 & otherwise \end{cases}$$

- What does this $R(\theta)$ function look like?

$$R^{class}(\theta) = \frac{1}{N}\sum_{i=1}^{N} step(-y_i\theta^T\boldsymbol{x}_i) = \frac{1}{4N}\sum_{i=1}^{N}(y_i - g(\theta^T\boldsymbol{x}_i))^2$$

Perceptron & Classification Loss

- Classification loss for the Risk leads to hard minimization
- What does this R(θ) function look like?

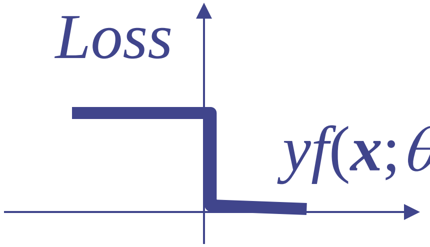
$$R^{class}(\theta) = \frac{1}{N}\sum_{i=1}^{N} step(-y_i \theta^T \boldsymbol{x}_i)$$

- Can't do gradient descent since the gradient is zero except at edges when a label flips

Perceptron & Perceptron Loss

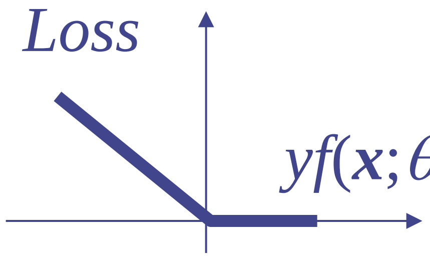
- Instead of Classification Loss

$$R^{class}(\theta) = \frac{1}{N}\sum_{i=1}^{N} step(-y_i\theta^T \boldsymbol{x}_i)$$



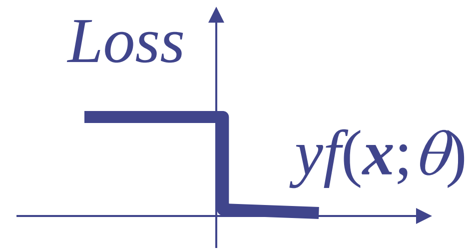
- Consider Perceptron Loss:

$$R^{per}(\theta) = \frac{1}{N}\sum_{i\in misclassified} y_i(\theta^T \boldsymbol{x}_i)$$



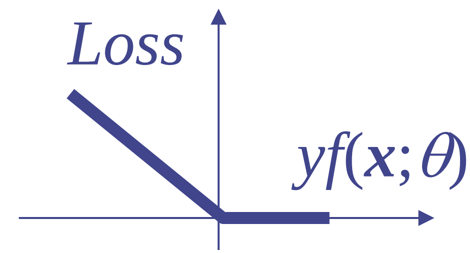
Perceptron & Perceptron Loss

- Instead of Classification Loss



$$R^{class}(\theta) = \frac{1}{N}\sum_{i=1}^{N} step(-y_i \theta^T \boldsymbol{x}_i)$$

- Consider Perceptron Loss:



$$R^{per}(\theta) = -\frac{1}{N}\sum_{i \in misclassified} y_i(\theta^T \boldsymbol{x}_i)$$

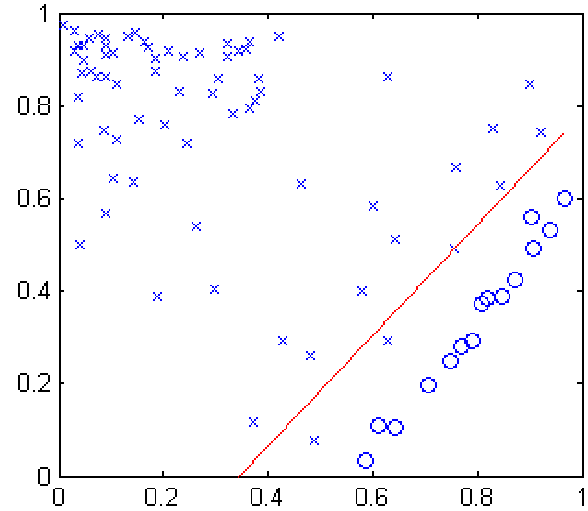
- Instead of staircase-shaped R get smooth piece-wise linear
- Get reasonable gradients for gradient descent

$$\nabla_\theta R^{per}(\theta) = -\frac{1}{N}\sum_{i \in misclassified} y_i \boldsymbol{x}_i$$

$$\theta^{t+1} = \theta^t - \eta \nabla_\theta R^{per}\Big|_{\theta^t} = \theta^t + \eta \frac{1}{N}\sum_{i \in misclassified} y_i \boldsymbol{x}_i$$

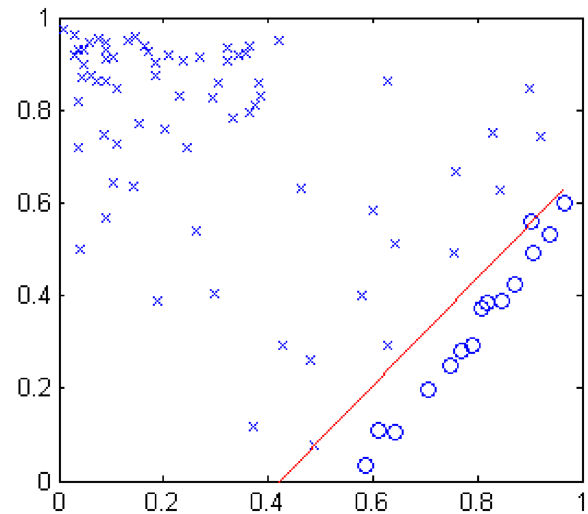
Perceptron vs. Linear Regression

- Linear regression gets close but doesn't do perfectly



classification error = 2
squared error = 0.139

- Perceptron gets zero error



classification error = 0
perceptron err = 0

Stochastic Gradient Descent

- Gradient Descent vs. Stochastic Gradient Descent

- Computing average gradient for all points for taking a step

$$\nabla_\theta R^{per}(\theta) = -\frac{1}{N} \sum_{i \in misclassified} y_i \boldsymbol{x}_i$$

Stochastic Gradient Descent

- Gradient Descent vs. Stochastic Gradient Descent

- Instead of computing the average gradient for all points and then taking a step

$$\nabla_\theta R^{per}(\theta) = -\frac{1}{N} \sum_{i \in misclassified} y_i \boldsymbol{x}_i$$

- Update the gradient for each mis-classified point by itself

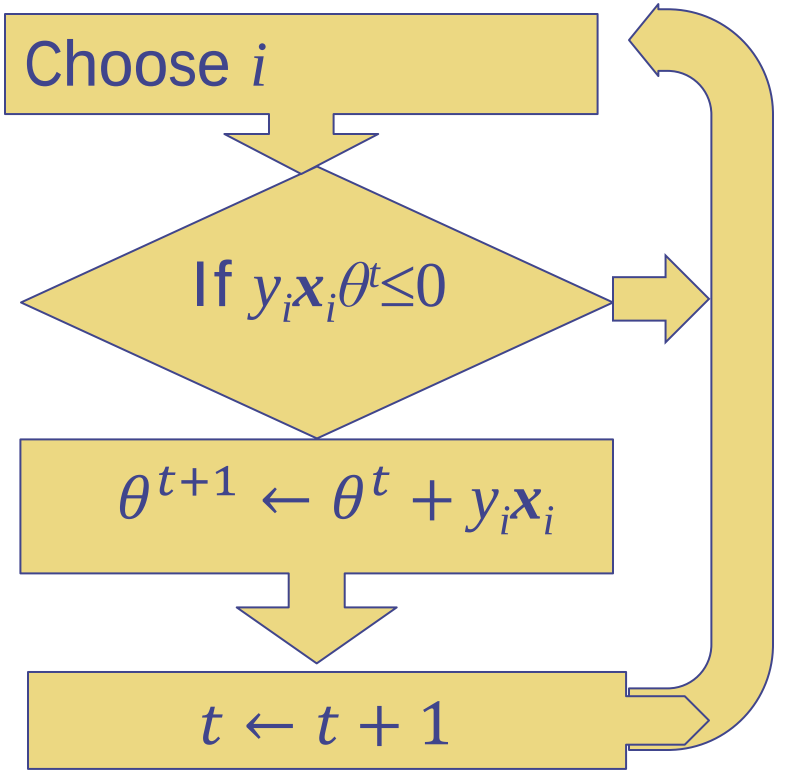
$\nabla_\theta Loss^{per}(\theta) = -y_i \boldsymbol{x}_i$ if i misclassified

- Also, set η to 1

$\theta^{t+1} = \theta^t - \eta \nabla_\theta Loss^{per}|_{\theta^t} = \theta^t + y_i \boldsymbol{x}_i$ if i misclassified

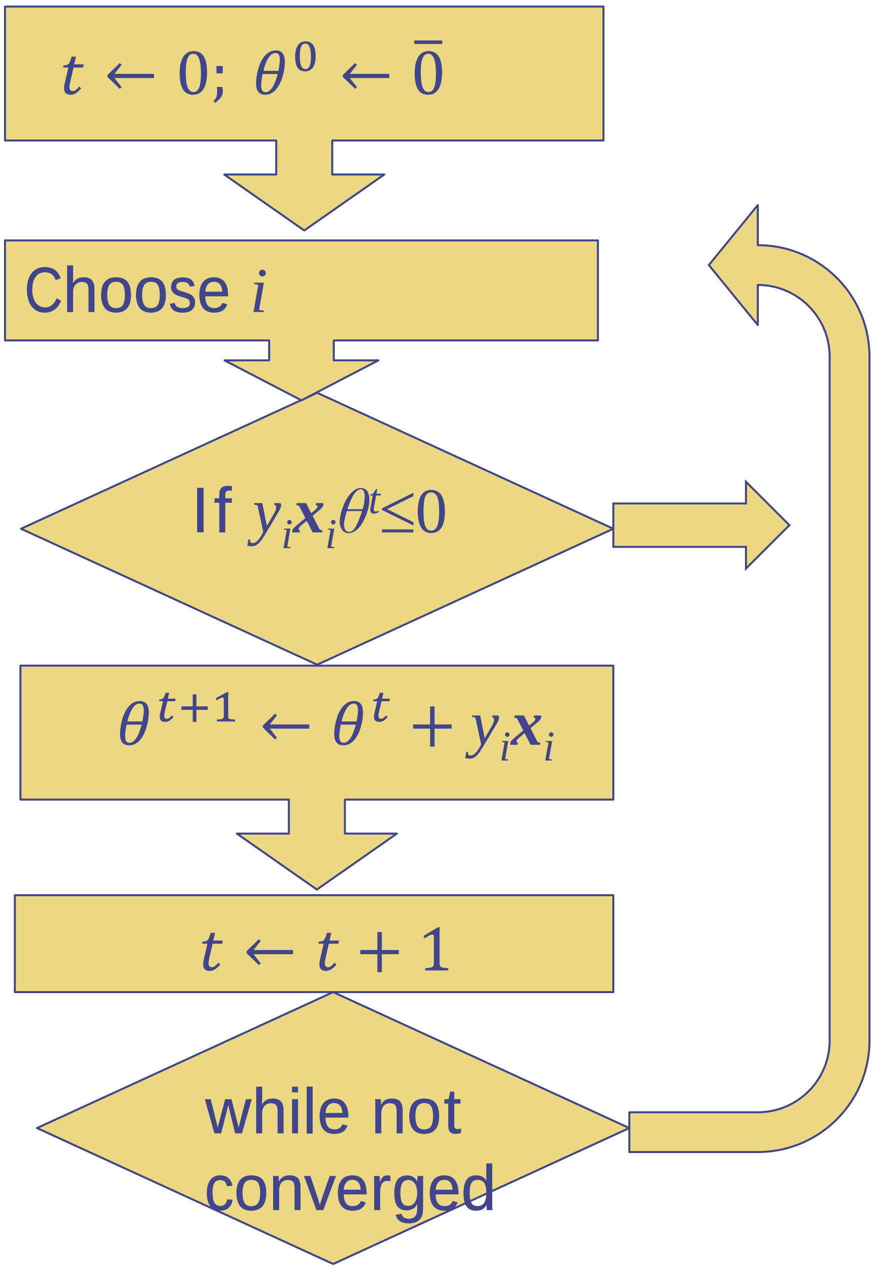
Online Perceptron

- Apply stochastic gradient descent to a perceptron
- Get the “online perceptron” algorithm:



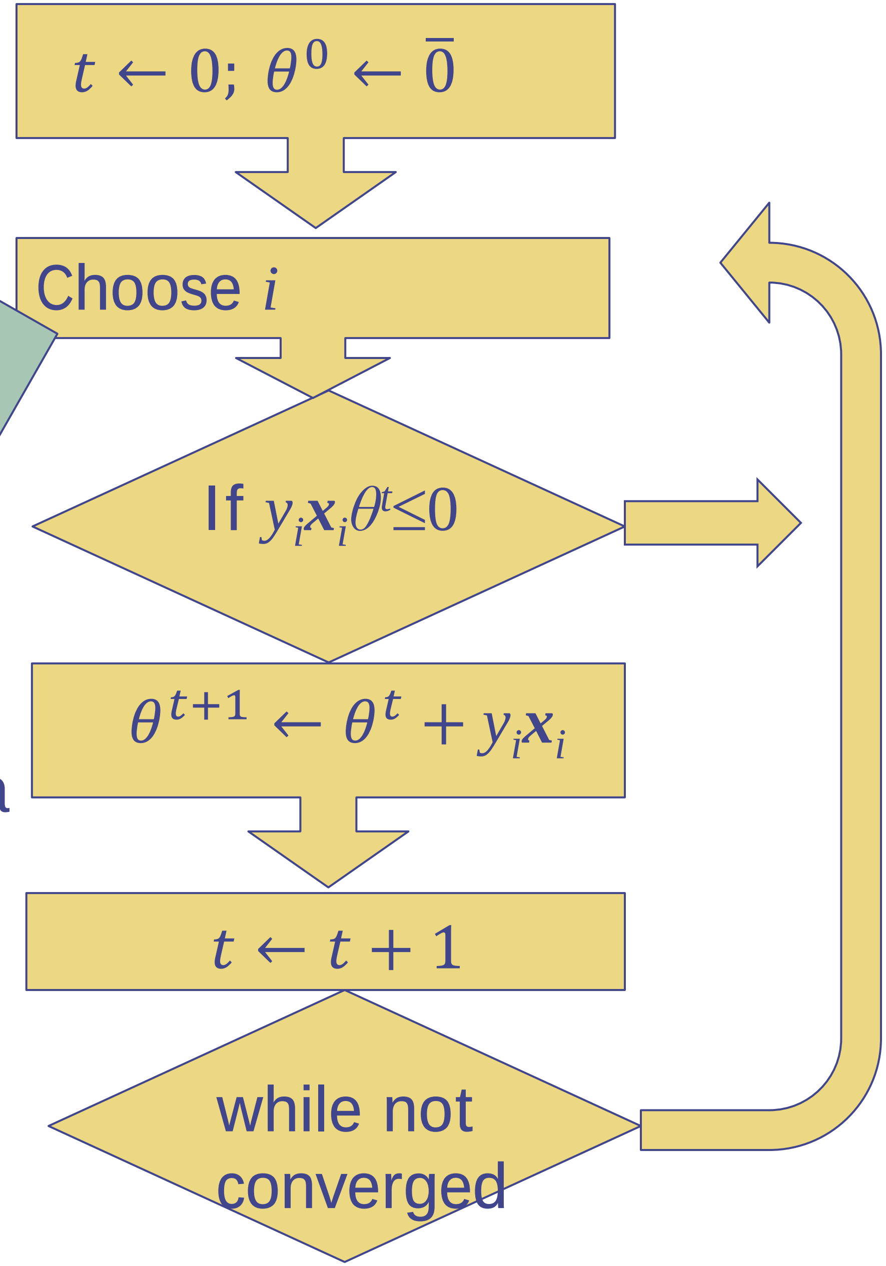
Online Perceptron

- Initialize & repeat



Online Perceptron

- Initialize & repeat



- If the algorithm stops, we have θ that separates data

- t =total number of mistakes

Online Perceptron Theorem

Theorem: the online perceptron algorithm converges to zero error in finite t if we assume

1) all data inside a sphere of radius r: $\|x_i\| \leq r \; \forall i$
2) data is separable with margin γ: $y_i (\theta^*)^T \boldsymbol{x}_i \geq \gamma \forall i$

Online Perceptron Theorem

<u>Theorem</u>: the online perceptron algorithm converges to zero error in finite t if we assume

1) all data inside a sphere of radius r: $\|x_i\| \leq r\ \forall i$
2) data is separable with margin γ: $y_i(\theta^*)^T \boldsymbol{x}_i \geq \gamma \forall i$

<u>Proof:</u>

•Part 1) Look at inner product of current θ^t with θ^*

assume we just updated a mistake on point i:

$$(\theta^*)^T\theta^t = (\theta^*)^T\theta^{t-1} + y_i(\theta^*)^T\boldsymbol{x}_i \geq (\theta^*)^T\theta^{t-1} + \gamma$$

after applying t such updates, we must get:

$$(\theta^*)^T\theta^t \geq t\gamma$$

Online Perceptron Proof

- Part 1) $(\theta^*)^T\theta^t \geq t\gamma$
- Part 2) $||\theta^t||^2 =$

Online Perceptron Proof

- Part 1) $(\theta^*)^T\theta^t \geq t\gamma$
- Part 2) $\|\theta^t\|^2 = \|\theta^{t-1} + y_i\boldsymbol{x}_i\|^2 = \|\theta^{t-1}\|^2 + 2y_i(\theta^{t-1})^T\boldsymbol{x}_i + \|\boldsymbol{x}_i\|^2$
  $\leq \|\theta^{t-1}\|^2 + \|\boldsymbol{x}_i\|^2 \leq \|\theta^{t-1}\|^2 + r^2 \leq tr^2$

since only update mistakes
middle term is negative

Online Perceptron Proof

- Part 1) $(\theta^*)^T\theta^t \geq t\gamma$
- Part 2) $\|\theta^t\|^2 = \|\theta^{t-1} + y_i\boldsymbol{x}_i\|^2 = \|\theta^{t-1}\|^2 + 2y_i(\theta^{t-1})^T\boldsymbol{x}_i + \|\boldsymbol{x}_i\|^2$
  $\leq \|\theta^{t-1}\|^2 + \|\boldsymbol{x}_i\|^2 \leq \|\theta^{t-1}\|^2 + r^2 \leq tr^2$

  since only update mistakes middle term is negative
- Part 3) Angle between optimal & current solution

$$\cos(\theta^*,\theta^t) = \frac{(\theta^*)^T\theta^t}{\|\theta^t\|\|\theta^*\|} \geq \frac{t\gamma}{\|\theta^t\|\|\theta^*\|} \geq \frac{t\gamma}{\sqrt{tr^2}\|\theta^*\|}$$

apply part 1 then part 2

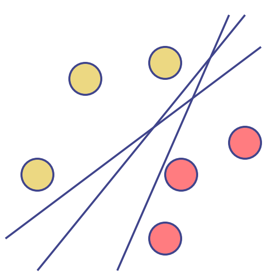
- Since $cos \leq 1$, $\frac{t\gamma}{\sqrt{tr^2}} \leq 1$, thus $t \leq \frac{r^2}{\gamma^2}\|\theta^*\|^2$

...so t is finite!

Minimum Training Error?

- Is minimizing Empirical Risk the right thing?
- Are Perceptrons & Neural Networks giving the best classifier?

- Perceptrons are giving a bunch of solutions:

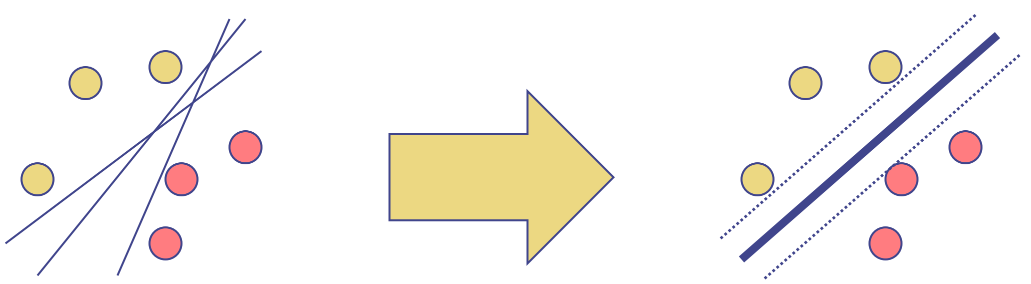


Minimum Training Error?

- Is minimizing Empirical Risk the right thing?
- Are Perceptrons & Neural Networks giving the best classifier?

- We are getting: minimum training error
  not minimum testing error

- Perceptrons are giving a bunch of solutions:



... a better solution → gap tolerant classifier

Empirical Risk Minimization

- Recall linear classifier $f(\boldsymbol{x};\theta) = sign(\theta^T\boldsymbol{x} + \theta_0) \in \{-1,1\}$
- Recall ERM: $R_{emp}(\theta) = \frac{1}{N}\sum_i^N Loss(y_i, f(\boldsymbol{x}_i;\theta)) \in [0,1]$
- Some loss functions: quadratic: $Loss(y,\boldsymbol{x},\theta) = \frac{1}{2}(y - f(\boldsymbol{x};\theta))^2$

  linear: $Loss(y,\boldsymbol{x},\theta) = |y - f(\boldsymbol{x};\theta)|$

  binary: $Loss(y,\boldsymbol{x},\theta) = step(-yf(x;\theta))$
- Empirical $R_{emp}(\theta)$ *approximates* the true risk (expected error)

$$R(\theta) = E_P\{Loss(y,\boldsymbol{x},\theta)\} = \int_{\boldsymbol{X}\times Y} P(\boldsymbol{x},y) Loss(y,\boldsymbol{x},\theta)\, dxdy \in [0,1]$$

Empirical Risk Minimization

- Recall ERM: $R_{emp}(\theta) = \frac{1}{N}\sum_i^N Loss(y_i, f(\boldsymbol{x}_i;\theta)) \in [0,1]$
- Empirical $R_{emp}(\theta)$ *approximates* the true risk (expected error)

$$R(\theta) = E_P\{Loss(y, \boldsymbol{x}, \theta)\} = \int_{\boldsymbol{X}\times Y} P(\boldsymbol{x}, y) Loss(y, \boldsymbol{x}, \theta)\, dxdy \in [0,1]$$

- But, we don't know the true $P(\boldsymbol{x},y)$!
- Good news: for any θ, if infinite data, by *law of large numbers*:

$$\lim_{n\to\infty} R_{emp}(\theta) = R(\theta)$$

- Bad news: ERM may not converge to optimum even if $N\to\infty$:

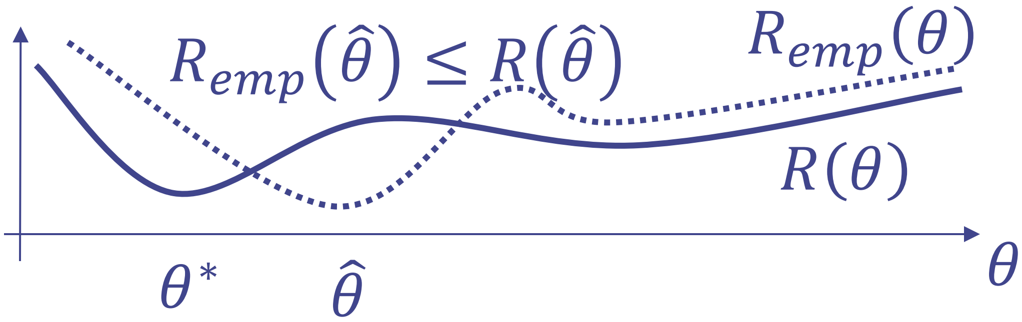
$$argmin_\theta R_{emp}(\theta) \neq argmin_\theta R(\theta)$$

...ERM is not consistent

Bounding the True Risk

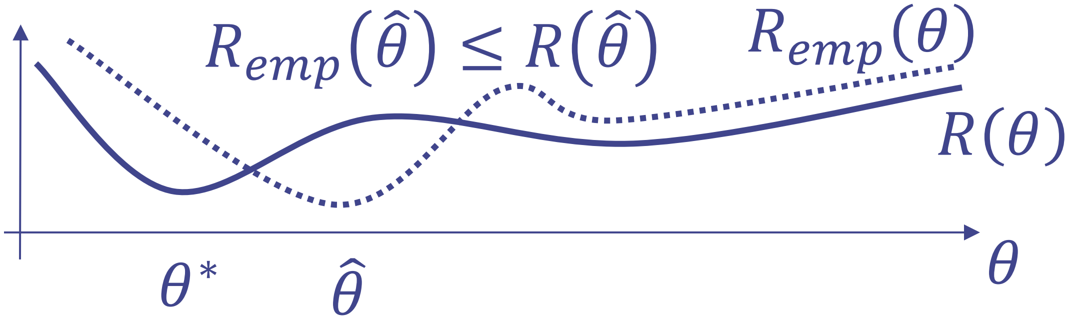
Bounding the True Risk

- ERM's risk is not guaranteed since it may do better on training than on test!

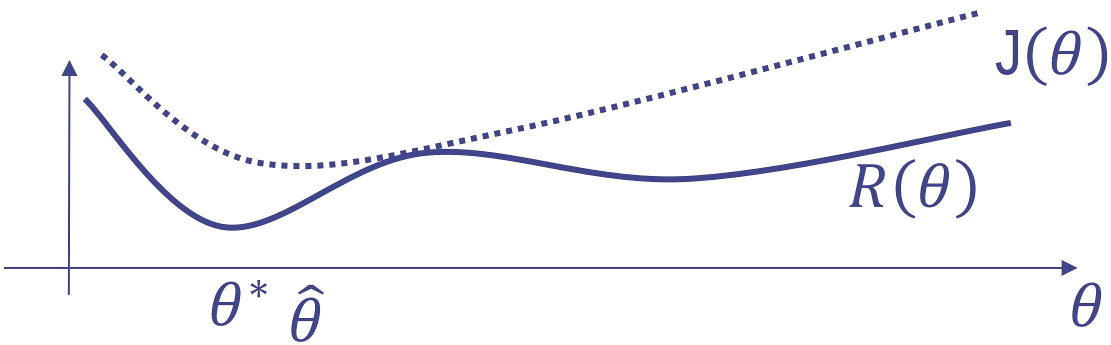


Bounding the True Risk

- ERM's risk is not guaranteed since it may do better on training than on test!



- Idea: add a prior or regularizer to $R_{emp}(\theta)$
- Define capacity or confidence $C(\theta)$ which favors simpler θ
- If $J(\theta) = R_{emp}(\theta) + C(\theta) \geq R(\theta)$, then it is guaranteed risk



- After train, can guarantee future error rate is $\leq \min_{\theta} J(\theta)$
- Structural Risk Minimization: minimize risk bound $J(\theta)$

Bound the True Risk with VCD

- Idea: Rely on the capacity of the classifier class $f(.;\theta)$
  - $h \triangleq$ # of datasets it can perfectly classify ($\neq$#parameters!)
  - Independent of the true $P(x,y)$ so gives *worst case bound*

- Theorem (Vapnik):

With probability 1-η where $\eta \in [0,1]$, $R(\theta) \leq J(\theta)$ where:

$$J(\theta) = R_{emp}(\theta) + \frac{2h\log\left(\frac{2eN}{h}\right)+2\log\left(\frac{4}{\eta}\right)}{N}\left(1 + \sqrt{1 + \frac{NR_{emp}(\theta)}{h\log\left(\frac{2eN}{h}\right)+\log\left(\frac{4}{\eta}\right)}}\right)$$

N = number of data points

h = Vapnik-Chervonenkis (VC) dimension (1970's)

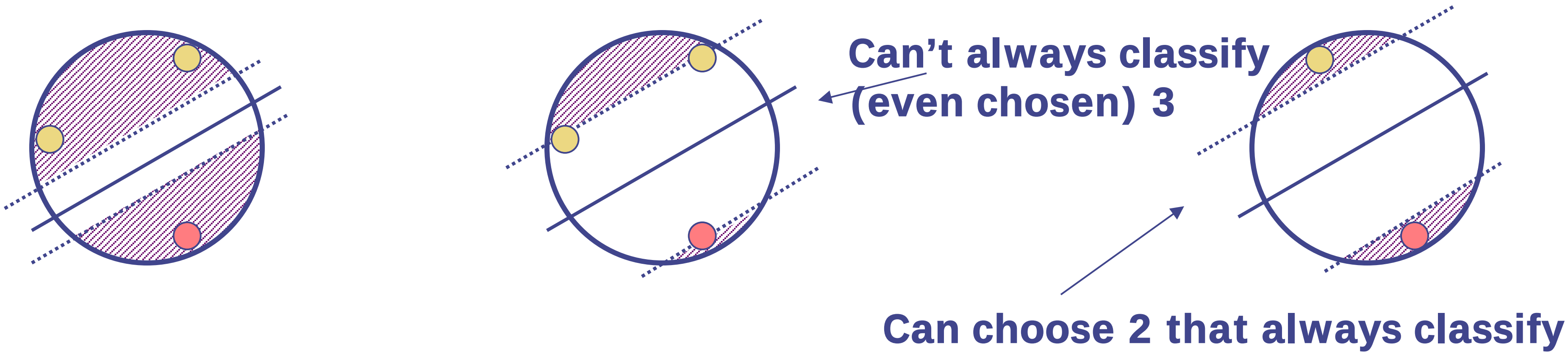
measure classifying ability of a function family

True Risk Bound & Gaps

- Wider gap (w.r.t. universe) means the function family is weaker

True Risk Bound & Gaps

- Wider gap (w.r.t. universe) means the function family is weaker



- Best bound = weakest family = widest gap

Summary

- Perceptrons:
  - Shoot for perfect classification
  - Optimized by online (stochastic) Gradient Descent
  - Convergence guaranteed
- Gaps required to guard against overfit