CMSC 510 – L05
Regularization Methods for Machine Learning

Instructor:
Dr. Tom Arodz

Big picture…

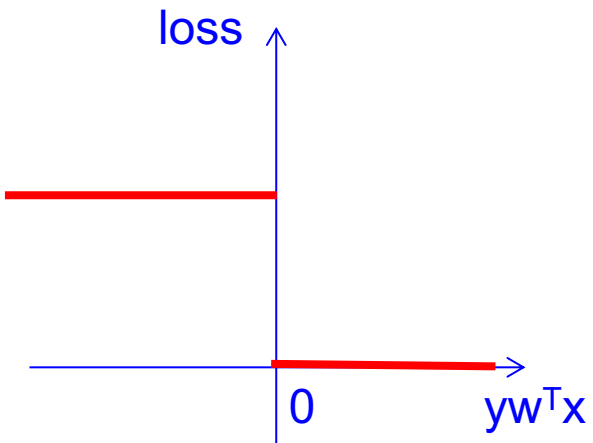
- Choose predictive model
  - We're focusing on h(x)=wx+b (linear models)
- Choose loss function
  - E.g. loss=$(y-h(x))^2$
  - Loss depends on y,x – given to us (training set)
  - Loss also depends on w,b – we need to find those
- Choose the algorithm for finding model parameters (w,b for linear models) that lead to low avg. loss
  - Gradient descent is our default choice
    - $w_{next} = w_{current}$ - d loss / d w

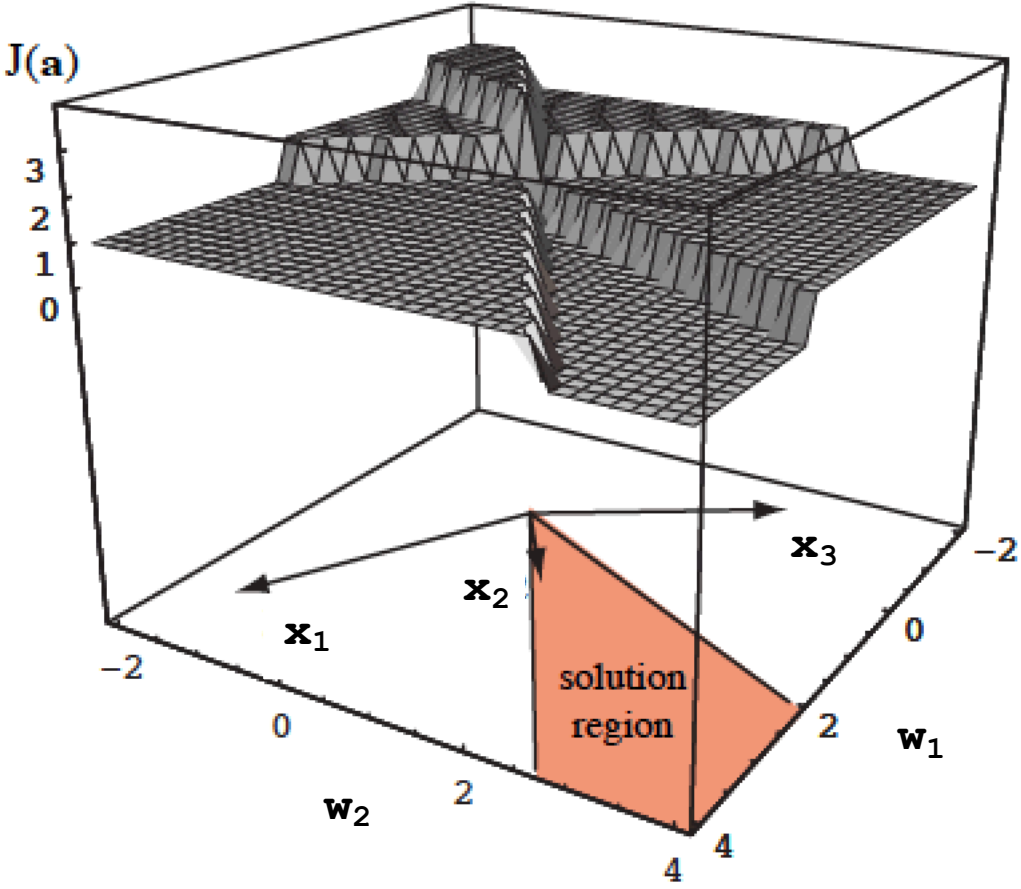
Recap: 0/1 loss

- **Present a sample x and predict:**

$$h(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\mathbf{x} - w_0\right)$$

- Why not use 0/1 loss?
- It's flat! We don't know in which direction to move to get a better solution!
- At each possible w, we have # of misclassified points as our empirical risk





Recap: Perceptron loss

- **Present a sample x and predict:**

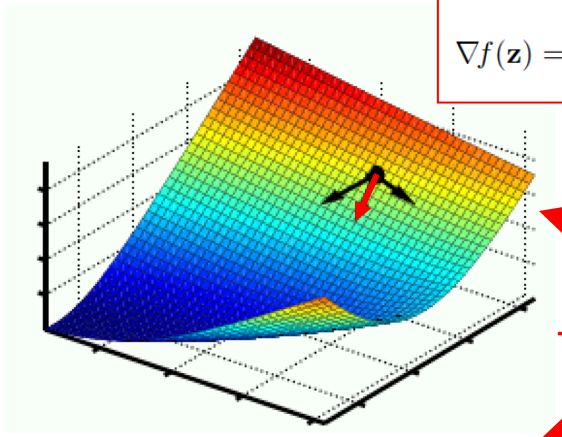
$$h(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\mathbf{x} - w_0\right)$$

- **Compare true class *y* with predicted class h(x)**
- **If prediction is wrong, update weights**

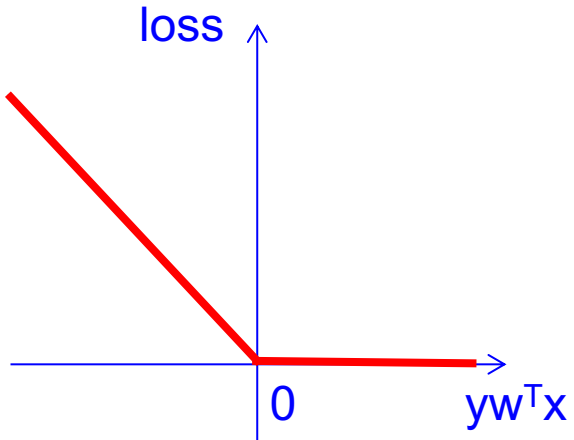
$$\mathbf{w}^\dagger{}_{t+1} = \mathbf{w}^\dagger{}_t + 2cy\mathbf{x}^\dagger$$

- Minimize empirical risk of perceptron loss:
- By modifying w after seeing each sample z

$$\nabla f(\mathbf{z}) = \left(\frac{\partial}{\partial x_1} f(\mathbf{x})|_{\mathbf{x}=\mathbf{z}}, \ldots, \frac{\partial}{\partial x_n} f(\mathbf{x})|_{\mathbf{x}=\mathbf{z}}\right)$$



–gradient (stochastic)



$$\mathbf{w}^\dagger{}_{t+1} = \mathbf{w}^\dagger{}_t - 2c\left.\frac{\partial \ell(h, \mathbf{z})}{\partial \mathbf{w}^\dagger}\right|_{\mathbf{w}^\dagger = \mathbf{w}^\dagger{}_t}$$

$$\ell(h, \mathbf{z}) = \begin{cases} 0 & \text{for } y\mathbf{w}^{\dagger T}\mathbf{x}^\dagger \geq 0 \\ -y\mathbf{w}^{\dagger T}\mathbf{x}^\dagger & \text{for } y\mathbf{w}^{\dagger T}\mathbf{x}^\dagger < 0 \end{cases}$$

(stochastic) gradient descent

- Our function to be minimized is (proportional to) empirical risk:
  $f(w)=\sum_i f_i(w) = \sum_i \text{loss}(y_i,x_i,w)$

- Gradient descent (batch learning):
  - in a loop, modify w to minimize f(w)
    - that means we need to evaluate the sum over all samples
    - we use gradient of $\sum_i \text{loss}(y_i,x_i,w)$ w.r.t. w
  - We present all samples, and only then we update the weights

- Stochastic gradient descent (online learning):
  - in a loop, randomly pick a single i, modify w to minimize $f_i(w)$
    - that means we need to evaluate only one sample at a time
    - we use gradient of $\text{loss}(y_i,x_i,w)$ w.r.t. w
  - We present one sample, and immediately update the weights
  - Each step is then a "noisy" gradient, but the noise should average itself out over many iterations
- mini-batch stochastic gradient descent :
  - In between the above: $\sum_i \text{loss}(y_i,x_i,w)$ over e.g. randomly chosen 64 samples from the whole training set

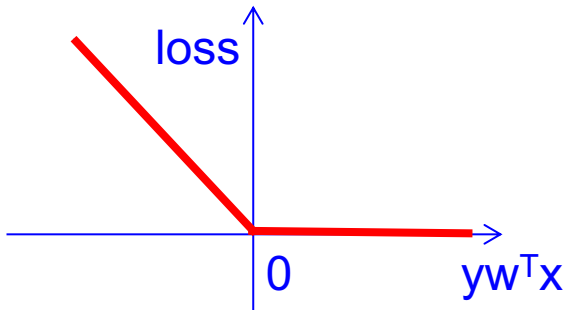
Perceptron

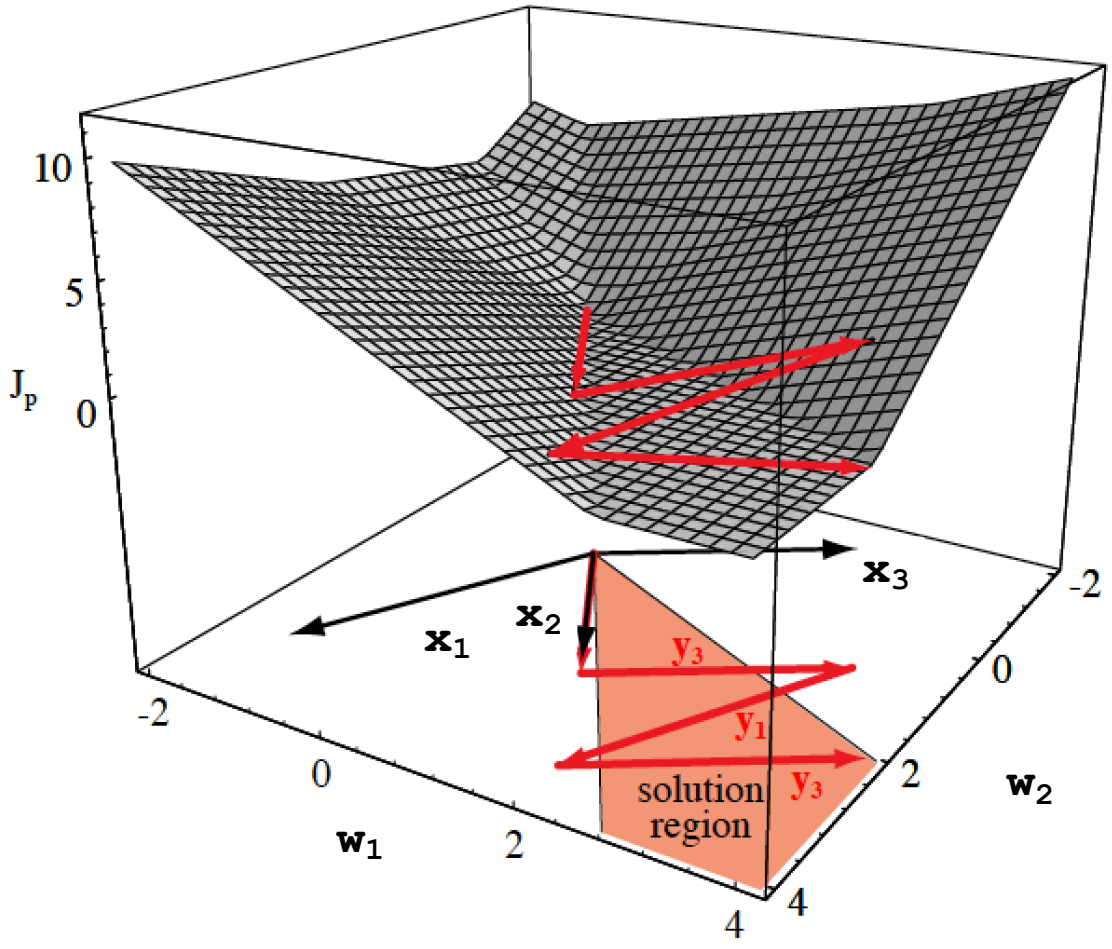
- **Present a sample x and predict:**

$$h(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\mathbf{x} - w_0\right)$$

  - Example: three points
  - Two parameters
  - Surface plot of risk:

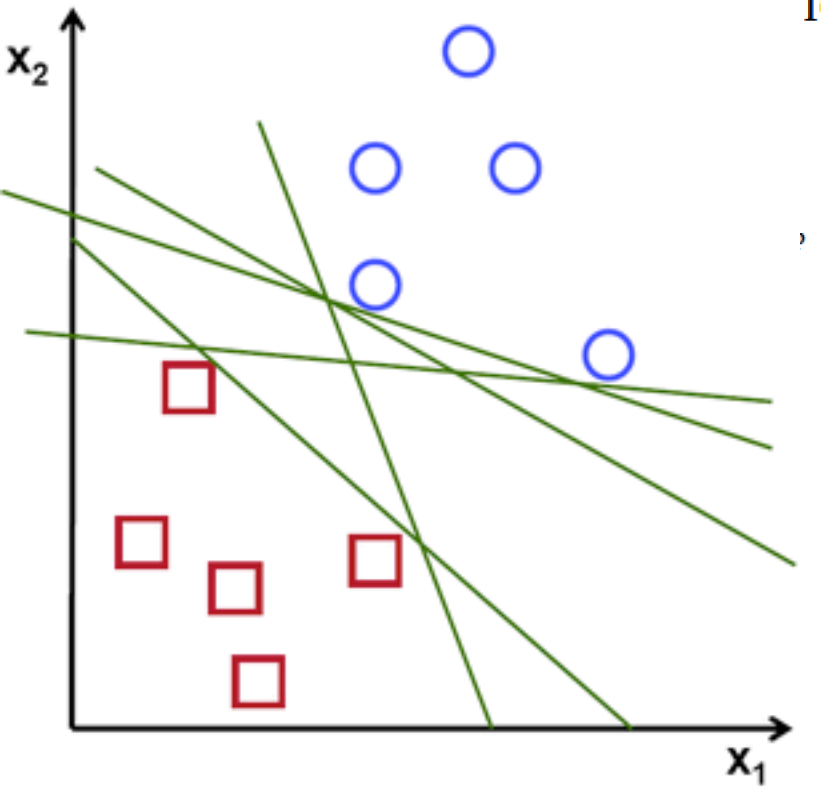
$$\widehat{R}_{S_m}(h) = \frac{1}{m}\sum_{i=1}^{m}\ell\left(h, \mathbf{z}_i\right)$$

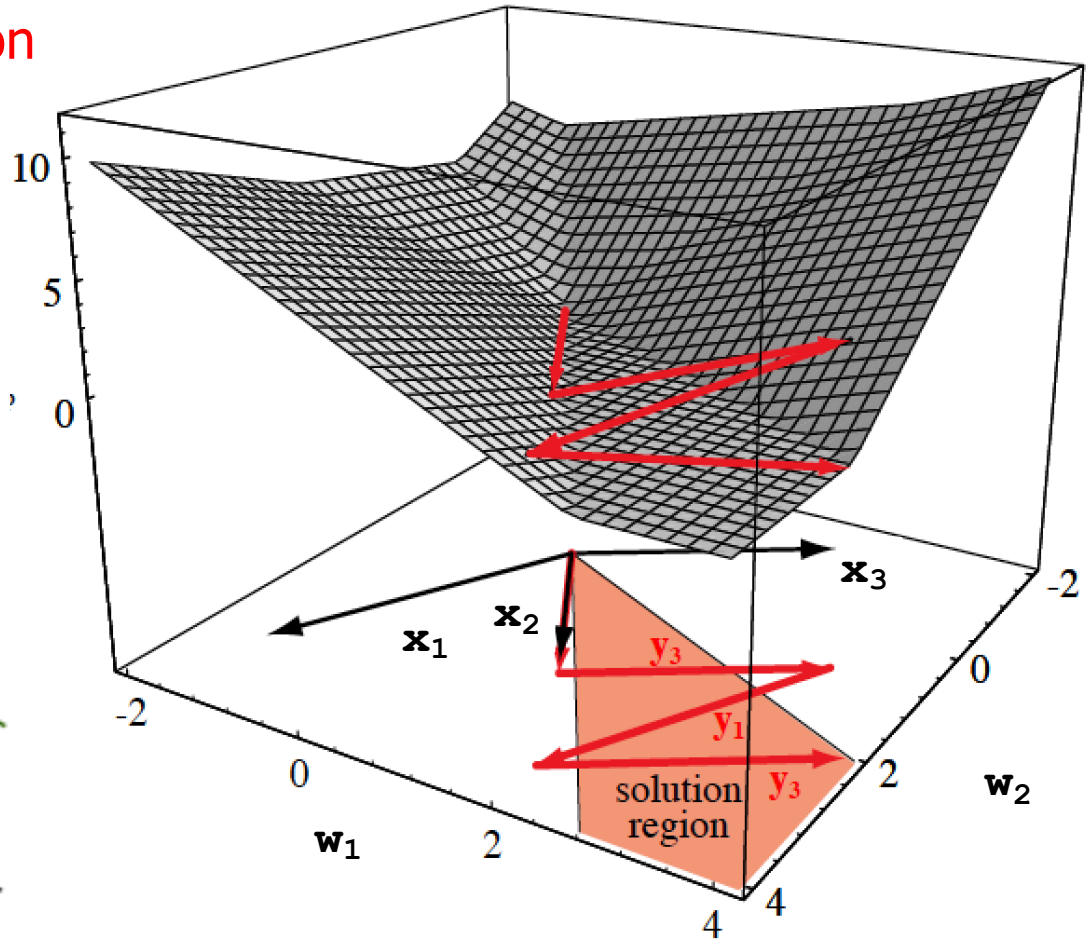




Solution region

- If the two classes can be separated by a straight line (problem/dataset is linearly separable)
  - Then there's infinite number of lines that separate the classes
  - The coefficients w for those lines form the solution region

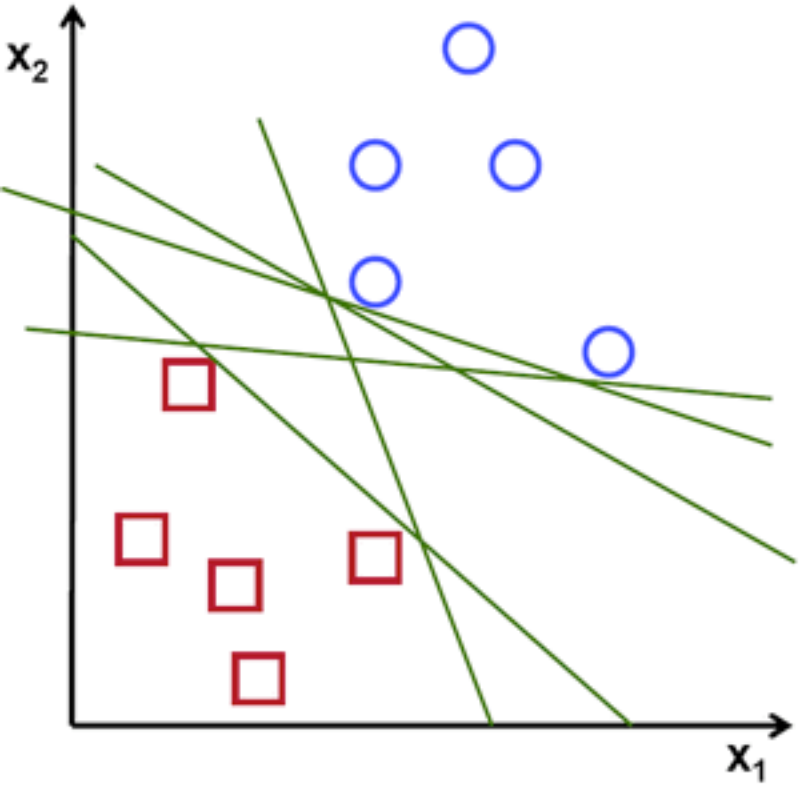




Separable problems

- Some problems/datasets are linearly non-separable

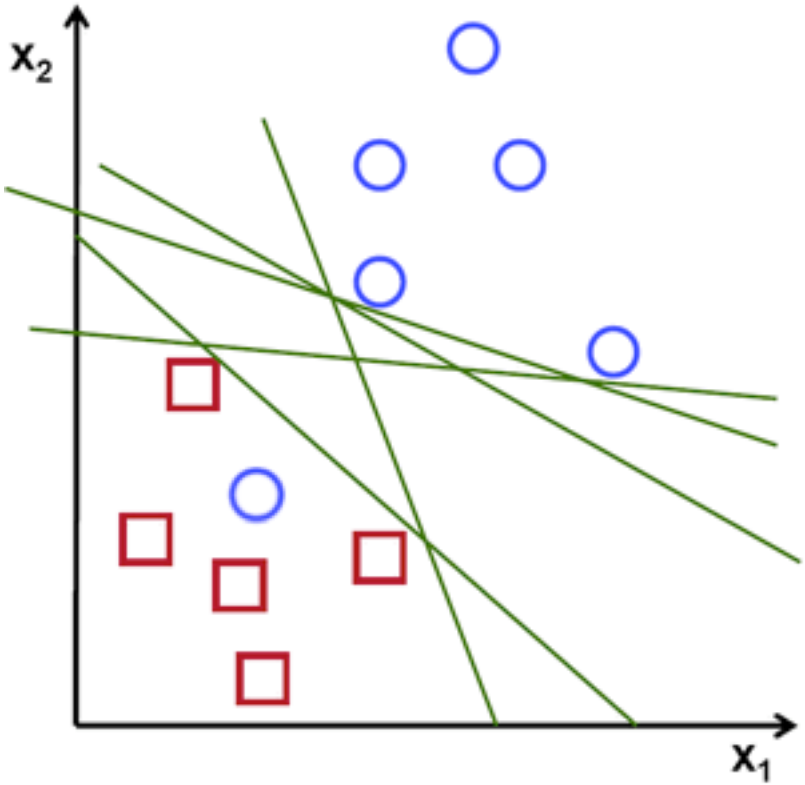
Separable problem



Non-separable problem

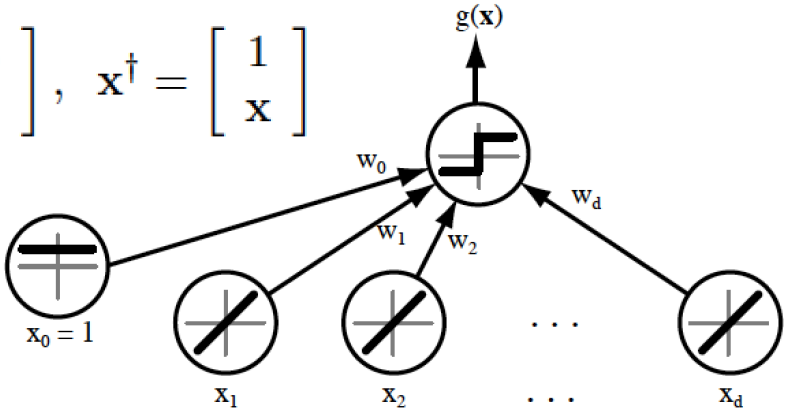
none of the lines works

no other line would work either



Perceptron

$$\mathbf{w}^\dagger = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}, \quad \mathbf{x}^\dagger = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}$$

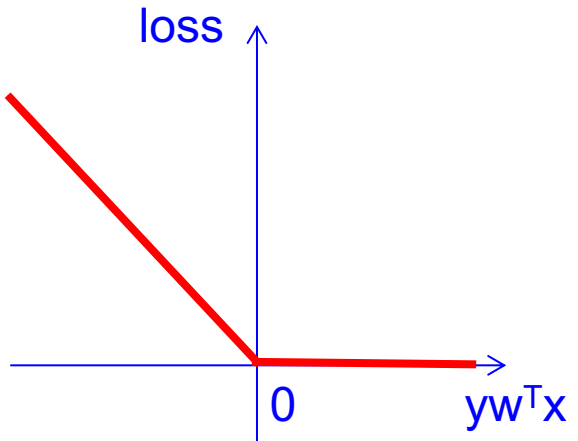


- **Present a sample x and predict:**

$$h(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\mathbf{x} - w_0\right)$$

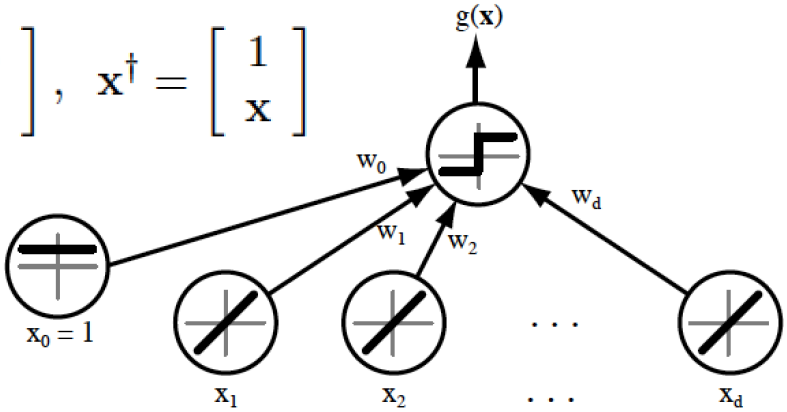
- Some mathematical problems:
  - Perceptron loss is not differentiable at 0
  - We would have to use subgradients instead of gradients
  - Loss is convex, so risk is convex
  - For separable problems, the algorithm eventually converges to optimal solution with 0 risk
  - For non-separable problems, we'll always have errors
    - So the risk will always be >0 ???

$$\ell(h, \mathbf{z}) = \begin{cases} 0 & \text{for } y\mathbf{w}^{\dagger T}\mathbf{x}^\dagger \geq 0 \\ -y\mathbf{w}^{\dagger T}\mathbf{x}^\dagger & \text{for } y\mathbf{w}^{\dagger T}\mathbf{x}^\dagger < 0 \end{cases}$$



Perceptron

$$\mathbf{w}^\dagger = \begin{bmatrix} w_0 \\ \mathbf{w} \end{bmatrix}, \quad \mathbf{x}^\dagger = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}$$

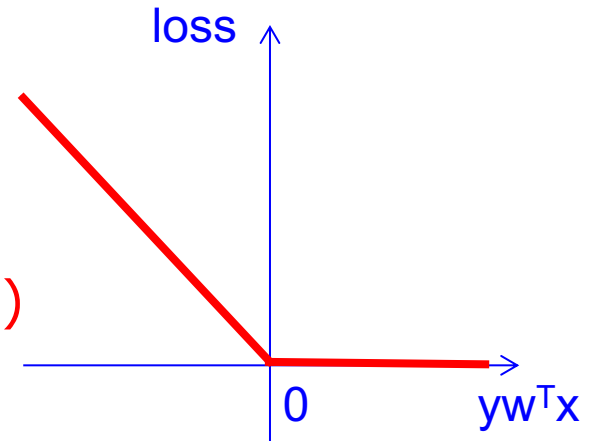


- **Present a sample x and predict:**

$$h(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\mathbf{x} - w_0\right)$$

- Some mathematical problems:
  - Perceptron loss is not differentiable at 0
  - We would have to use subgradients instead of gradients
  - Loss is convex, so risk is convex
  - For separable problems, the algorithm eventually converges to optimal solution with 0 risk
  - For non-separable problems, risk has a single global minimum (with risk=0) at w=0 (useless!!!)

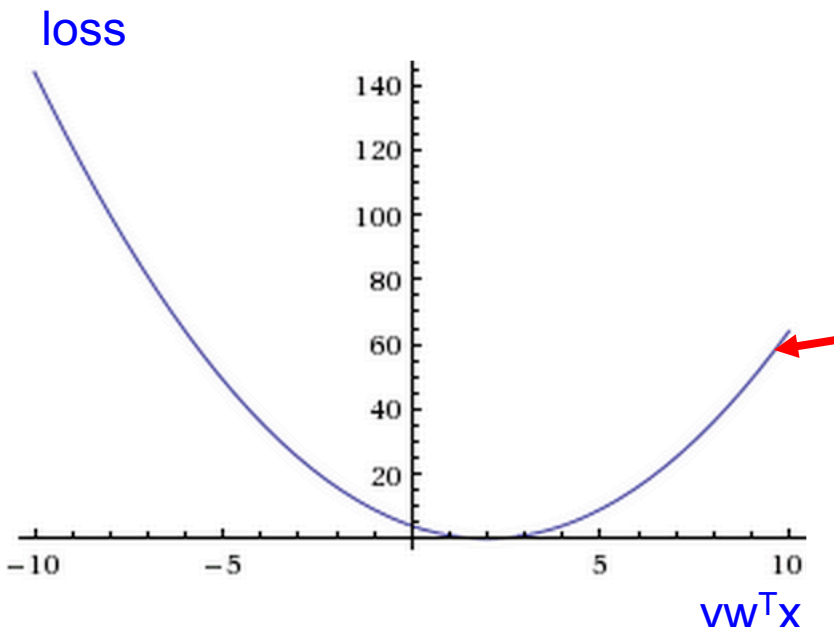
$$\ell(h, \mathbf{z}) = \begin{cases} 0 & \text{for } y\mathbf{w}^{\dagger T}\mathbf{x}^\dagger \geq 0 \\ -y\mathbf{w}^{\dagger T}\mathbf{x}^\dagger & \text{for } y\mathbf{w}^{\dagger T}\mathbf{x}^\dagger < 0 \end{cases}$$



Losses so far

All have their problems:

- 0/1 loss: flat
- Perceptron loss: ”no prediction”
- MSE for classification: penalty for correct predictions
  - Let’s try to address that

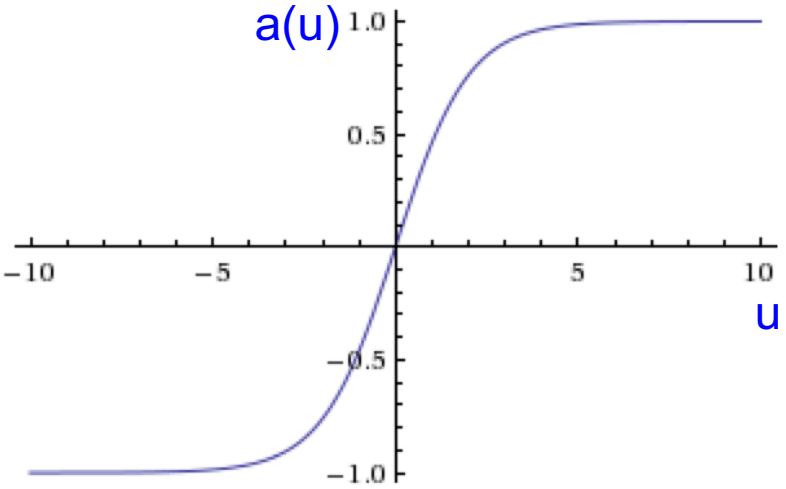


Differentiable perceptron (y=-1/1)

- Let's limit the predictions to [-1,1]: $h(x)=a(w^T x)$
  - If $h(x)>0$ we predict +1
  - If $h(x)<0$ we predict -1

$$a(u) = \frac{2}{1+e^{-u}} - 1$$

a(u) is "activation function"



<- bipolar sigmoid activation function

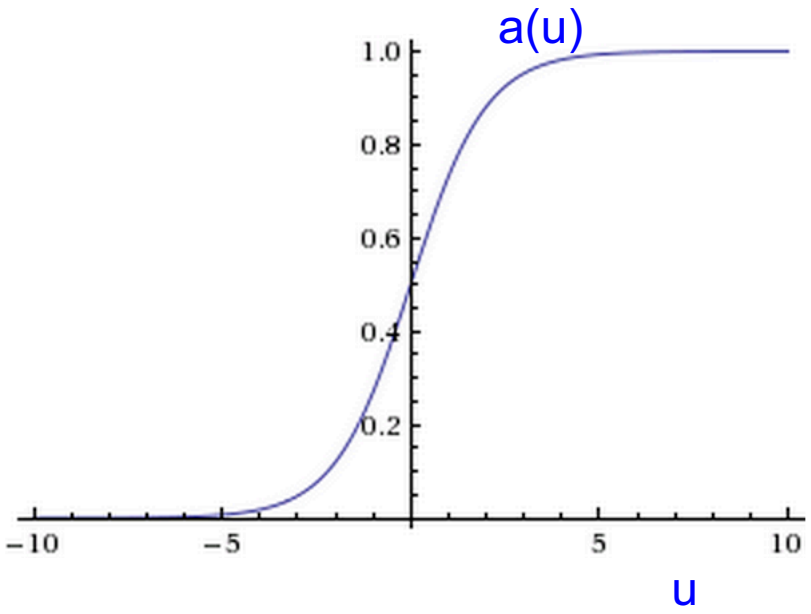
- Derivative of a(u) with respect to u is (verify at home!):

$$a'(u) = \frac{2e^{-u}}{(1+e^{-u})^2} = \frac{1}{2}\left(1 - a^2(u)\right)$$

Differentiable perceptron (y=0/1)

- Another variant: $h(x)=a(w^T x)$
  - If $h(x)>0.5$ we predict +1
  - If $h(x)<0.5$ we predict 0

$$a(u) = \frac{1}{1+e^{-u}}$$



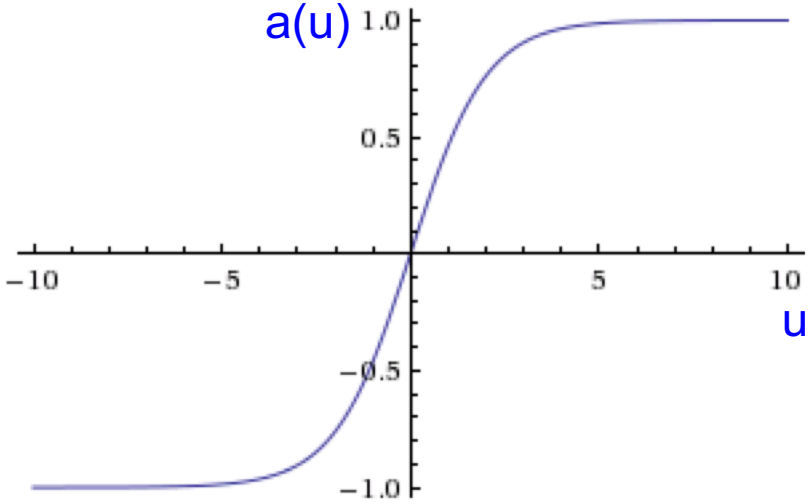
a(u) is “activation function”

<- unipolar sigmoid activation function

Differentiable perceptron

- Let's make h differentiable: $h(x)=a(w^T x)$
  - If $h(x)>0$ we predict +1
  - If $h(x)<0$ we predict -1

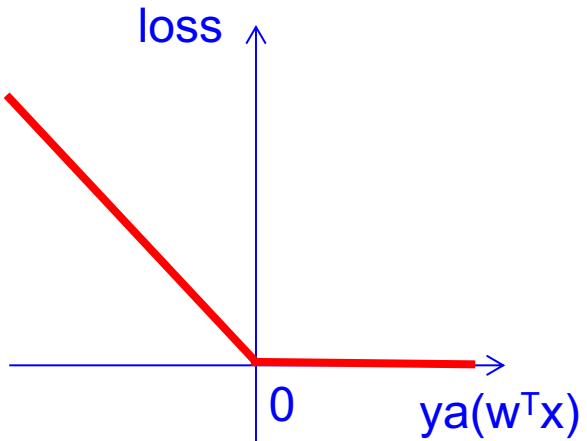
$$a(u) = \frac{2}{1+e^{-u}} - 1$$



a(u) is "activation function"

<- sigmoid activation function

- If we use the loss: $\max(0,-ya(w^T x))$
  - Still the same problem:
    - No prediction (w=0) is optimal

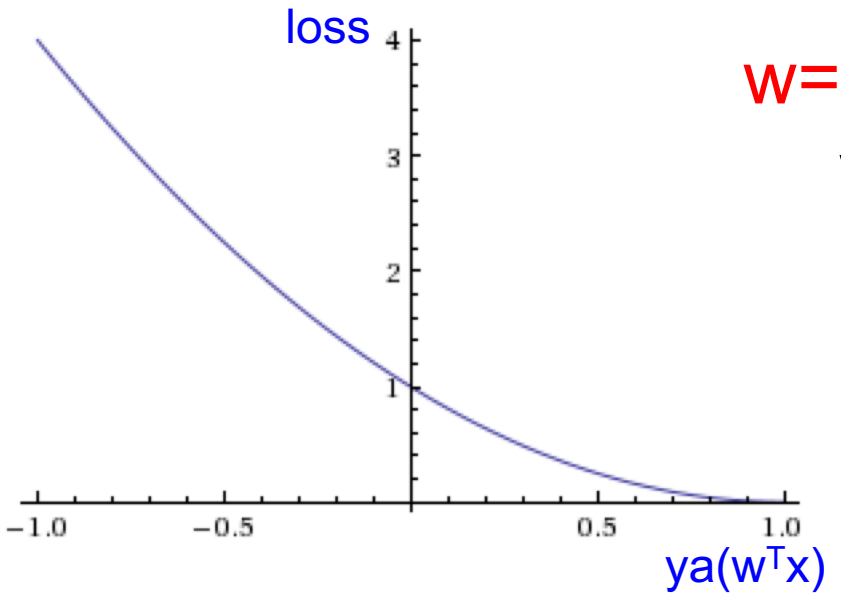


Differentiable perceptron

- $h(x)=a(w^T x)$ $\quad a(u) = \frac{2}{1+e^{-u}} - 1$
- Quadratic loss on sigmoid predictions:

$$\ell(h, \mathbf{z}) = \left(y - a\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2 = \left(1 - ya\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2$$

How did we get this?



w=0 is no longer the best solution!

we won't be stuck with "no prediction" as a solution that is always a global minimum

Differentiable perceptron

- Assuming: $$a(u) = \frac{2}{1+\mathrm{e}^{-u}} - 1$$

$$\ell(h, \mathbf{z}) = \left(y - a\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2 = \left(1 - ya\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2$$

- We can start deriving the update of weights:

$$\mathbf{w}^{\dagger}{}_{t+1} = \mathbf{w}^{\dagger}{}_{t} - \frac{c}{2}\left.\frac{\partial \ell(h, \mathbf{z})}{\partial \mathbf{w}^{\dagger}}\right|_{\mathbf{w}^{\dagger}=\mathbf{w}^{\dagger}{}_{t}}$$

$$\mathbf{w}^{\dagger}{}_{t+1} = \mathbf{w}^{\dagger}{}_{t} - \frac{c}{2}\left.\frac{\partial \left(1 - ya\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2}{\partial \mathbf{w}^{\dagger}}\right|_{\mathbf{w}^{\dagger}=\mathbf{w}^{\dagger}{}_{t}}$$

$$\mathbf{w}^{\dagger}{}_{t+1} = \mathbf{w}^{\dagger}{}_{t} - \frac{c}{2}\left.\frac{\partial \left(1 - ya\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2}{\partial a\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)}\frac{\partial a\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)}{\partial \mathbf{w}^{\dagger}}\right|_{\mathbf{w}^{\dagger}=\mathbf{w}^{\dagger}{}_{t}}$$

Differentiable perceptron

- Assuming: $a(u) = \frac{2}{1+e^{-u}} - 1$ $\quad a'(u) = \frac{2e^{-u}}{(1+e^{-u})^2} = \frac{1}{2}(1-a^2(u))$

- Continuing the derivation of the update of weights:

$$\mathbf{w}^{\dagger}{}_{t+1} = \mathbf{w}^{\dagger}{}_{t} - \frac{c}{2} \frac{\partial \left(1 - ya\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2}{\partial a\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)} \frac{\partial a\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)}{\partial \mathbf{w}^{\dagger}} \Bigg|_{\mathbf{w}^{\dagger}=\mathbf{w}^{\dagger}{}_{t}}$$

$$\mathbf{w}^{\dagger}{}_{t+1} = \mathbf{w}^{\dagger}{}_{t} - \frac{c}{2}\left[2y^2 a\left(\mathbf{w}^{\dagger T}_{t}\mathbf{x}^{\dagger}\right) - 2y\right] \frac{\partial a\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)}{\partial \mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}} \frac{\partial \mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}}{\partial \mathbf{w}^{\dagger}} \Bigg|_{\mathbf{w}^{\dagger}=\mathbf{w}^{\dagger}{}_{t}}$$

$$\mathbf{w}^{\dagger}{}_{t+1} = \mathbf{w}^{\dagger}{}_{t} + c\left[y - a\left(\mathbf{w}^{\dagger T}_{t}\mathbf{x}^{\dagger}\right)\right]\frac{1}{2}\left[1 - a^2\left(\mathbf{w}^{\dagger T}_{t}\mathbf{x}^{\dagger}\right)\right]\mathbf{x}^{\dagger}$$

$$\mathbf{w}^{\dagger}{}_{t+1} = \mathbf{w}^{\dagger}{}_{t} + \frac{c}{2}\left[y - a\left(\mathbf{w}^{\dagger T}_{t}\mathbf{x}^{\dagger}\right)\right]\left[1 - a^2\left(\mathbf{w}^{\dagger T}_{t}\mathbf{x}^{\dagger}\right)\right]\mathbf{x}^{\dagger}$$

Differentiable perceptron

- a perceptron with differentiable activation function $a(w^T x)$:
- quadratic loss:

$$\ell(h, \mathbf{z}) = \left(y - a\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2 = \left(1 - ya\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2$$

- This weight update rule is called delta rule:

$$\mathbf{w}^{\dagger}{}_{t+1} = \mathbf{w}^{\dagger}{}_t - \frac{c}{2}\left.\frac{\partial \ell(h, \mathbf{z})}{\partial \mathbf{w}^{\dagger}}\right|_{\mathbf{w}^{\dagger}=\mathbf{w}^{\dagger}{}_t} = \mathbf{w}^{\dagger}{}_t + c\left[y - a\left(\mathbf{w}^{\dagger}{}_t^T \mathbf{x}^{\dagger}\right)\right] a'\left(\mathbf{w}^{\dagger}{}_t^T \mathbf{x}^{\dagger}\right) \mathbf{x}^{\dagger}$$

- In our specific case of activation function $a(u) = \frac{2}{1+\mathrm{e}^{-u}} - 1$ we get:

$$\mathbf{w}^{\dagger}{}_{t+1} = \mathbf{w}^{\dagger}{}_t - \frac{c}{2}\left.\frac{\partial \ell(h, \mathbf{z})}{\partial \mathbf{w}^{\dagger}}\right|_{\mathbf{w}^{\dagger}=\mathbf{w}^{\dagger}{}_t} = \mathbf{w}^{\dagger}{}_t + \frac{c}{2}\left[y - a\left(\mathbf{w}^{\dagger}{}_t^T \mathbf{x}^{\dagger}\right)\right]\left[1 - a^2\left(\mathbf{w}^{\dagger}{}_t^T \mathbf{x}^{\dagger}\right)\right] \mathbf{x}^{\dagger}$$

Differentiable perceptron $h(x)=a(w^T x)$

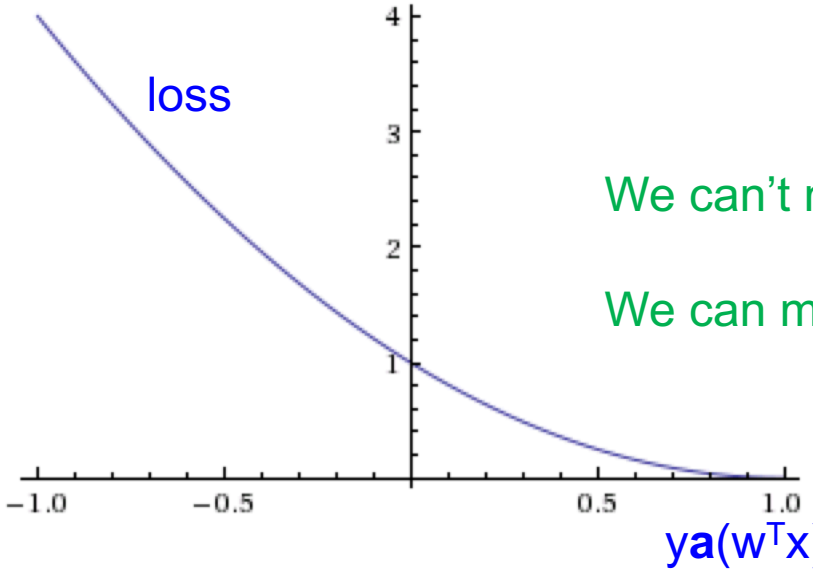
- $h(x)=a(w^T x)$

$$a(u) = \frac{2}{1+e^{-u}} - 1$$

- Loss as a function of y and of $a(w^T x)$ is:

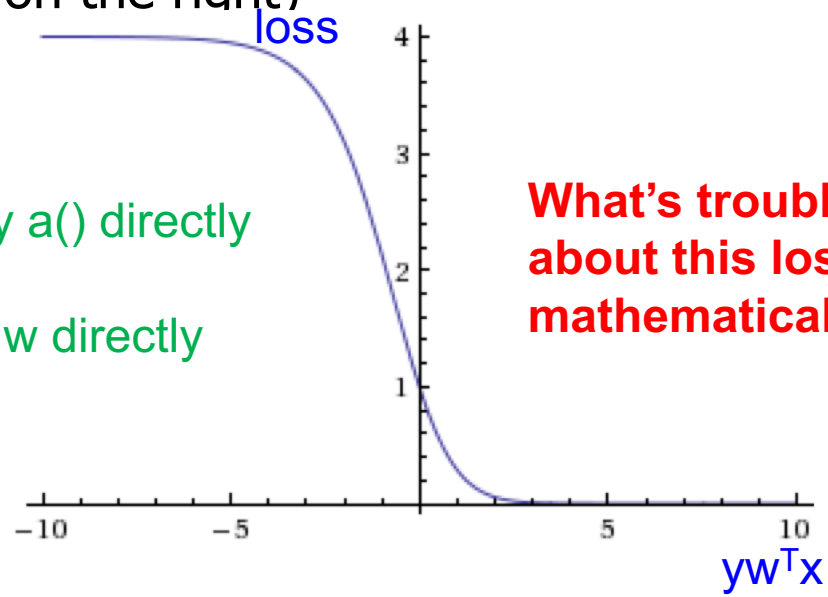
$$\ell(h, \mathbf{z}) = \left(y - a\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2 = \left(1 - ya\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2$$

- If we modify **w** for fixed sample (x,y), how is loss affected?
  - Loss should be expressed in function of w (or $yw^T x$, which is proportional to w for fixed training sample, and makes "correct" always on the right)



We can't modify a() directly

We can modify w directly



What's troubling about this loss, mathematically?

Differentiable perceptron

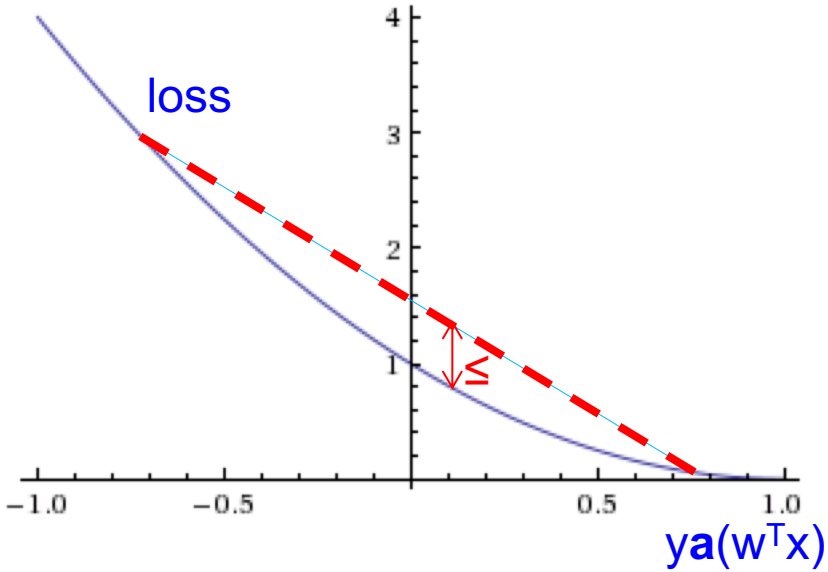
- $h(x)=a(w^T x)$

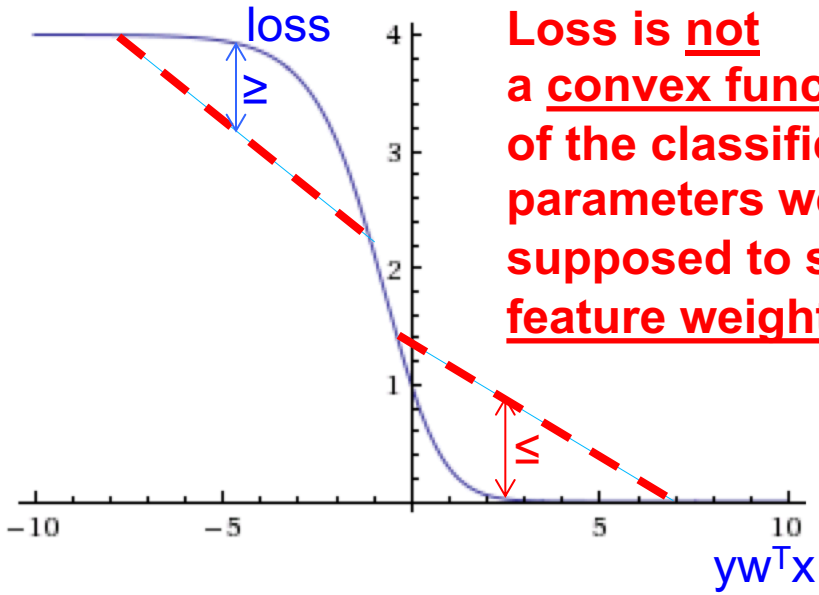
$$a(u) = \frac{2}{1+e^{-u}} - 1$$

- Loss is:

$$\ell(h, \mathbf{z}) = \left(y - a\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2 = \left(1 - ya\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2$$

- If we modify **w** for fixed sample (x,y), how is loss affected?
  - Loss should be expressed in function of w
    (or $yw^T x$ since y,x are constants coming from the dataset)

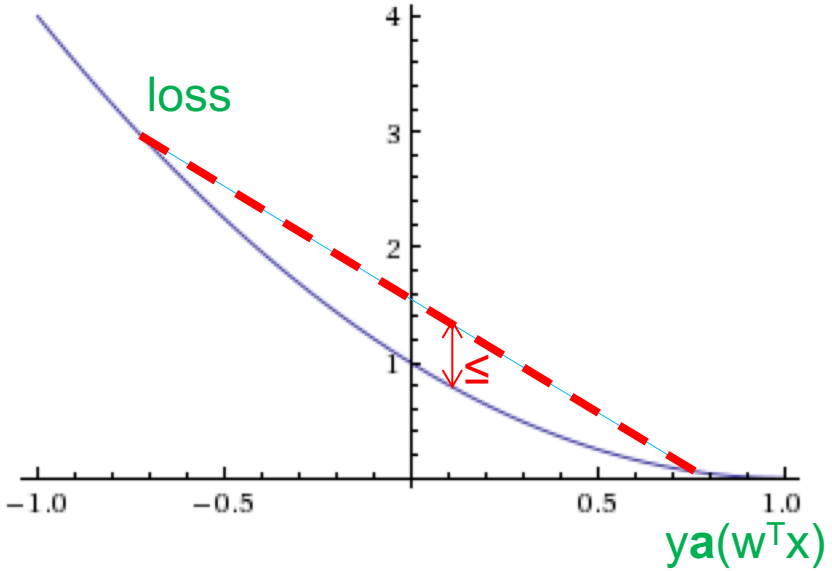


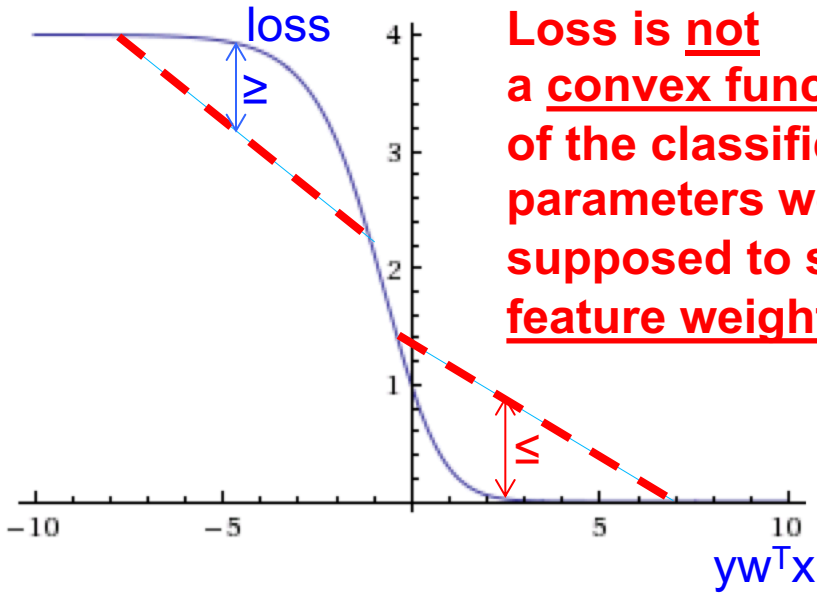


Loss is not a convex function of the classifier parameters we are supposed to set: feature weights *w*

Differentiable perceptron

- Looking at loss related to parameters w
  as *a function of* y**a**($w^T x$) vs. as *a function of* y$w^T x$
  - Let's say loss is: *loss*(g(w))
    - if g(w) is a linear function of w (e.g. g(w)=y$w^T x$)
      then if *loss*(g) is a convex function of g,
      also *loss*(g(w)) is a convex function of w
    - if g(w) is not a linear function of w (e.g. g(w)=y**a**($w^T x$) where a() is a sigmoid)
      then even if *loss*(g) is a convex function of g,
      loss(g(w)) may not be a convex function of w



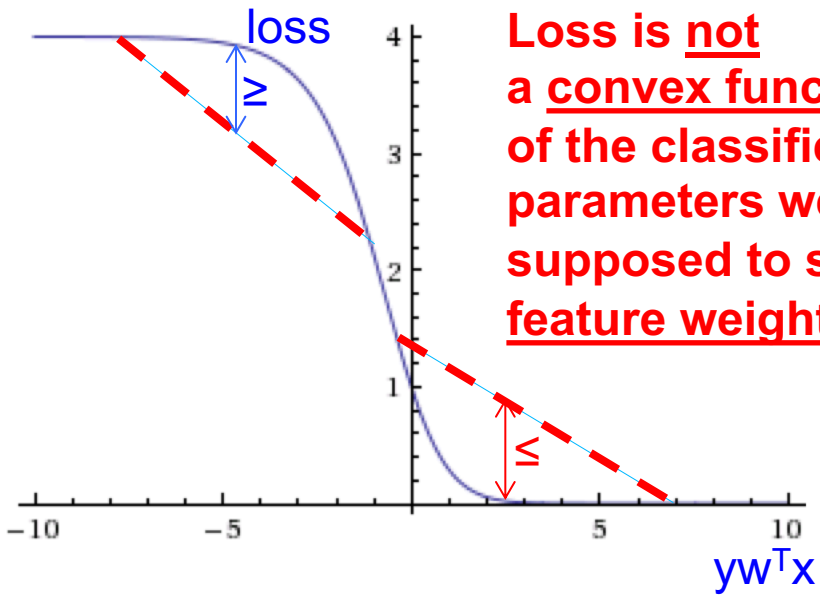


Loss is not a convex function of the classifier parameters we are supposed to set: feature weights *w*

Convexity

- Even if the loss is not convex,
  it still has one minimum (high positive yw^Tx)

- So where is the problem?



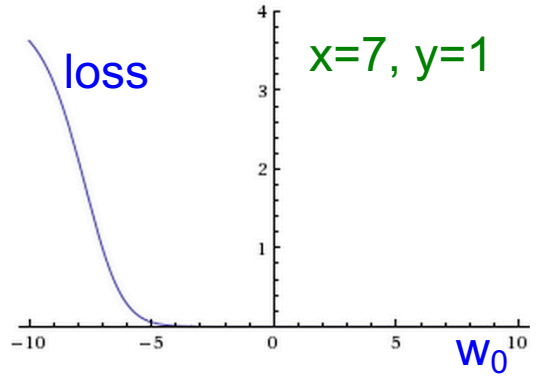
Loss is not a convex function of the classifier parameters we are supposed to set: feature weights *w*

Example

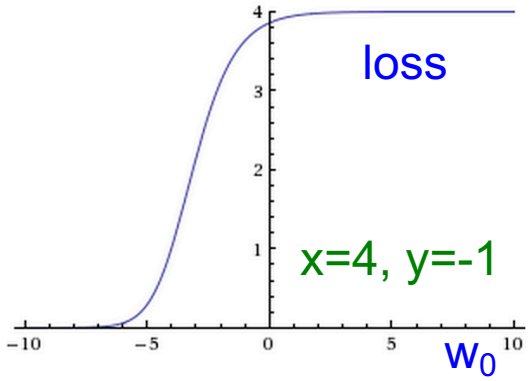
$$\ell(h, \mathbf{z}) = \left(y - a\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2 \quad a(u) = \frac{2}{1+e^{-u}} - 1$$

- Let's fix w=1, try changing w_0 and observe loss

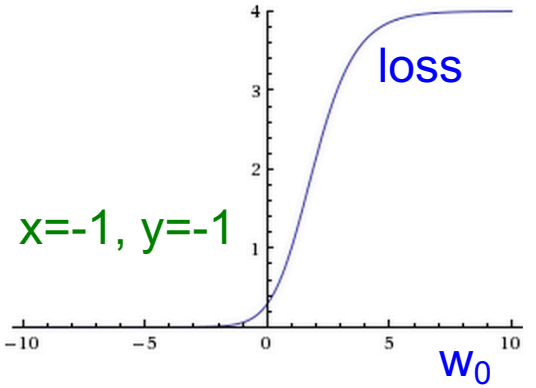
$$\left(1 - \left(\frac{2}{1+\exp(-7-w_0)} - 1\right)\right)^2$$



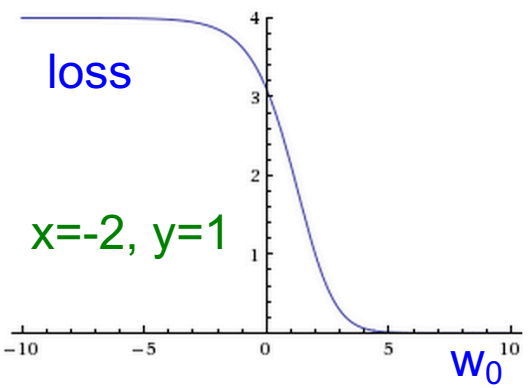
$$\left(-1 - \left(\frac{2}{1+\exp(-4-w_0)} - 1\right)\right)^2$$



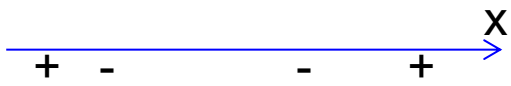
$$\left(-1 - \left(\frac{2}{1+\exp(1-w_0)} - 1\right)\right)^2$$



$$\left(1 - \left(\frac{2}{1+\exp(2-w_0)} - 1\right)\right)^2$$



- h(x)=a(wx+w_0)
  - w=1: h(x)=a(x+w_0)
- We have four samples:
  - x=7, y=1
  - x=4, y=-1
  - x=-1, y=-1
  - x=-2, y=1



- What is the plot of *empirical risk* for those four points, as we alter w_0?

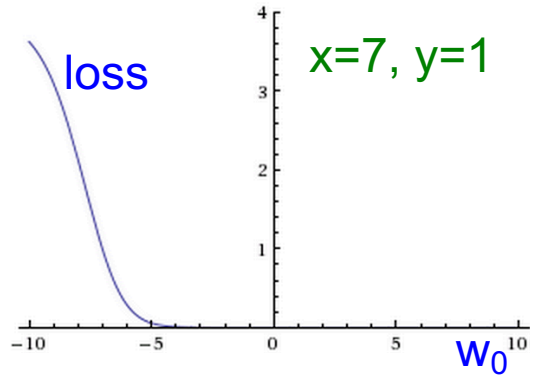
$$\widehat{R}_{S_m}(h) = \frac{1}{m}\sum_{i=1}^{m} \ell(h, \mathbf{z}_i)$$

Example

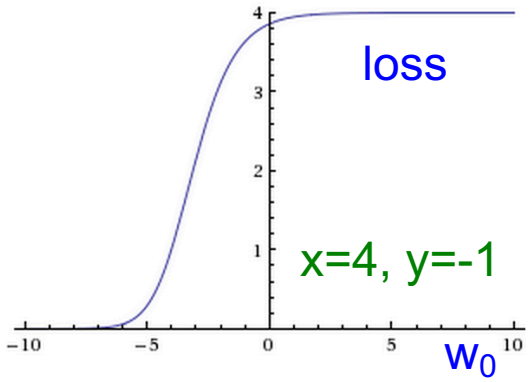
$$\ell(h, \mathbf{z}) = \left(y - a\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2 \quad a(u) = \frac{2}{1+e^{-u}} - 1$$

- Let's fix w=1, try changing w_0 and observe loss

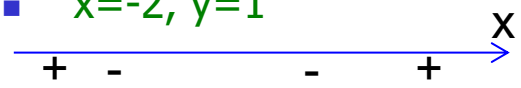
$$\left(1 - \left(\frac{2}{1+\exp(-7-w_0)} - 1\right)\right)^2$$



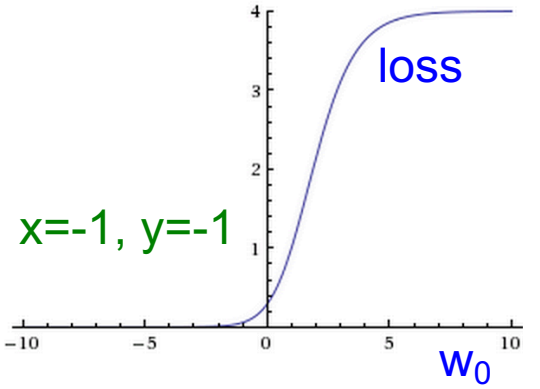
$$\left(-1 - \left(\frac{2}{1+\exp(-4-w_0)} - 1\right)\right)^2$$



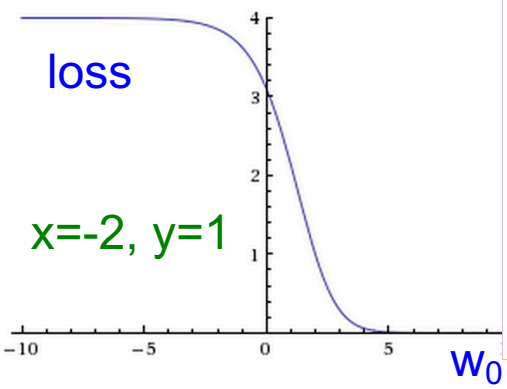
- $h(x)=a(wx+w_0)$
  - w=1: $h(x)=a(x+w_0)$
- We have four samples:
  - x=7, y=1
  - x=4, y=-1
  - x=-1, y=-1
  - x=-2, y=1

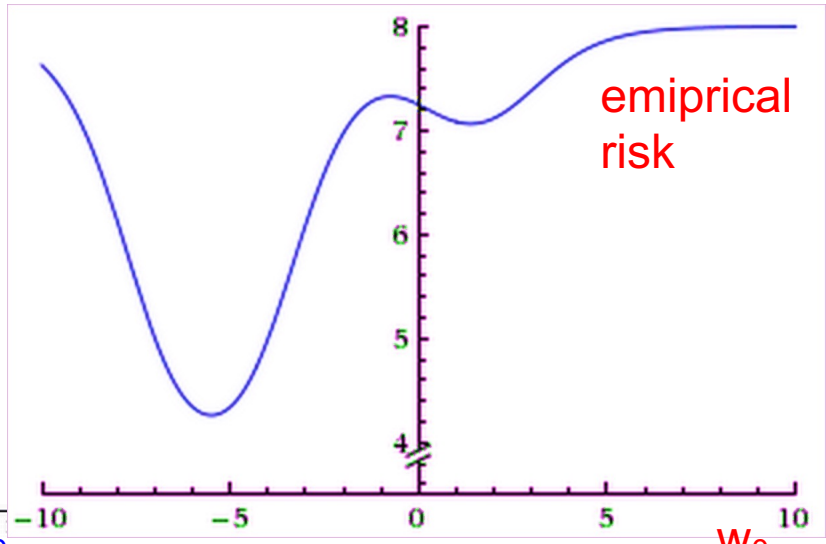


$$\left(-1 - \left(\frac{2}{1+\exp(1-w_0)} - 1\right)\right)^2$$



$$\left(1 - \left(\frac{2}{1+\exp(2-w_0)} - 1\right)\right)^2$$

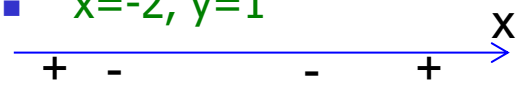


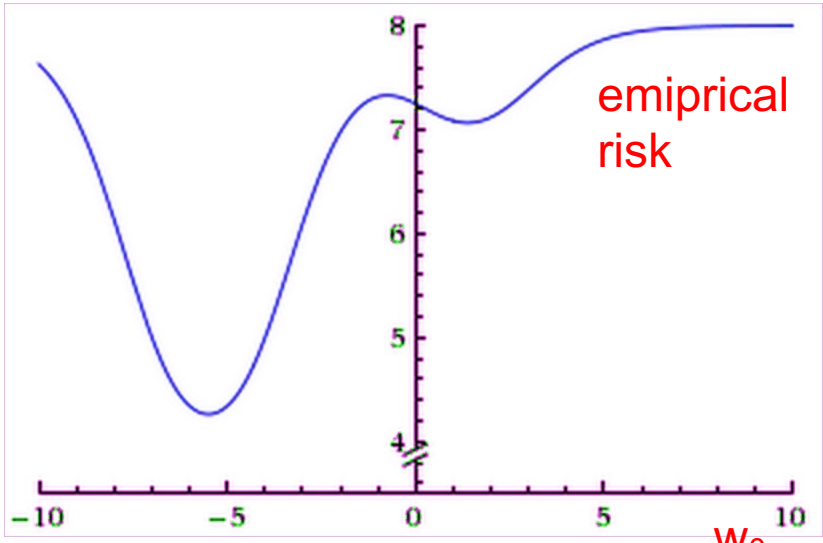


Example

$$\ell(h, \mathbf{z}) = \left(y - a\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2 \quad a(u) = \frac{2}{1+e^{-u}} - 1$$

- Let's fix w=1, try changing w_0 and observe loss
- Predictions for 4 samples:
  - If we start with w_0=0 and apply gradient descent we'll land at w_0=2
  - Solution w_0=2:
    - 7+2=9 ok
    - 4+2=6 wrong
    - -1+2=1 wrong
    - -2+2=0 wrong-ish
  - Solution w_0=-5:
    - 7-5=2 ok
    - 4-5=-1 ok
    - -1-5=-6 ok
    - -2-5=-7 wrong
- $h(x)=a(wx+w_0)$
  - w=1: $h(x)=a(x+w_0)$
- We have four samples:
  - x=7, y=1
  - x=4, y=-1
  - x=-1, y=-1
  - x=-2, y=1

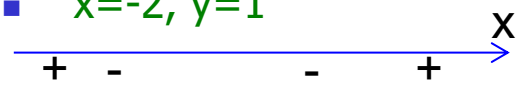


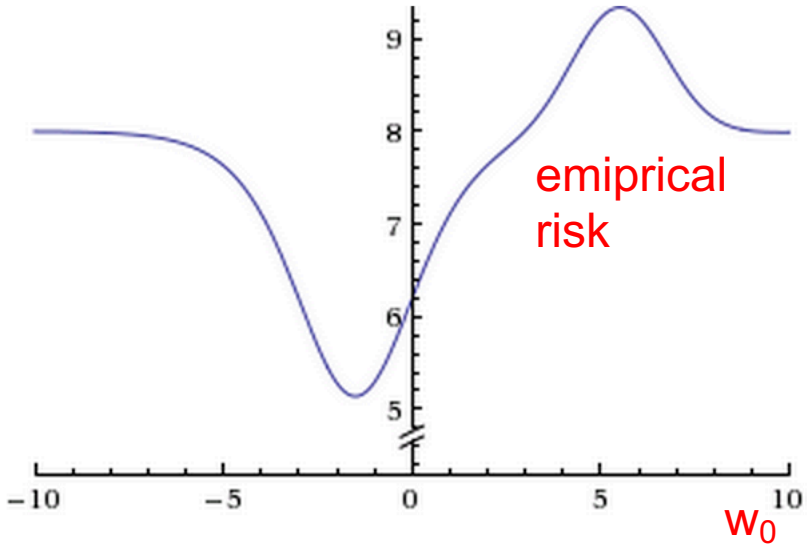


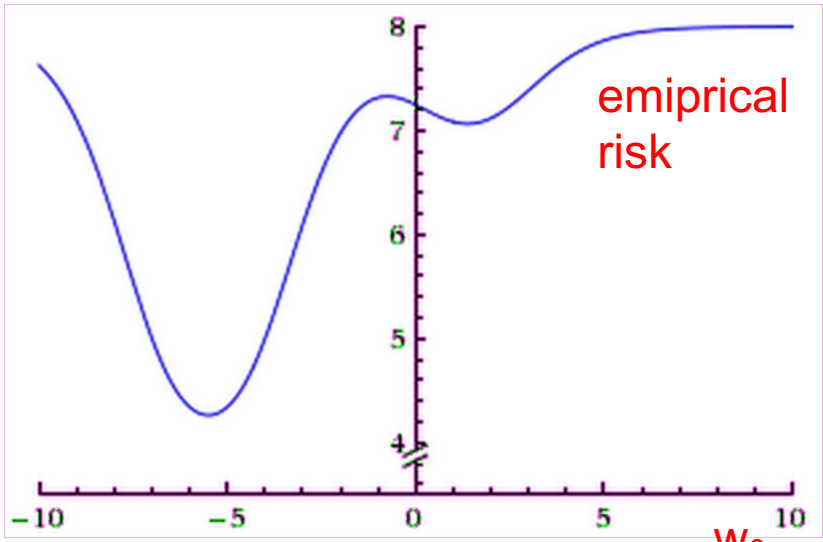
Example

$$\ell(h, \mathbf{z}) = \left(y - a\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2 \quad a(u) = \frac{2}{1+e^{-u}} - 1$$

- Let's fix w, try changing w_0 and observe loss
- What if we fix w=-1?
  - w_0=-1.5 is best
  - again, one wrong prediction
    - x=-2 predicted as 1
    - x=-1, 4, 7 predicted as -1
- h(x)=a(wx+w_0)
  - w=1: h(x)=a(x+w_0)
- We have four samples:
  - x=7, y=1
  - x=4, y=-1
  - x=-1, y=-1
  - x=-2, y=1



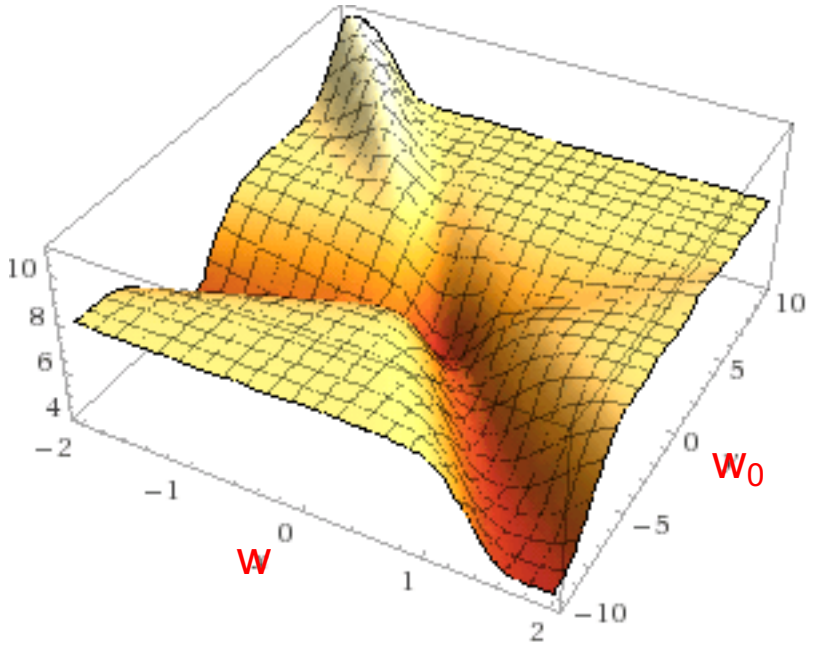




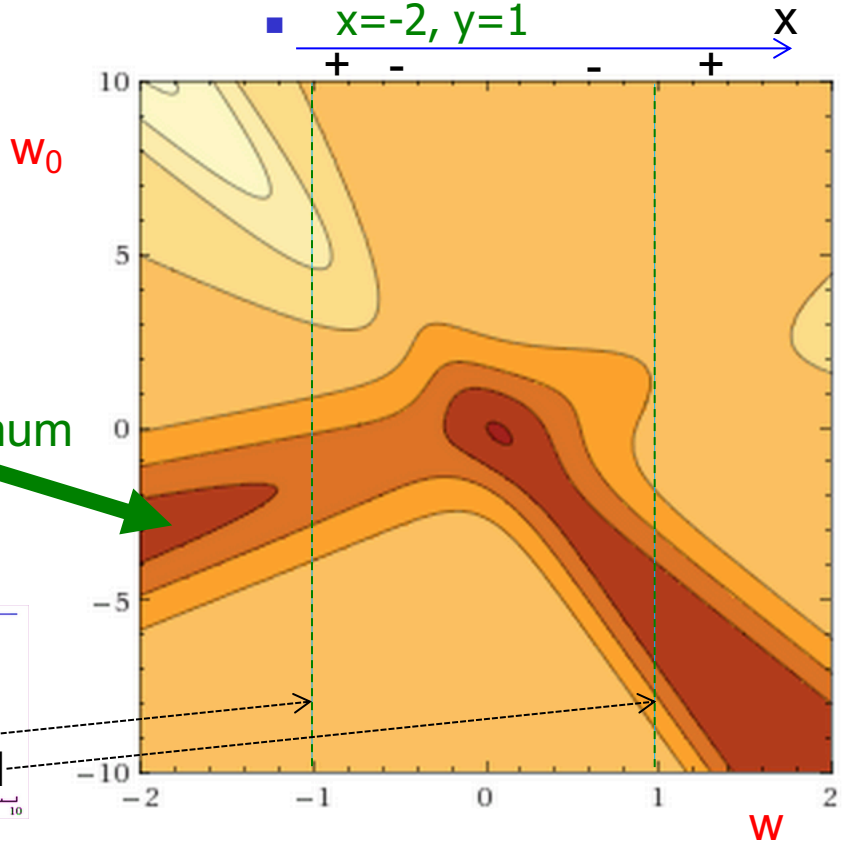
Example

$$\ell(h, \mathbf{z}) = \left(y - a\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2 \quad a(u) = \frac{2}{1+e^{-u}} - 1$$

- Empirical risk for any w, w_0

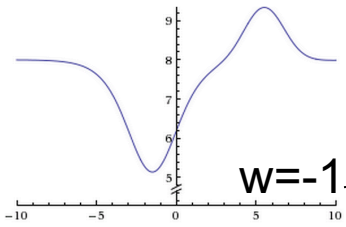


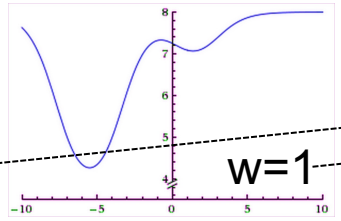
- $h(x)=a(wx+w_0)$
- We have four samples:
  - x=7, y=1
  - x=4, y=-1
  - x=-1, y=-1
  - x=-2, y=1



Where is global minimum?

Local minimum



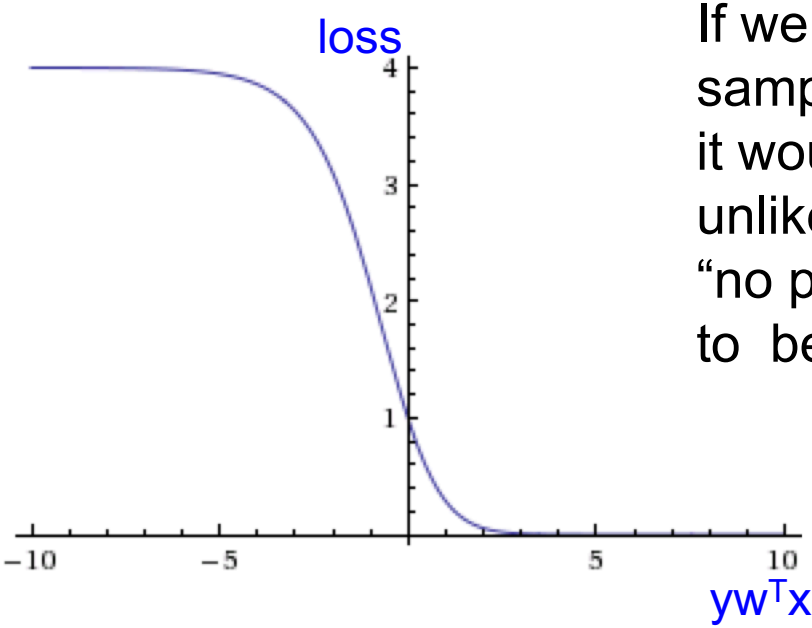


Example

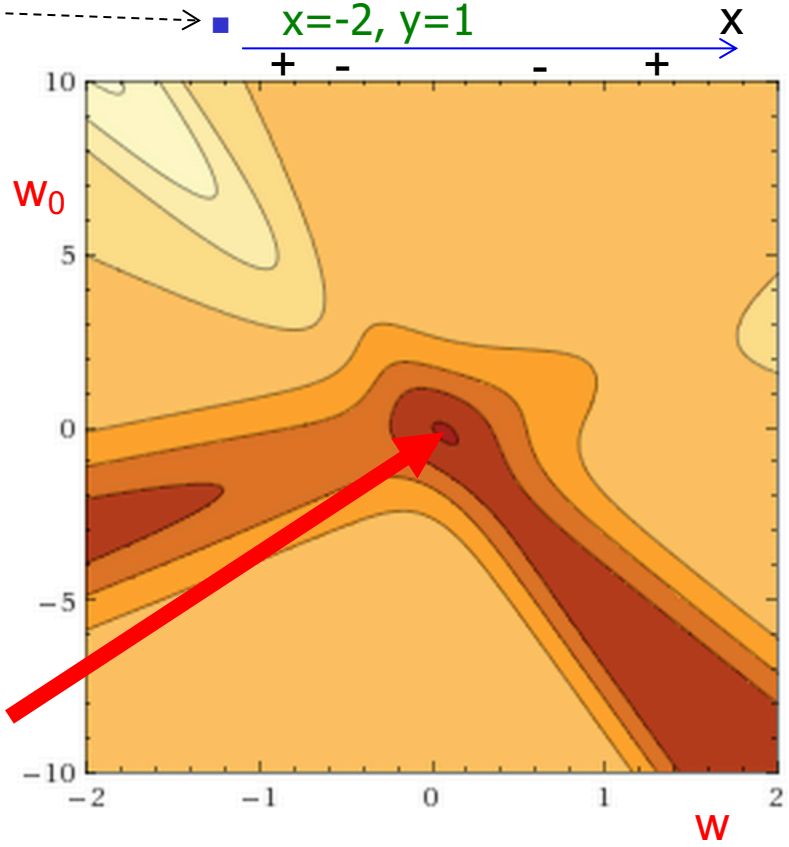
$$\ell(h,\mathbf{z}) = \left(y - a\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2 \quad a(u) = \frac{2}{1+e^{-u}} - 1$$

- "No prediction" still the best choice
- Solution (w=1, w_0=-5.5):
  - 2 correct-ish prediction => loss≈0.1
  - 1 correct predictions => loss=0
  - 1 error prediction => loss≈4
- Solution (w=0, w_0=0):
  - 4 x "no prediction" => risk=4

- h(x)=a(wx+w_0)
- We have four samples:
  - x=7, y=1
  - x=4, y=-1
  - x=-1, y=-1
  - x=-2, y=1



If we had more samples, it would be unlikely for "no prediction" to be optimum



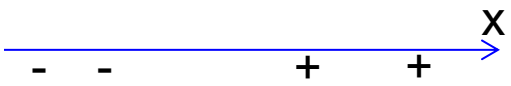
Example

$$\ell(h, \mathbf{z}) = \left(y - a\left(\mathbf{w}^{\dagger T}\mathbf{x}^{\dagger}\right)\right)^2 \quad a(u) = \frac{2}{1+e^{-u}} - 1$$

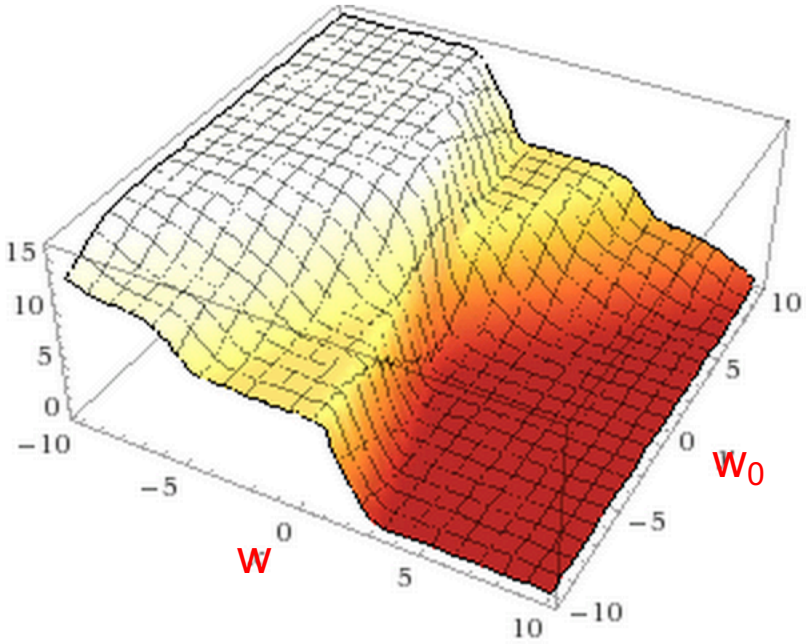
- Different, easier set of samples

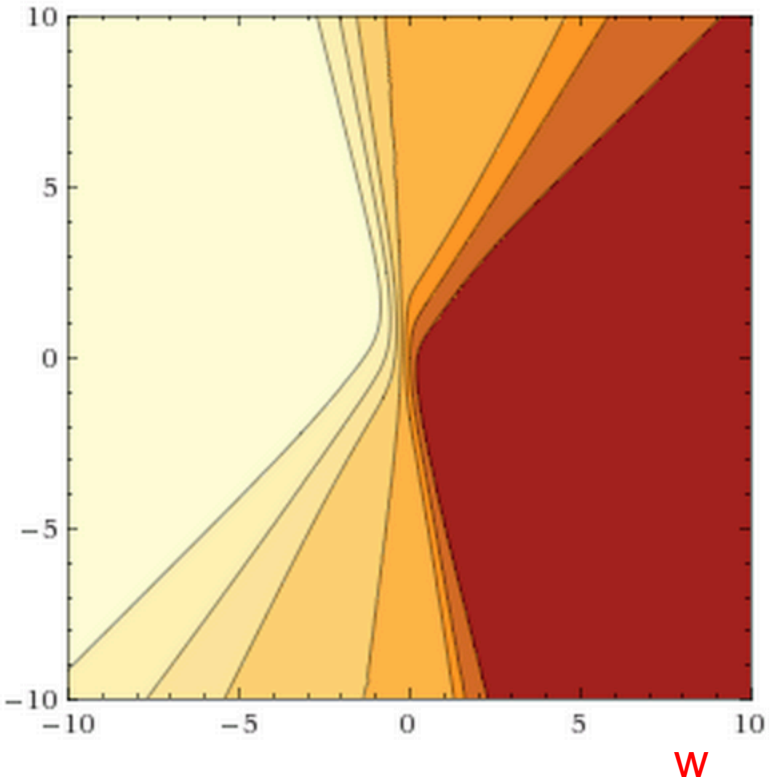
  $h(x)=a(wx+w_0)$

  - x=7, y=1
  - x=4, y=1
  - x=-1, y=-1
  - x=-2, y=-1



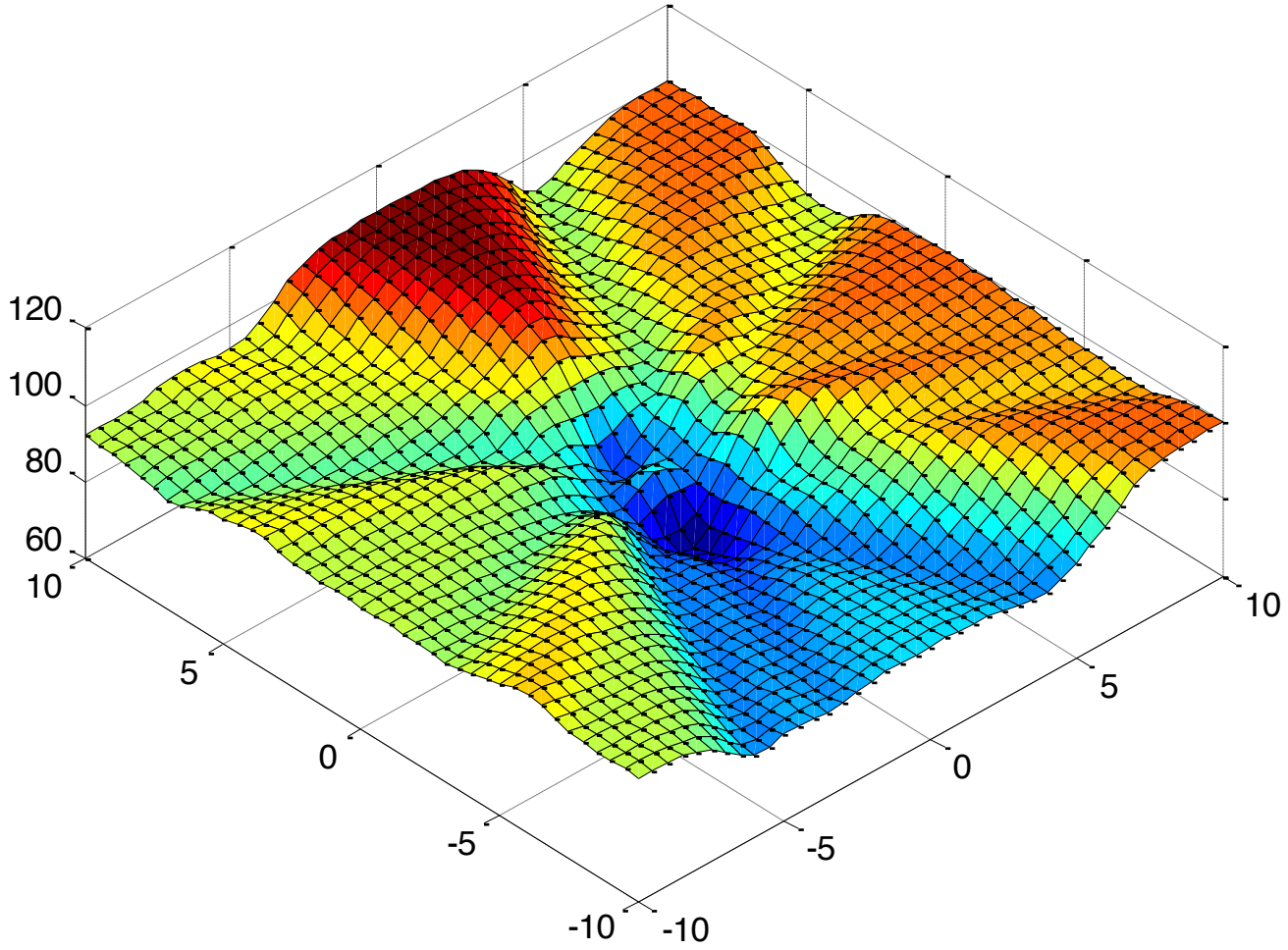
- Empirical risk for any w, w_0





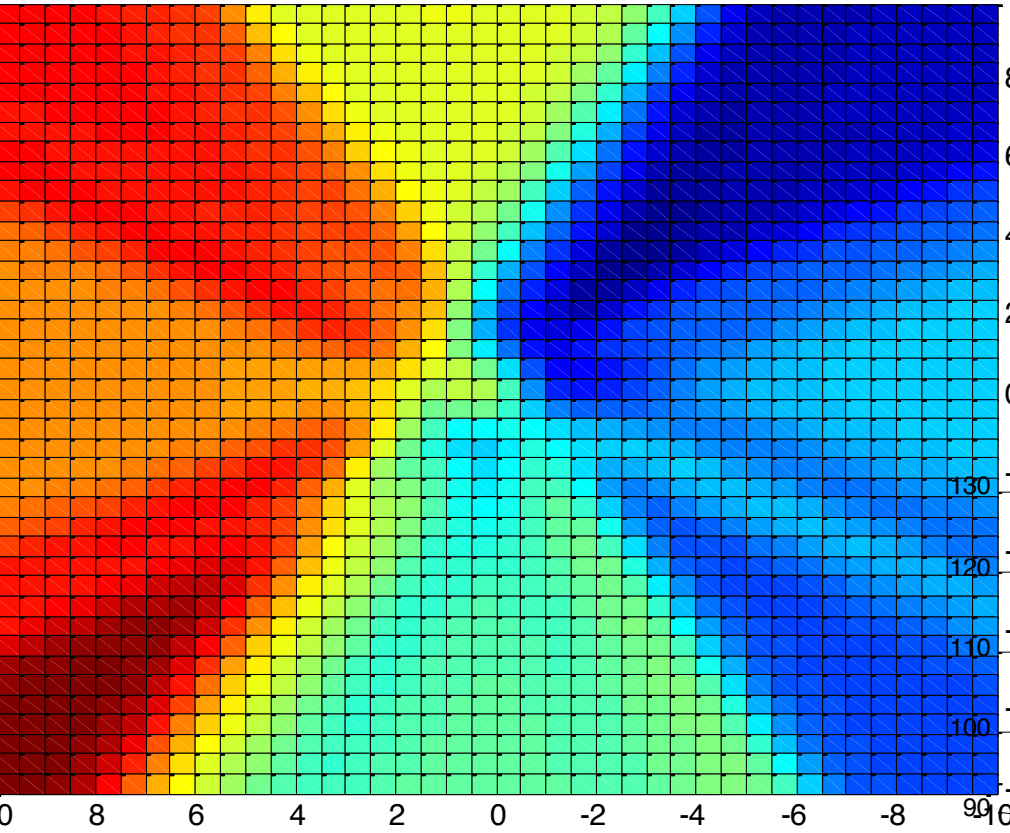
Differentiable perceptron

- Some other examples: empirical risk surface

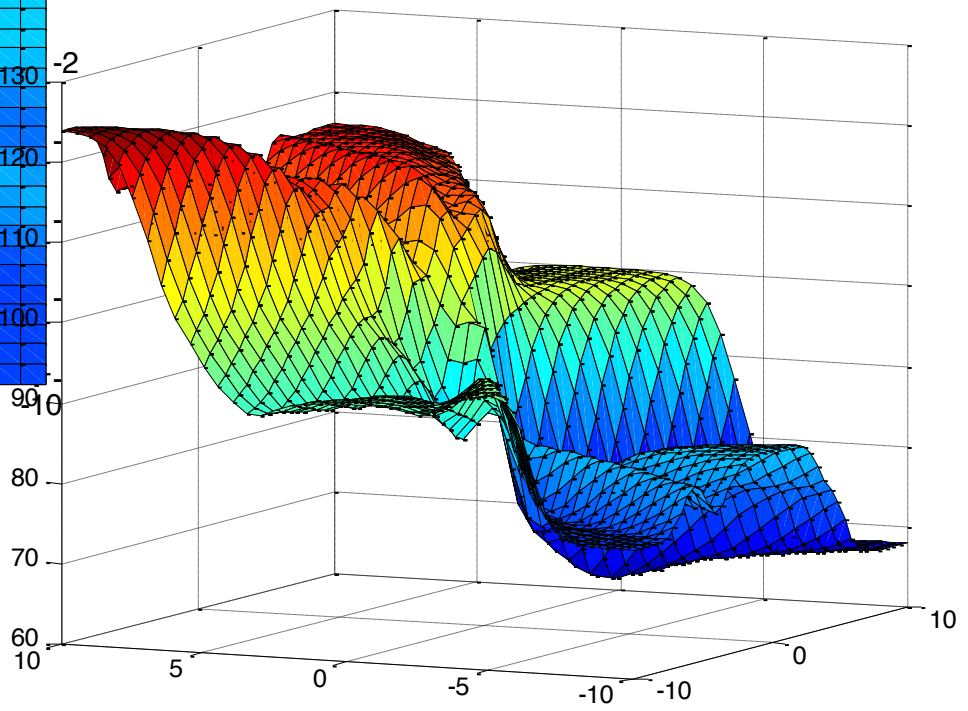


Differentiable perceptron

- Risk surface for quadratic loss over sigmoid

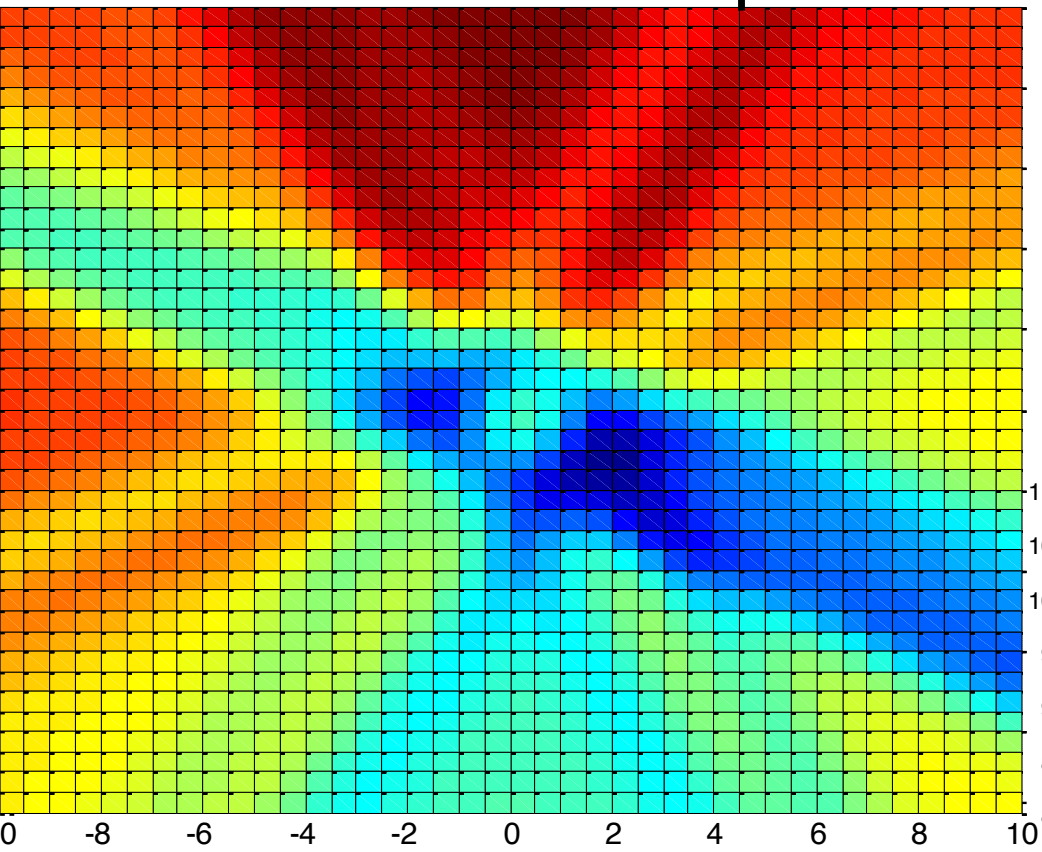


- Local minima!

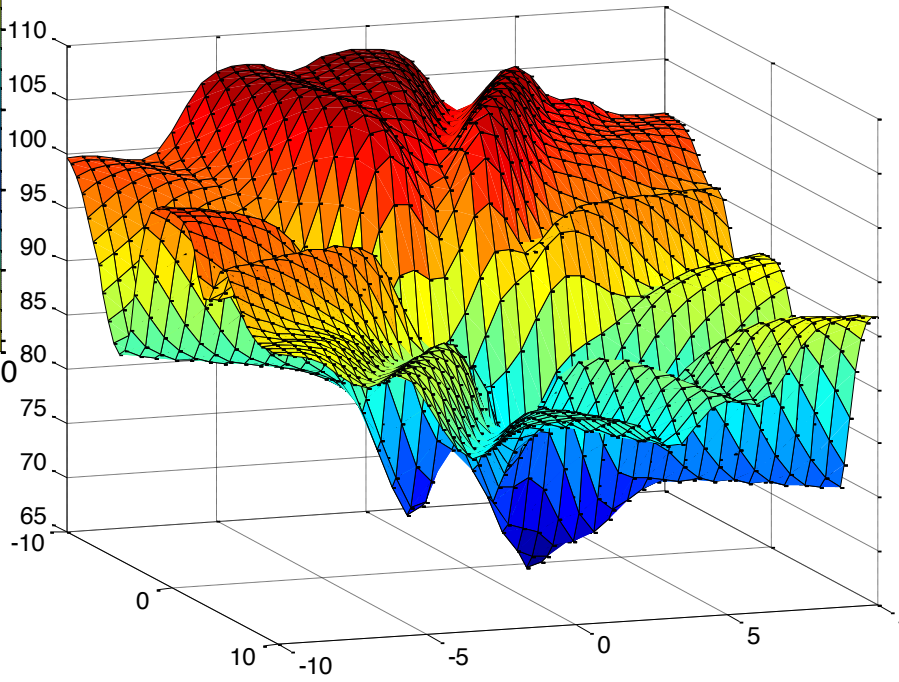


Differentiable perceptron

- Risk surface for quadratic loss over sigmoid

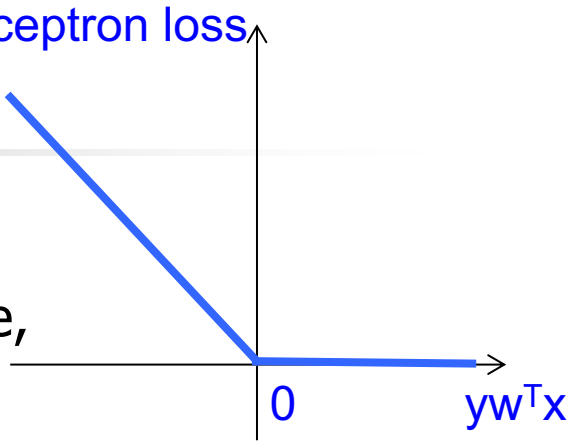


- We may get trapped in a shallow valley: and return a vector **w** that's far from the best one!

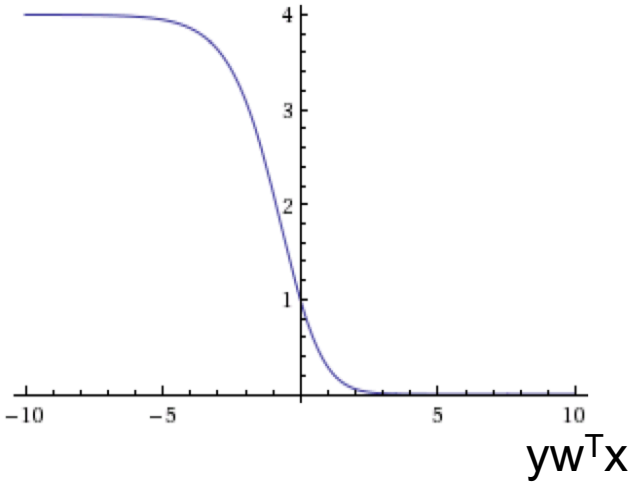


Recap

- Perceptron loss
  - convex, non-increasing, but non-differentiable, and w=0 is always a global minimum
- Quadratic loss on sigmoid activation
  - differentiable, non-increasing but non-convex => local minima
- LDA: quadratic loss directly on yw^Tx
  - convex, differentiable, but non-monotonic (increasing for large yw^Tx)

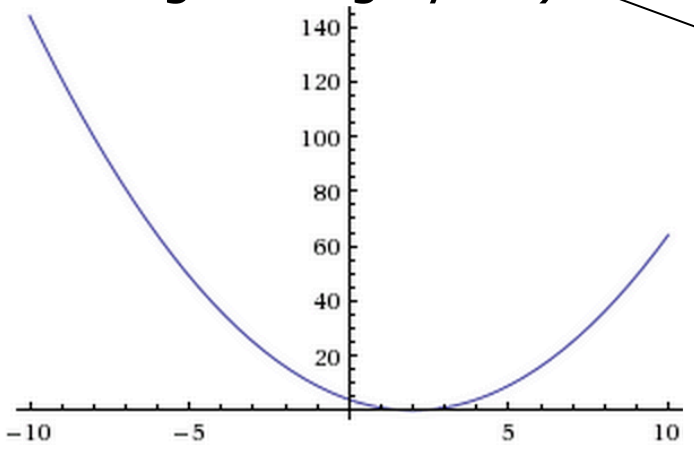


delta rule loss



This wasn't a problem for quadratic over sigmoid

Square loss

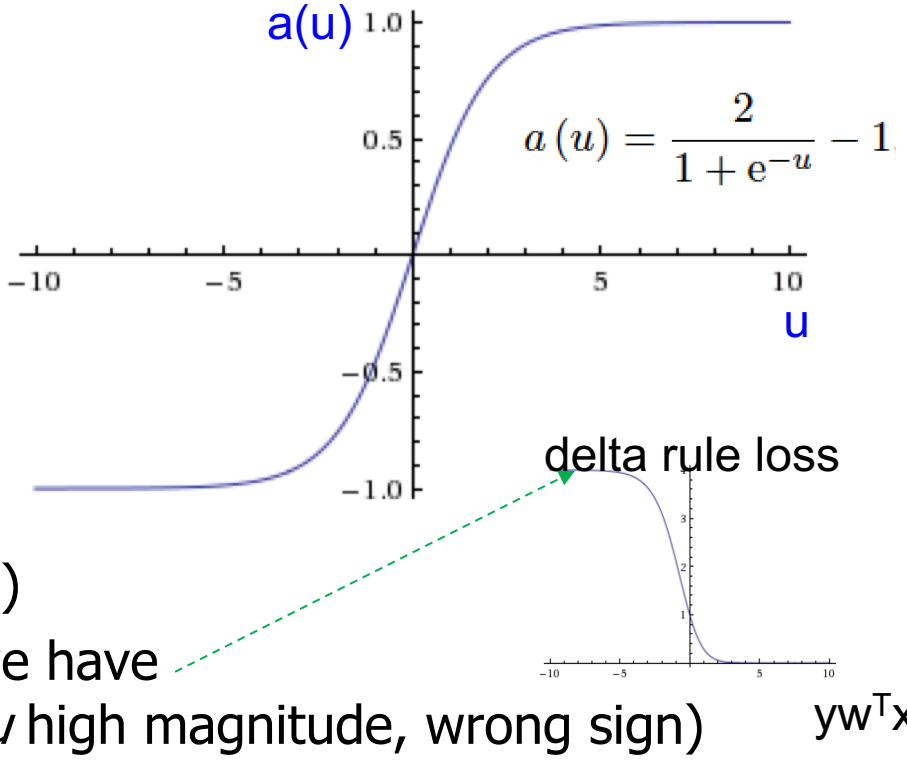


Differentiable perceptron h(x)=a($w^T x$)

- Another problem with delta rule:

$$\mathbf{w}^{\dagger}{}_{t+1} = \mathbf{w}^{\dagger}{}_{t} - \frac{c}{2} \left. \frac{\partial \ell\left(h, \mathbf{z}\right)}{\partial \mathbf{w}^{\dagger}} \right|_{\mathbf{w}^{\dagger}=\mathbf{w}^{\dagger}{}_{t}} = \mathbf{w}^{\dagger}{}_{t} + c\left[y - a\left(\mathbf{w}^{\dagger}{}_{t}^{T} \mathbf{x}^{\dagger}\right)\right] a'\left(\mathbf{w}^{\dagger}{}_{t}^{T} \mathbf{x}^{\dagger}\right) \mathbf{x}^{\dagger}$$

- The update depends on:
  - (y-a(u))
    - Smaller as we approach correct prediction
  - a'(u)
    - Much smaller as we approach correct prediction (*u* v.large)
    - And also very small when we have really incorrect prediction (*u* high magnitude, wrong sign)



- Very slow learning for large |u|

For next time

Please review:

Expected value

$$\mathrm{E}[X] = \sum_{i=1}^{k} x_i\, p_i = x_1 p_1 + x_2 p_2 + \cdots + x_k p_k.$$