Introduction to Machine Learning

Lecture 11

Instructor:
Dr. Tom Arodz

Recap

- Modern machine learning:

```
minimizeθ: error_metric(
   S, some_f(θ)(x in S).to_class_probs())
```

- Filling the gaps:
  - `some_f(θ)(x)` or `some_f(θ;x)`
    - What family of functions? Simple choice that often works:
      `some_f(θ;x) = linear(w,b;x) = w`^{T}`x+b`
      Linear models - simple and often very effective
  - `to_class_probs()`
    - `some_f(θ;x)` may not return probabilities, how to convert to: ≥0, sum=1
  - `error_metric`
    - Classification error won't work. MSE is not ideal. We need something better!
  - `minimize`$_\theta$
    - **Gradient descent, over parameters** θ **of the model** `some_f(θ;x)`
    - $\theta_{n+1} = \theta_n - c\,\nabla_\theta$ **error_metric(S, model(**θ_n**, S))**

Recap: Losses so far

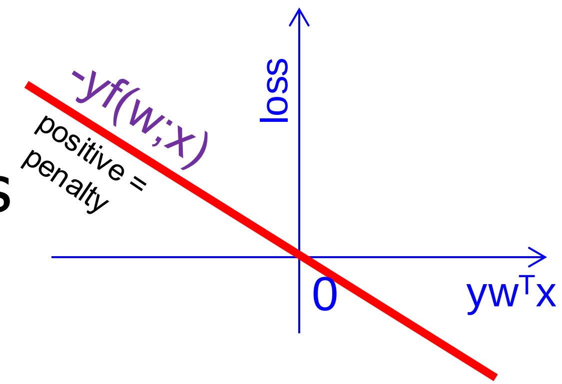
All have some problems:

- 0/1 loss (is_incorrect): flat
- -yf(x): rewards can lead to errors
- MSE for classification: penalty for correct predictions

11.1. Perceptron loss

Loss function

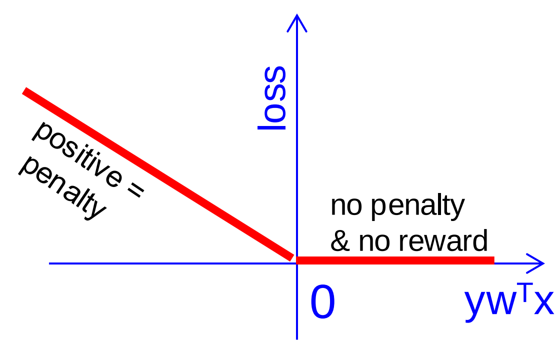
- -yf(w;x) is not a good error metric, it gives "rewards" which leads to problems



- One possible solution:
  remove the reward part!
  - **Perceptron_loss** = `max(0,-y(wᵀx+b))`

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

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

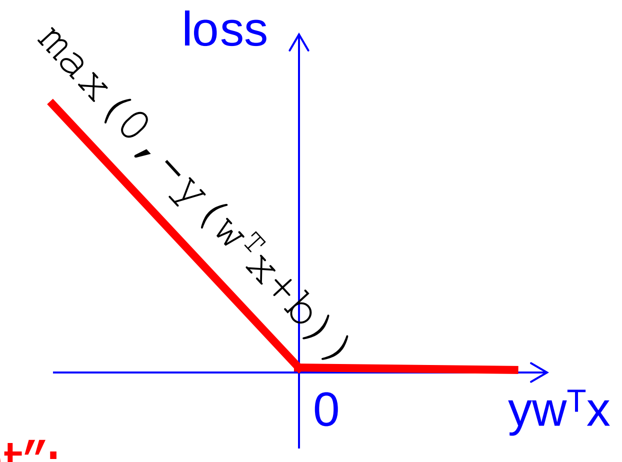


Perceptron loss

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

- **Minimize perceptron loss:**

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



- What is the gradient?
  $\nabla_{\mathbf{w}} = -y\mathbf{x}$ for wrong prediction (loss = $-y\mathbf{w}^T\mathbf{x}$)
  $\nabla_{\mathbf{w}} = 0$ for correct prediction (loss = 0)
- Update rule for a single sample is "negated gradient":

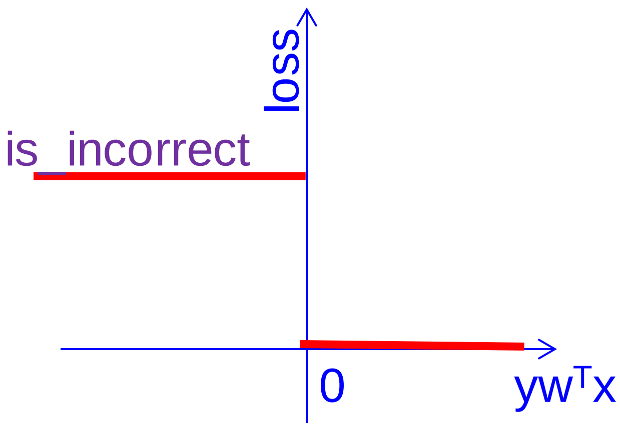
$$\mathbf{w}^{\dagger}_{t+1} = \mathbf{w}^{\dagger}_t - 2c \left. \frac{\partial \ell(h, \mathbf{z})}{\partial \mathbf{w}^{\dagger}} \right|_{\mathbf{w}^{\dagger} = \mathbf{w}^{\dagger}_t}$$

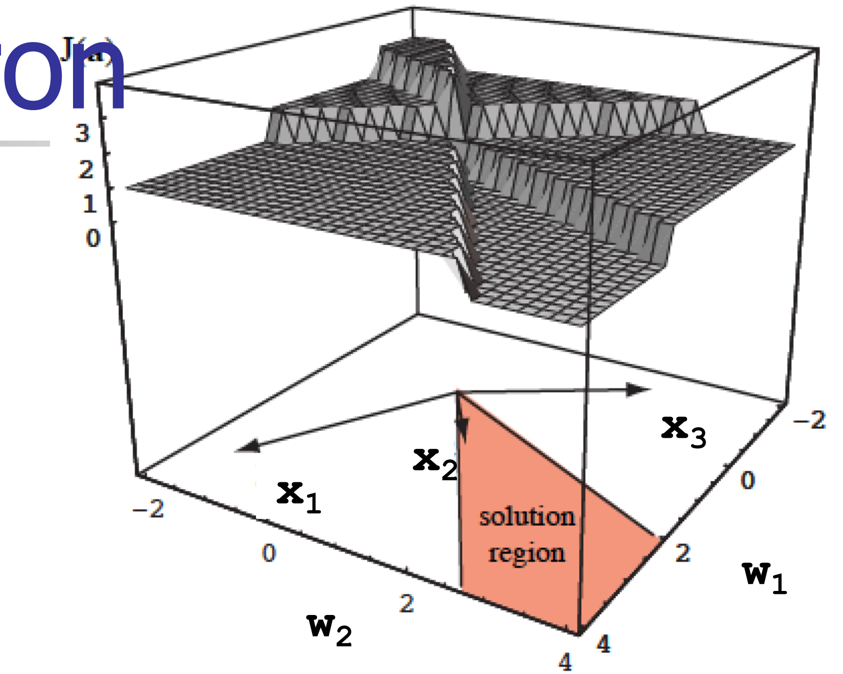
  - **Present a sample x and predict:**
    h(x) = $\mathbf{w}^T\mathbf{x}+b$
  - **Compare true class *y* with predicted class h(x)**
  - **If correct, do nothing ($\nabla_{\mathbf{w}} = 0$)**
  - **If prediction is wrong ($\nabla_{\mathbf{w}} = -y\mathbf{x}$), update weights:**

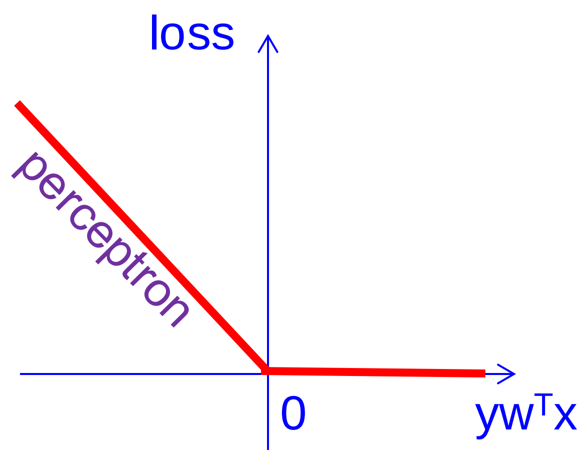
$$\mathbf{w}^{\dagger}_{t+1} = \mathbf{w}^{\dagger}_t + 2cy\mathbf{x}^{\dagger}$$

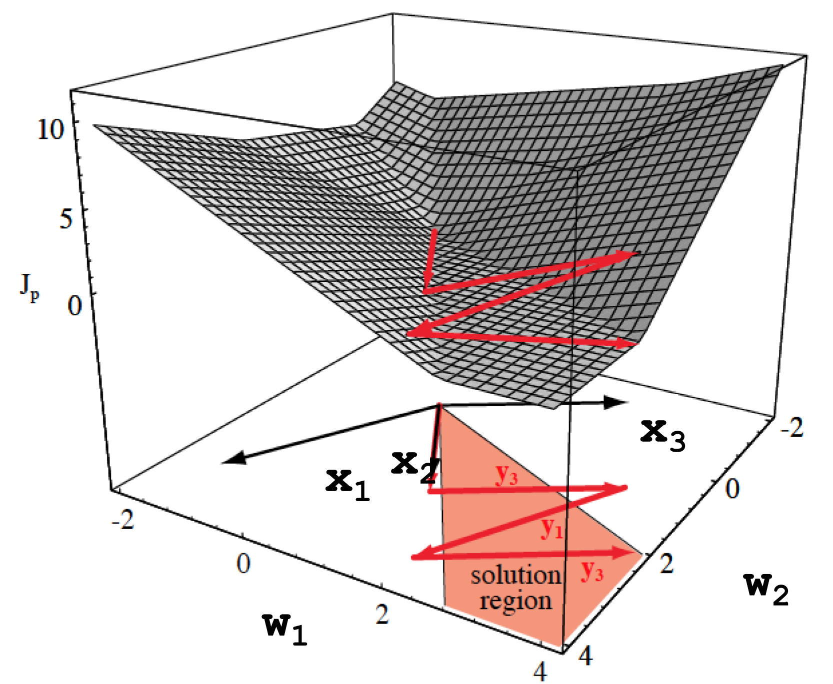
Similar to HW1

0/1 vs perceptron



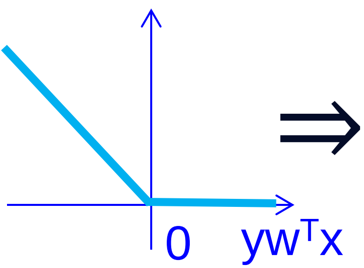




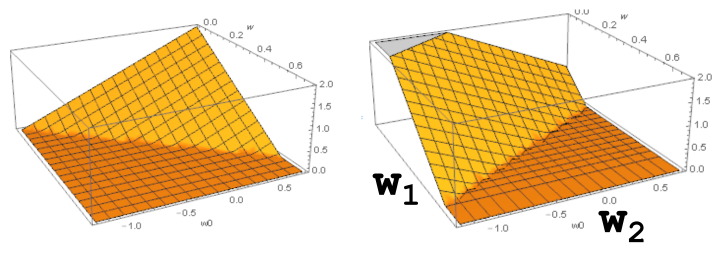


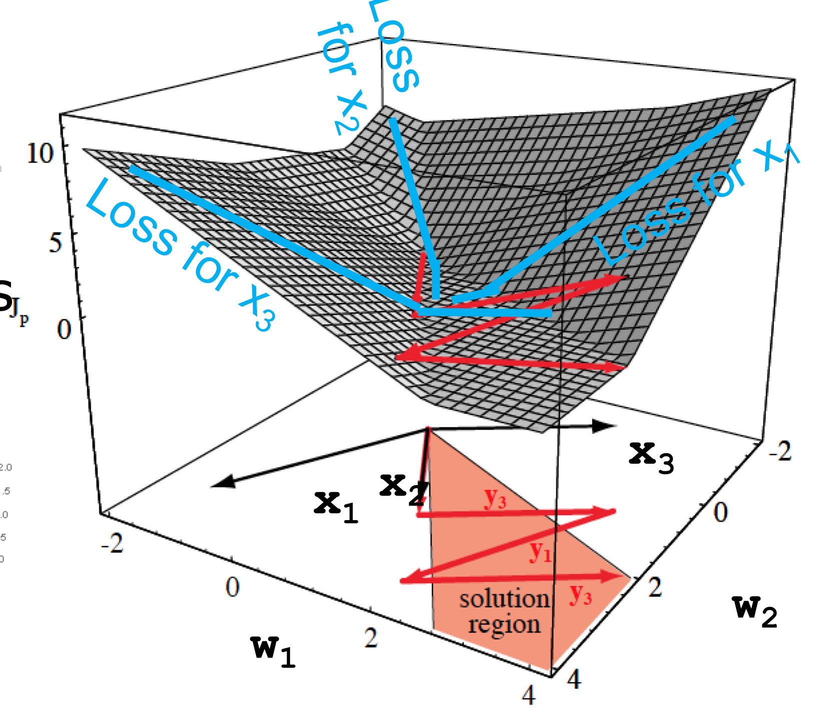
Loss surface

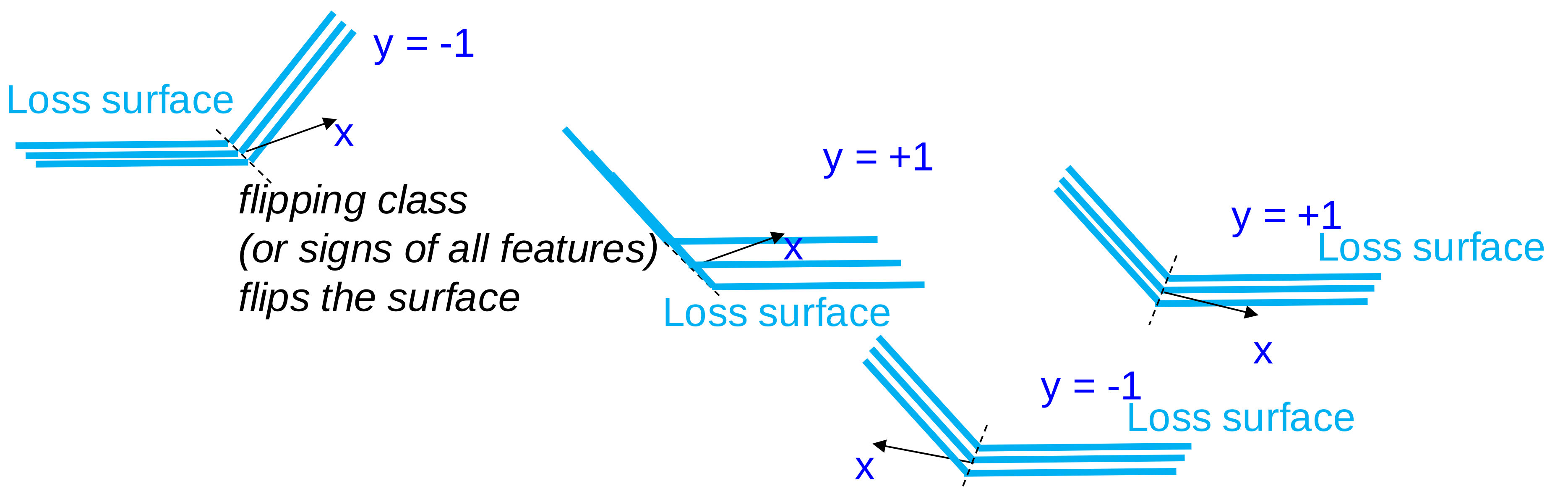
- The “piecewise linear” surface of loss for different values of w is a sum of three loss function surfaces, one per each sample.



$\Rightarrow$



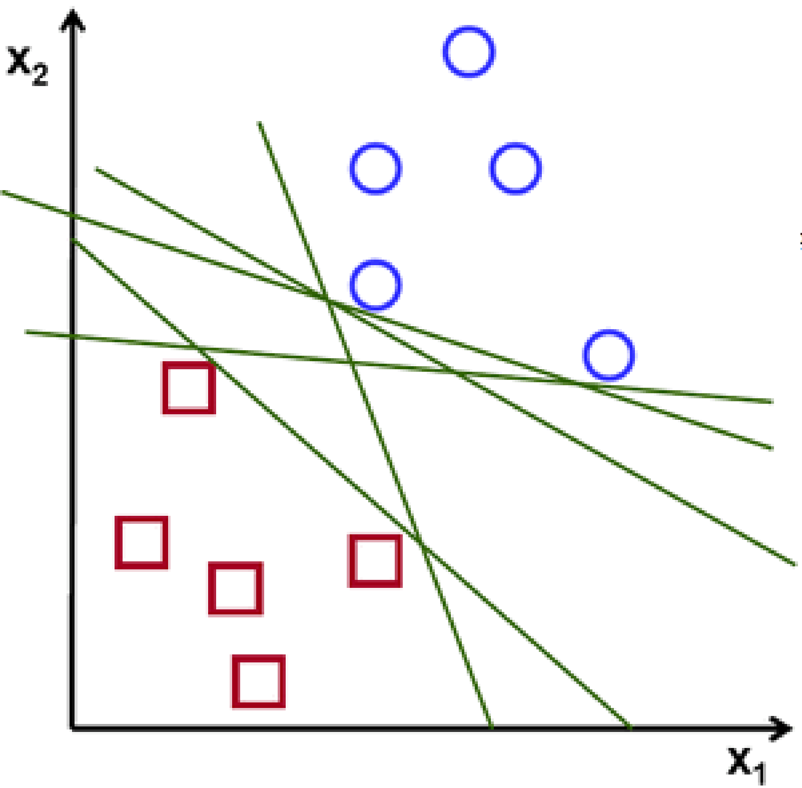


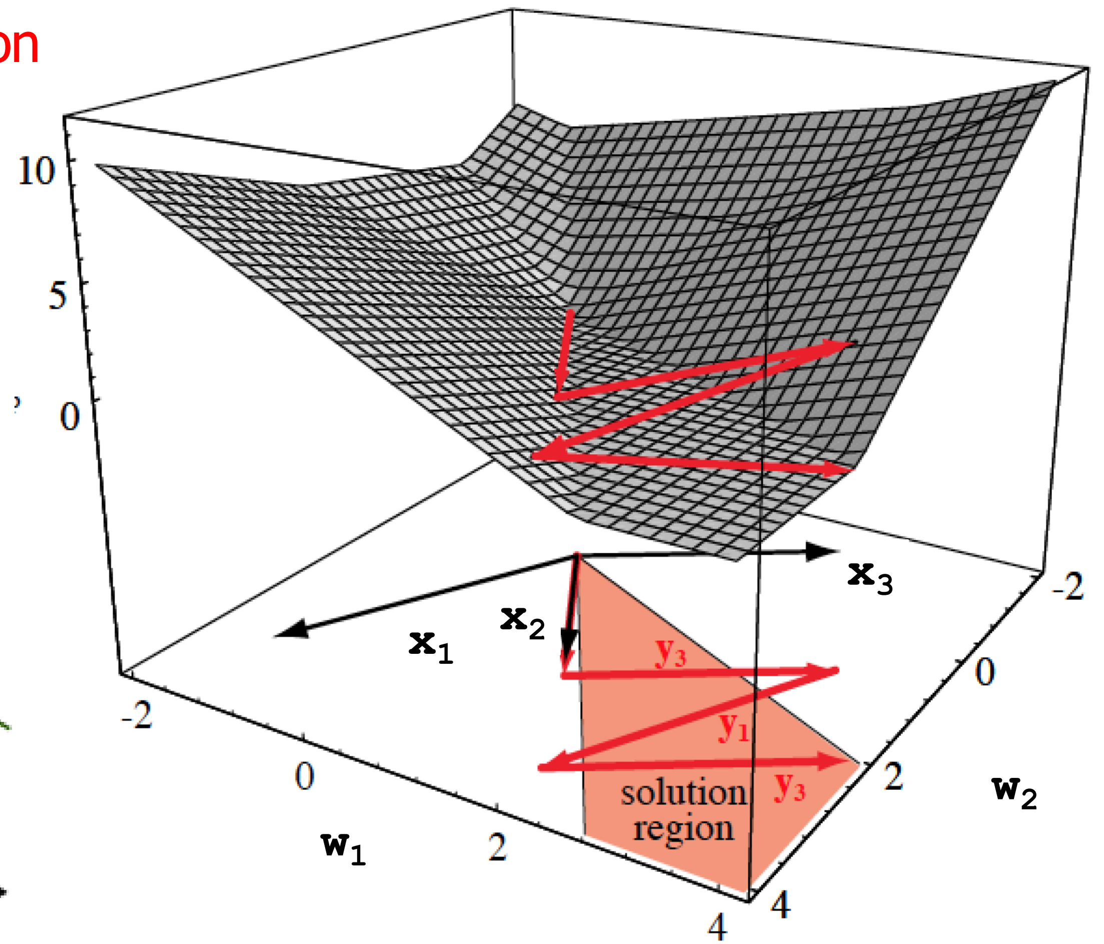


flipping signs in x AND flipping class:
no change to the loss surface (yx the same)

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

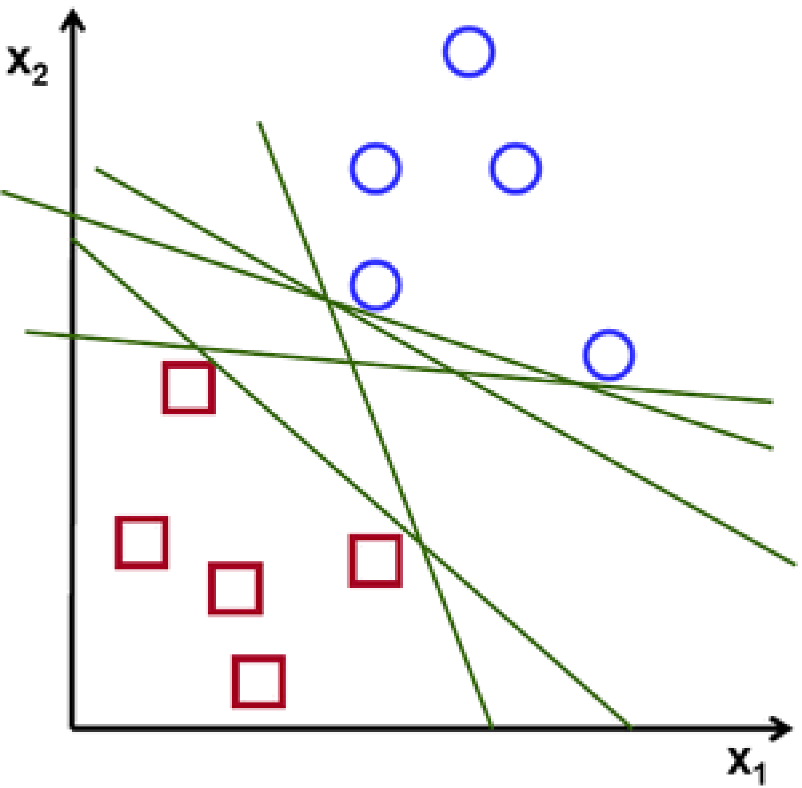




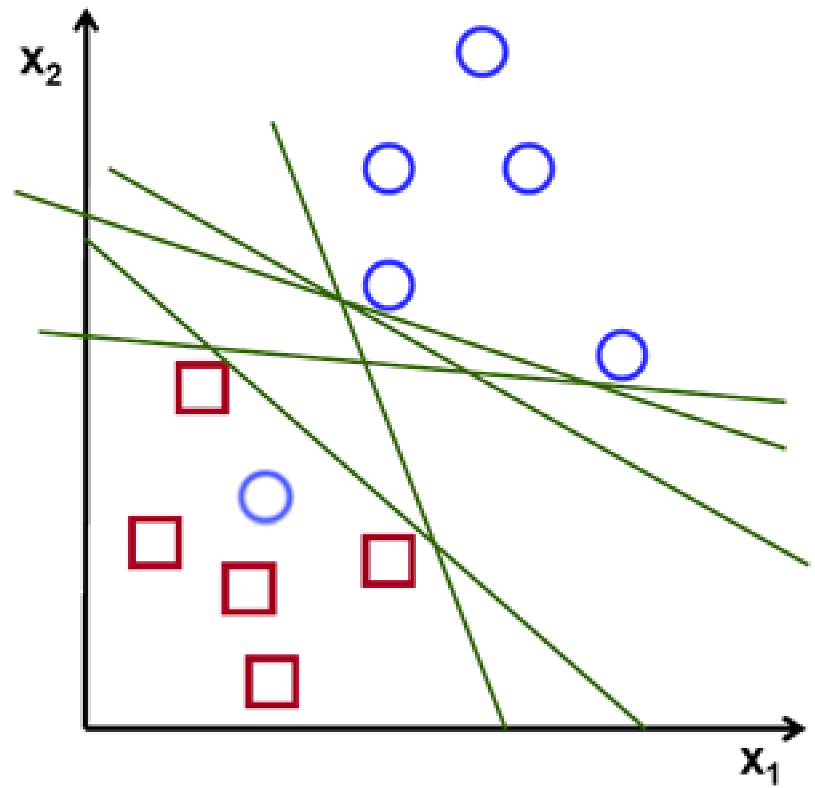
Linearly separable problems

- Some problems/datasets are linearly non-separable

Linearly separable problem



Non-linearly-separable problem
none of the lines works
no other line would work either

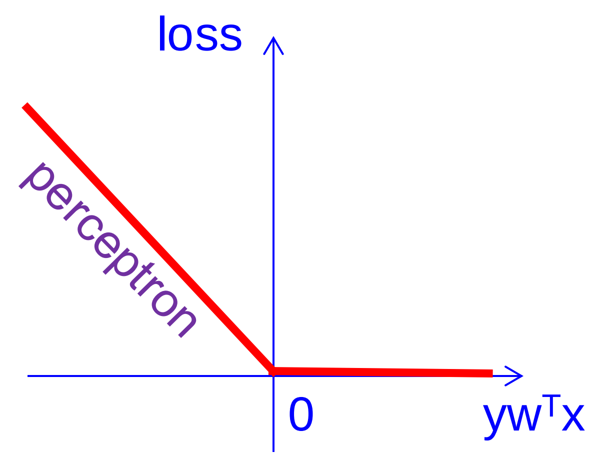


Perceptron

- **Perceptron loss for linear classifier:**

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

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

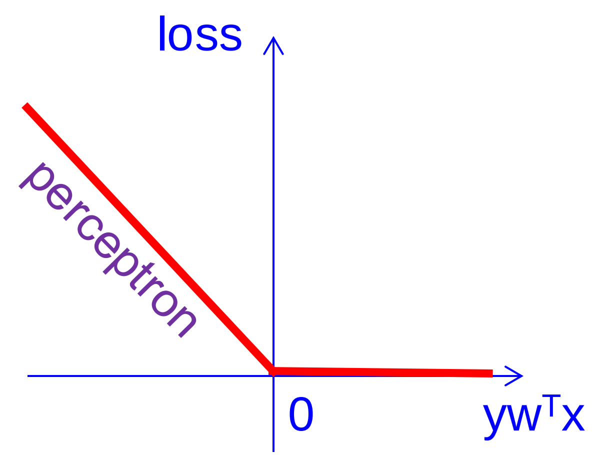


Perceptron

- **Perceptron loss for linear classifier:**

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

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



Losses so far

All have some problems:

- Flat: 0/1 loss
- Has rewards: -yf(x)
- Non-monotonic - high penalty for some correct predictions: MSE
- ”no prediction” is optimal: Perceptron loss

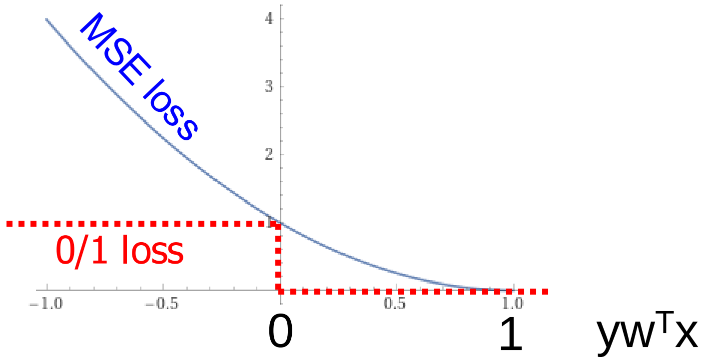
The loss should be:

non-flat (at least for incorrect predictions),
monotonic (decreasing), nonnegative,
loss(w=0)>0

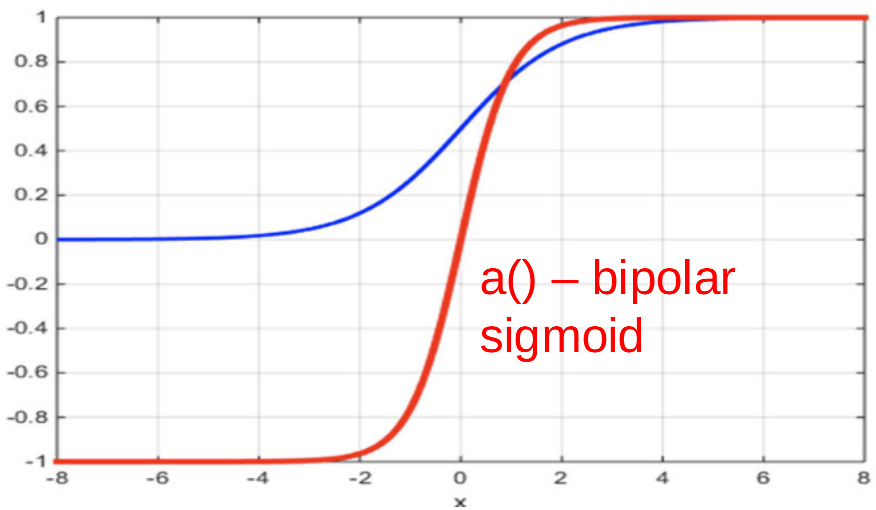
11.2. MSE over sigmoid

MSE over sigmoid

- MSE leads to problems only if we predict >1, or <-1
- Simple solution: cut our predictions to [-1,+1] range



- But we need to do it in a smooth (differentiable) way
  - We can use sigmoid

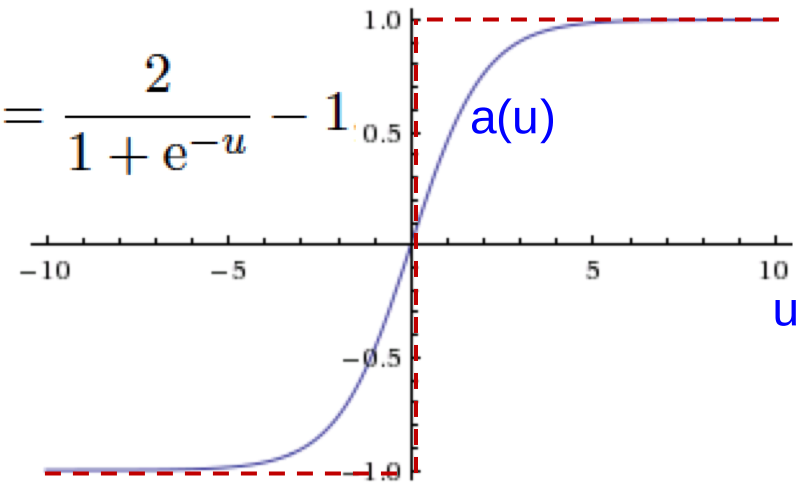


  - MSE-over-sigmoid-loss:
  - $(a(y_{pred}) - y_{true})^2$ instead of $(y_{pred} - y_{true})^2$

MSE over sigmoid

- We wrap prediction y_{pred} in bipolar sigmoid a()

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

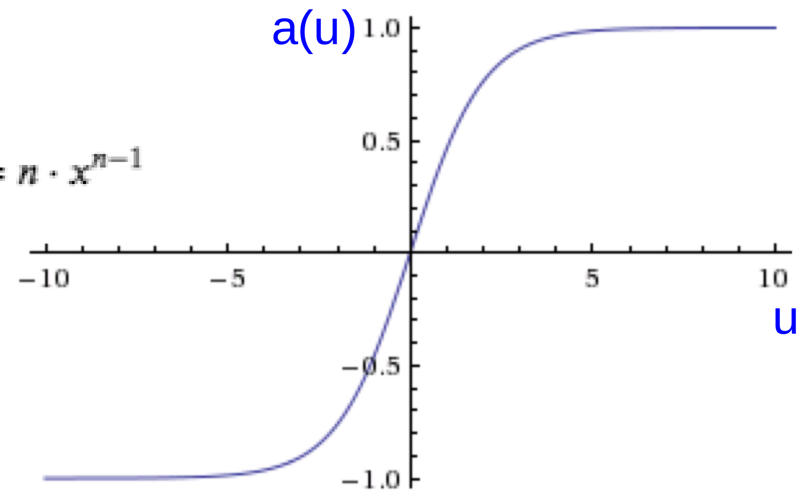


- Unlike 0/1 loss, it is smooth!
  - We can think of it as a smoothed-out sign function
  - Instead of sign(y_{pred}), which has flat regions ($\nabla_{\mathbf{w}} = 0$) we use a(y_{pred}), which has non-zero slope ($\nabla_{\mathbf{w}} \neq 0$)

MSE over sigmoid

- We wrap prediction y_{pred} in bipolar sigmoid a()

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



- Derivative of a(u) with respect to u is:

$$\frac{d}{dx} x^n = n \cdot x^{n-1}$$

$$\begin{aligned} a'(u) &= -\frac{2}{(1+e^{-u})^2} \cdot \frac{d}{du}(1+e^{-u}) \\ &= -\frac{2}{(1+e^{-u})^2} \cdot (-e^{-u}) \\ &= \frac{2e^{-u}}{(1+e^{-u})^2} \end{aligned}$$

- With further transformations (plug in a() on r.h.s and simplify) we can get:

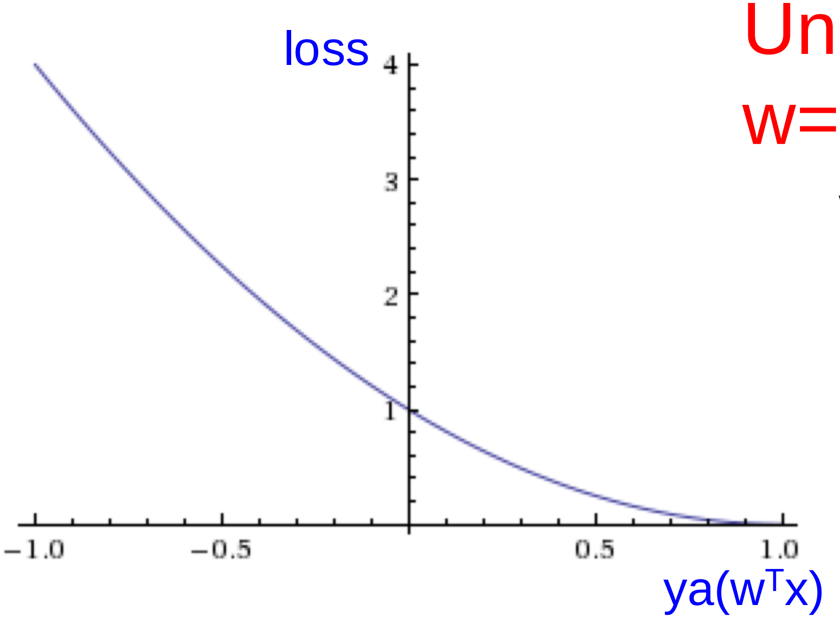
$$a'(u) = \frac{2\mathrm{e}^{-u}}{(1+\mathrm{e}^{-u})^2} = \frac{1}{2}\left(1 - a^2(u)\right)$$

MSE over sigmoid

- MSE on sigmoid as a loss (for y=+1/-1):

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

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



Unlike perceptron,
w=0 is no longer the best solution!

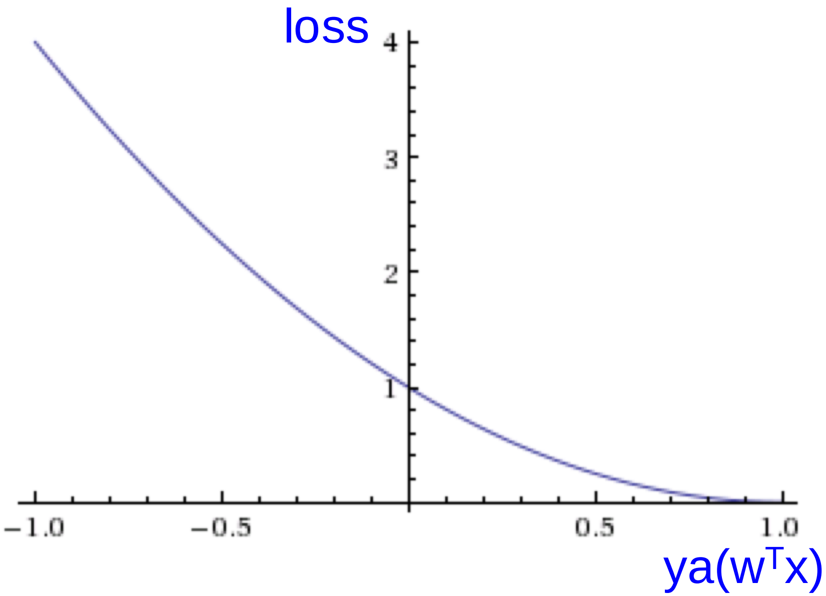
we won't be stuck with
"no prediction"
as a solution
that is always
a global minimum

MSE over sigmoid

- Loss = MSE on sigmoid:

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

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



Unlike MSE directly on $w^T x$ we don't penalize for correct predictions (loss goes to 0 for large $w^T x$, because $a(w^T x)$ goes to 1 as $w^T x$ goes to infinity)

MSE over sigmoid (grad only)

- Assuming:

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

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

- We can start deriving the update of weights:

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

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

chain rule

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

MSE over sigmoid (grad only)

- Assuming: $a(u) = \frac{2}{1+\mathrm{e}^{-u}} - 1$ $\quad a'(u) = \frac{2\mathrm{e}^{-u}}{(1+\mathrm{e}^{-u})^2} = \frac{1}{2}\left(1 - a^2(u)\right)$
- Continuing the derivation of the update of weights:

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

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

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

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

MSE over sigmoid (grad only)

- a linear model with MSE over sigmoid a($w^T x$):

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

- 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 a=bipolar sigmoid $a(u) = \frac{2}{1+\mathrm{e}^{-u}} - 1$ we get ($a'(u) = \frac{1}{2}\left(1 - a^2(u)\right)$):

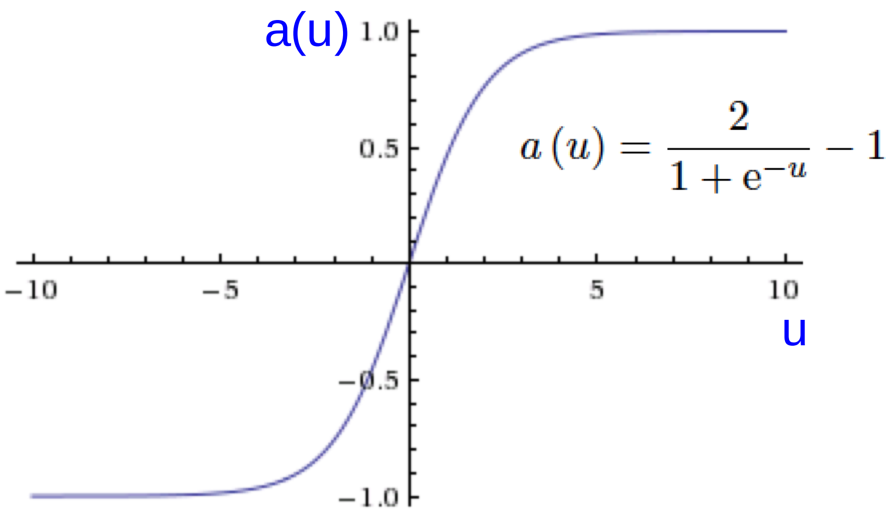
$$\mathbf{w}^\dagger{}_{t+1} = \mathbf{w}^\dagger{}_t - \frac{c}{2}\left.\frac{\partial \ell(h,\mathbf{z})}{\partial \mathbf{w}^\dagger}\right|_{\mathbf{w}^\dagger = \mathbf{w}^\dagger{}_t} = \mathbf{w}^\dagger{}_t + \frac{c}{2}\left[y - a\left({\mathbf{w}^\dagger}^T_t \mathbf{x}^\dagger\right)\right]\left[1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

MSE over sigmoid

- Problem with MSE over sigmoid / delta rule:

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

- 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)



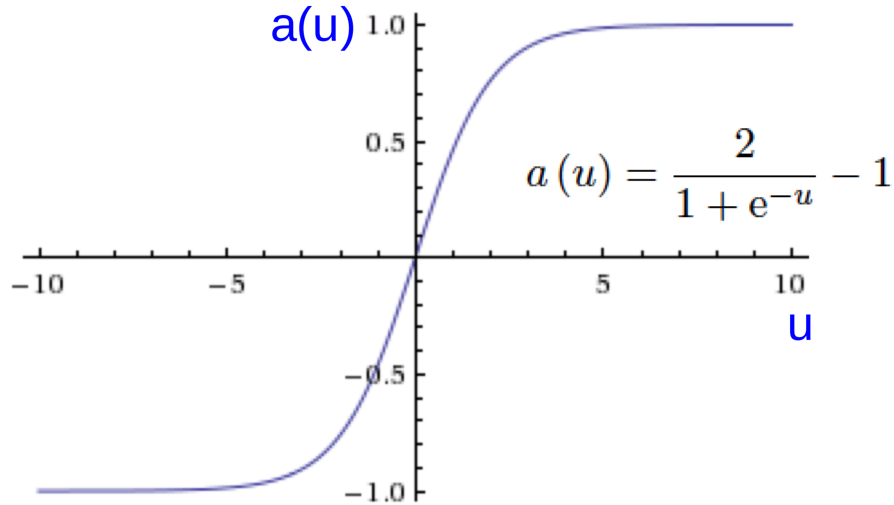
- Effect: Very slow learning for large |u|, including large and incorrect u

11.3. From MSE over sigmoid to logistic loss (a.k.a. cross-entropy)

MSE over sigmoid

- Problem with MSE over sigmoid / delta rule:

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



- The update depends on:
  - (y-a(u))
    - Smaller as we approach correct prediction
  - a'(u)
    - Much smaller (for uni- and bipolar sigmoid) as we approach correct prediction (*u* v.large)
    - And also very small when we have really incorrect prediction (*u* high magnitude, wrong sign)
- Effect: Very slow learning for large |u|, including large and incorrect u

Fixing delta rule: derive new loss

- Let's try with a unipolar sigmoid activation function
  - Suitable for encoding y_{true} as +1 or 0 (instead of +1/-1)

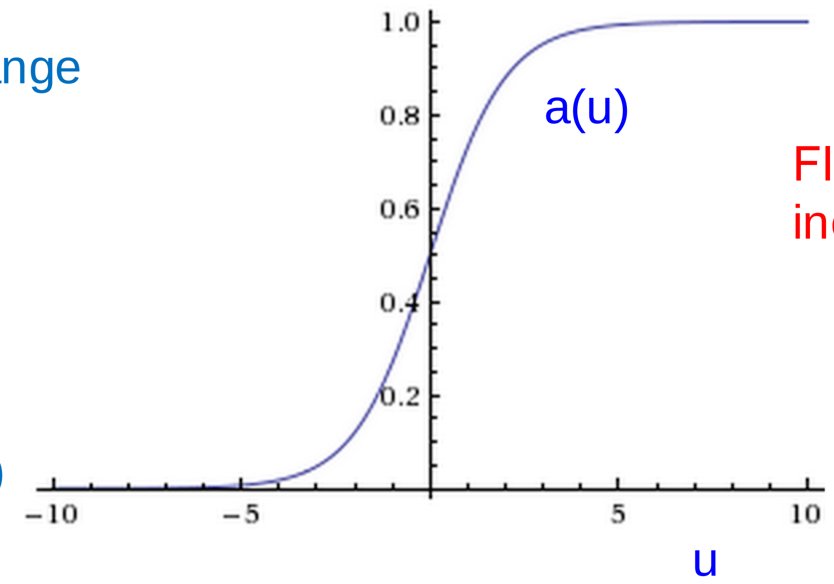
$$a(u) = \frac{1}{1+e^{-u}}$$

$$a(-u) = 1 - a(u)$$

$$a'(u) = a(u)(1 - a(u))$$

Bonus:

a(u) is in [0,1] range so prediction $a(w^T x+b)$ can be treated as probability: $p(+1)=a(w^T x+b)$ and $p(\text{not } +1)=1-a(u)$



Flat (tiny a') at both ends, including the incorrect one

Fixing MSE over sigmoid (grad-only)

$$a'(u) = a(u)(1-a(u))$$

- Problem with sigmoid+MSE:

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

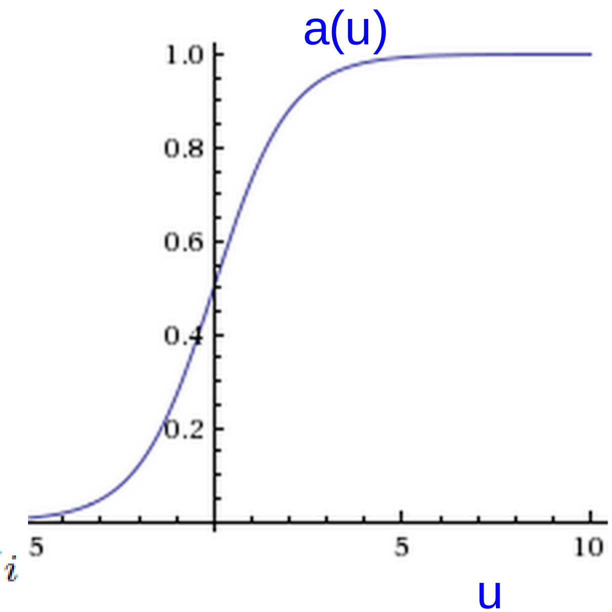
- Let's aim to get rid of the a′ term:

$$\boxed{-\frac{\partial \ell(a,y)}{\partial w_i} = (y-a)x_i}$$

- Our new loss would have to be:

$$\frac{\partial \ell}{\partial w_i} = \frac{\partial \ell(a,y)}{\partial a}\frac{\partial a(u)}{\partial u}\frac{\partial w^T x}{\partial w_i}$$

$$\frac{\partial \ell(a,y)}{\partial w_i} = \frac{\partial \ell(a,y)}{\partial a}\frac{\partial a(u)}{\partial u}\frac{\partial w^T x}{\partial w_i} = \frac{\partial \ell(a,y)}{\partial a}a(u)(1-a(u))x_i$$



$$\boxed{\frac{\partial \ell(a,y)}{\partial a} = -\frac{(y-a)}{a(1-a)}}$$

What formula for loss meets this equality?

Fixing MSE over sigmoid (grad-only)

$$a'(u) = a(u)(1 - a(u))$$

- Problem with sigmoid+MSE:

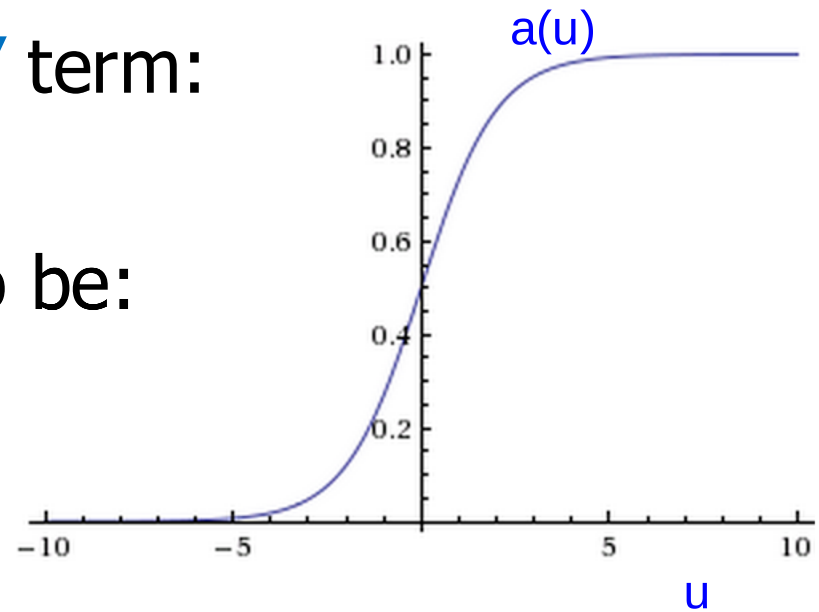
$$\mathbf{w}^{\dagger}{}_{t+1} = \mathbf{w}^{\dagger}{}_{t} - \frac{c}{2}\,\frac{\partial \ell\left(h, \mathbf{z}\right)}{\partial \mathbf{w}^{\dagger}}\bigg|_{\mathbf{w}^{\dagger} = \mathbf{w}^{\dagger}{}_{t}} = \mathbf{w}^{\dagger}{}_{t} + c\left[y - a\left(\mathbf{w}^{\dagger}{}_{t}^{T}\mathbf{x}^{\dagger}\right)\right]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

- Let's aim to get rid of the a′ term:

$$\boxed{-\frac{\partial \ell(a,y)}{\partial w_i} = (y-a)x_i}$$

- Our new loss would have to be:

$$\boxed{\frac{\partial \ell(a,y)}{\partial a} = -\frac{(y-a)}{a(1-a)}}$$



- We have y=1 or y=0, so we need:

$$\frac{\partial \ell(a,1)}{\partial a} = -\frac{(1-a)}{a(1-a)} = -\frac{1}{a}$$

$$\frac{\partial \ell(a,0)}{\partial a} = -\frac{(0-a)}{a(1-a)} = -\frac{-1}{1-a}$$

Fixing MSE over sigmoid (grad-only)

- Let's try to find a loss without a' term in gradient

$$\boxed{-\frac{\partial \ell(a,y)}{\partial w_i} = (y-a)x_i}$$

- Our loss would have to be: $\frac{\partial \ell(a,y)}{\partial a} = -\frac{(y-a)}{a(1-a)}$

$$\frac{\partial \ell(a,1)}{\partial a} = -\frac{(1-a)}{a(1-a)} = -\frac{1}{a} \qquad \frac{\partial \ell(a,0)}{\partial a} = -\frac{(0-a)}{a(1-a)} = -\frac{-1}{1-a}$$

- Formulas that use log() will work:

$$\frac{\partial -\log(a)}{\partial a} = -\frac{1}{a}$$

$$\frac{\partial -\log(1-a)}{\partial a} = \frac{\partial -\log(1-a)}{\partial 1-a}\frac{\partial(1-a)}{\partial a} = -\frac{-1}{1-a}$$

If we see derivative is 1/a then we think of the logarithm

  - In a form of a single equation:

$$\ell(a,y) = -[y\log(a) + (1-y)\log(1-a)]$$

y=1

y=0

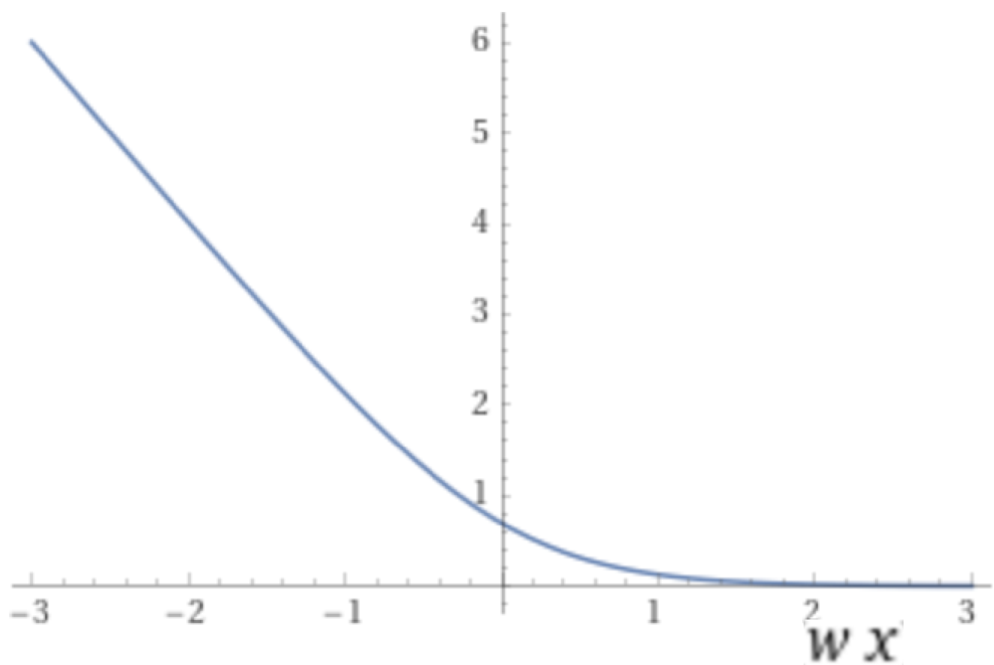
Cross-entropy loss

- For y=1 or 0, and for a=output of the model in [0,1] range, indicating probability of +1 class (a=p(+1)) being predicted p(+1),

$$\ell(a, y) = -[y \log(a) + (1-y)\log(1-a)]$$

  is called (binary) cross entropy

- The shape of the loss



Cross-entropy / logistic loss

- Let's try to find a loss without a' term in gradient

$$\boxed{-\frac{\partial \ell(a,y)}{\partial w_i} = (y-a)x_i}$$

- Our loss would have to be, for y=1 or 0 (cross-entropy loss):

$$\ell(a,y) = -[y\log(a) + (1-y)\log(1-a)]$$

- Plugging in the unipolar sigmoid $a(u) = \frac{1}{1+\mathrm{e}^{-u}}$:

observe the signs

$$\ell(a,1) = -\log(\frac{1}{1+e^{-w^T x}}) = \log(1+e^{-w^T x})$$

$$\ell(a,0) = -\log(1-\frac{1}{1+e^{-w^T x}}) = -\log(\frac{e^{-w^T x}}{1+e^{-w^T x}}) = \log(1+e^{w^T x})$$

- If we go back to y being y=+1 or -1 (logistic loss):

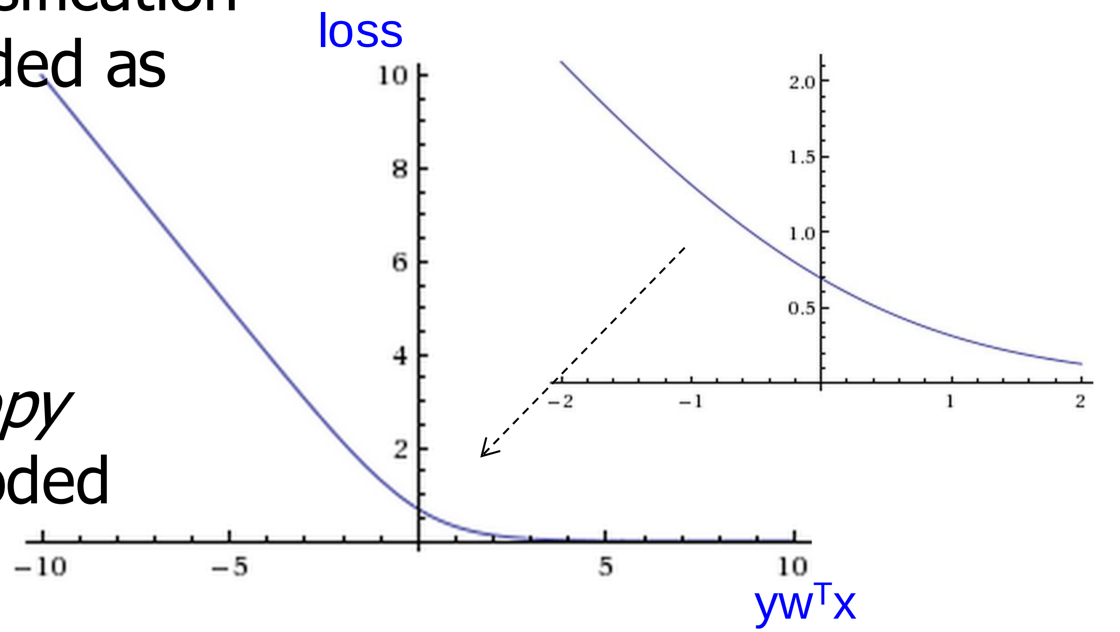
$$\ell(a,y) = \log(1+e^{-yw^T x})$$

Logistic loss

- **Logistic loss:**

$$\ell(h, z) = \ln(1 + \mathrm{e}^{-yw^T x})$$

- Derived from: $a(u) = \frac{1}{1+\mathrm{e}^{-u}}$
  - For two-class classification with classes encoded as y=+1/-1
  - The same as *binary cross entropy* if classes are encoded as y=1/0

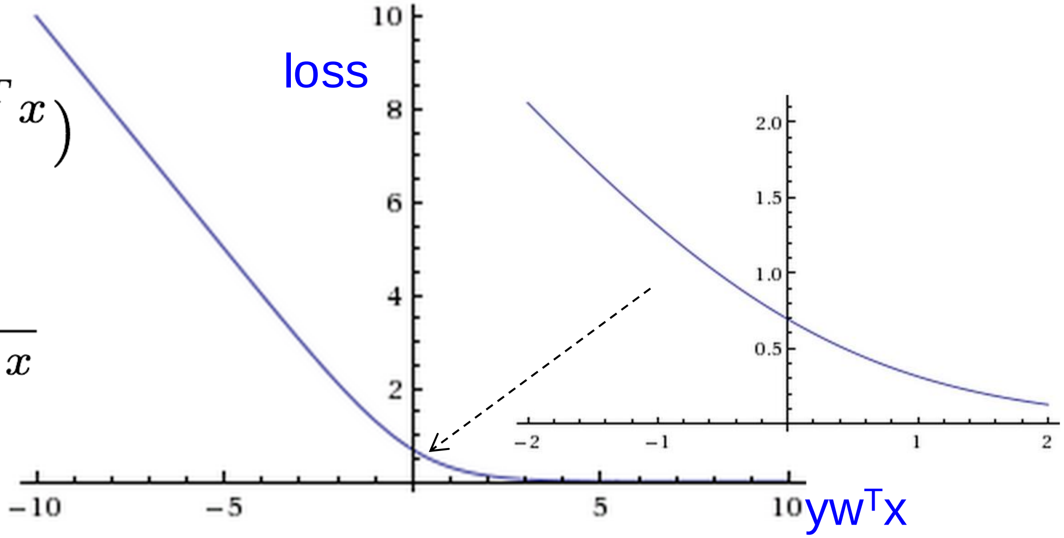


Logistic loss

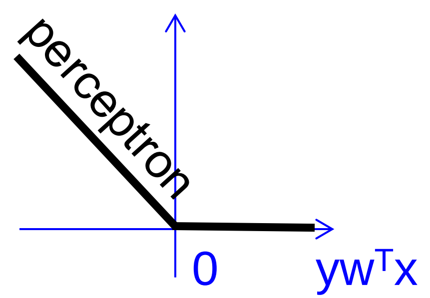
- Our derived loss function (for y=+1/-1):
  - Logistic loss / binary cross entropy loss

$$\ell(h, z) = \ln(1 + \mathrm{e}^{-yw^T x})$$

$$\nabla_w \ell(h, z) = \frac{yx}{1 + \mathrm{e}^{yw^T x}}$$

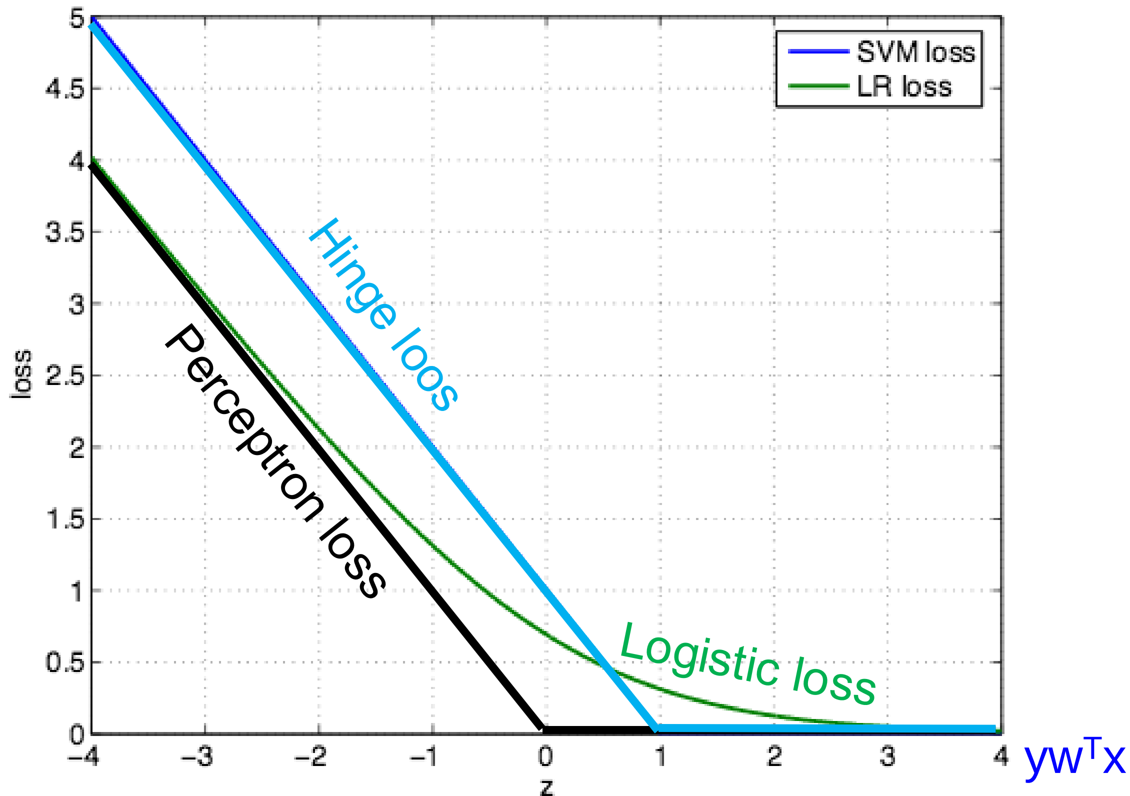


- Differentiable: gradient ∇_w easy to calculate
- Doesn't go to 0 too quickly for correct predictions
- Monotonic, non-increasing, no big penalty for correct predictions
- w=0 ("no prediction") does not lead to loss=0

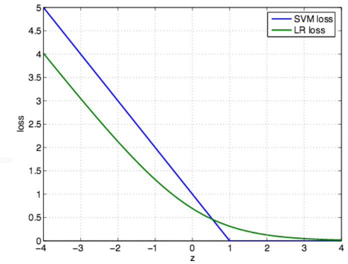


Another similar loss: Hinge loss

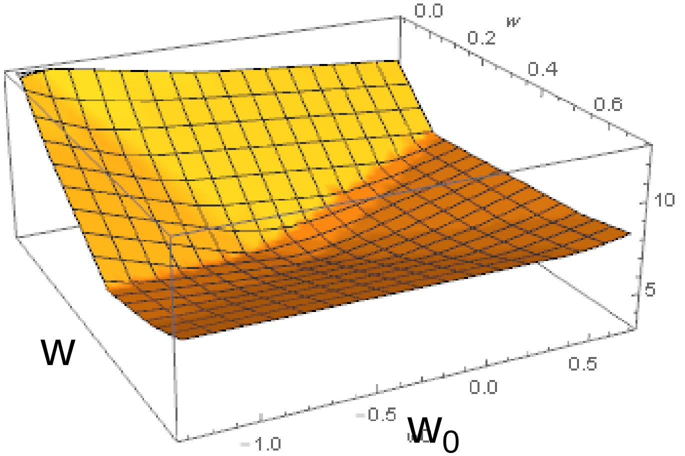
- Yet another loss: hinge loss
  - Loss = $[1\text{-}yh(x)]_+$ = max(0,1-yh(x))
- Popularized by Support Vector Machines
  - Like perceptron loss, but shifted to the right
    w=0 leads to loss=1

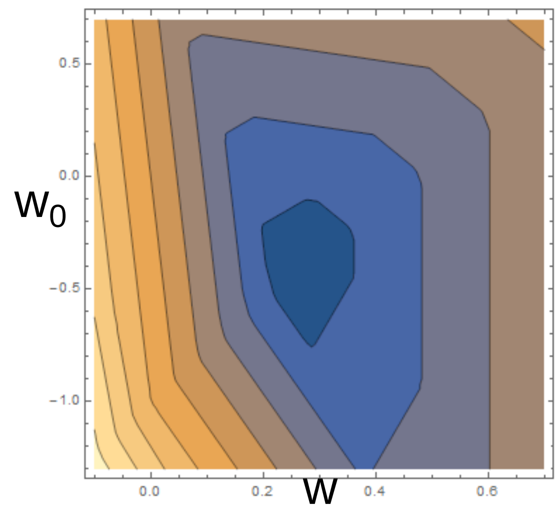


Support Vector Machine

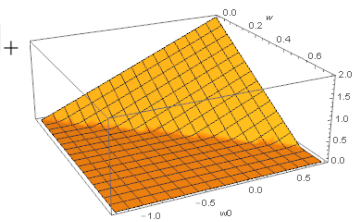


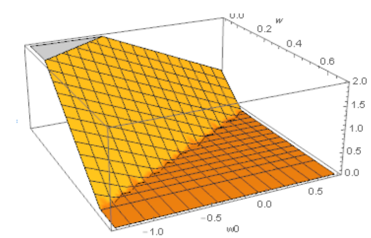
- Hinge empirical risk:



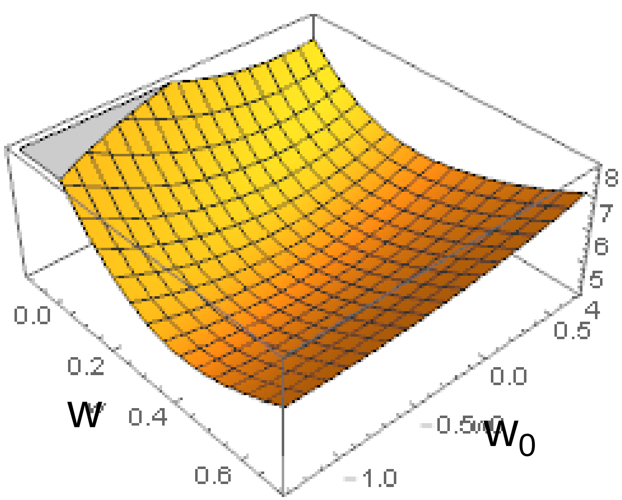


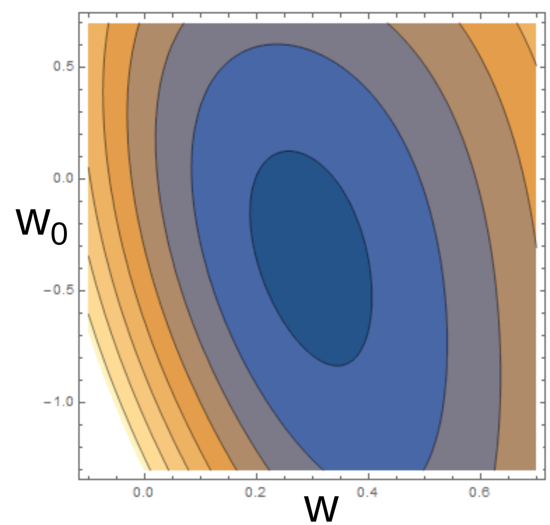
$$\sum_{i=1}^{m}[1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b)]_+$$



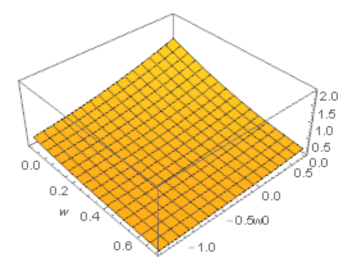


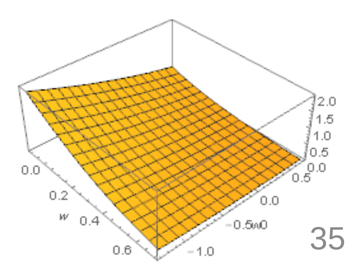
- Logistic emp. risk





$$\sum_{i=1}^{m}\ln(1 + e^{-y_i w^T x_i})$$





Recap

- Modern machine learning:

  $\texttt{minimize}_\theta\texttt{: error\_metric(}$
  $\quad\texttt{S, some\_f(}\theta\texttt{)(x in S).to\_class\_probs())}$

  - Filling the gaps:
    - `some_f(θ)(x)` or `some_f(θ;x)`
      - What family of functions? Simple choice that often works:
        `some_f(θ;x) = linear(w,b;x) =` $\mathbf{w^T x+b}$
        Linear models - simple and often very effective
    - `to_class_probs()`
      - `some_f(θ;x)` may not return probabilities, how to convert to: ≥0, sum=1
      - **for binary classification, use unipolar sigmoid a(**$\mathbf{w^T x+b}$**)**
    - `error_metric`
      - Classification error won't work. MSE is not ideal.
      - **Logistic loss (a.k.a. cross-entropy). Or hinge-loss.**
    - $\texttt{minimize}_\theta$
      - **Gradient descent, over parameters** θ **of** `some_f(θ;x)`

End of material for Test 1

- Test will be next Wednesday, 10/9

- See study problems in Canvas
  we will go over any questions related to them on Monday, 10/7