

★ We've seen ℓ_1 regularization (lasso):

$$\min_x \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda \|\mathbf{x}\|_1$$

as a way to promote sparsity. But if the goal is sparsity, why not penalize it directly? i.e.

$$\min_x \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda \text{nnz}(\mathbf{x})$$

where $\text{nnz}(\mathbf{x}) = \# \text{ of non-zeros}$?

★ we've seen least-squares classification:

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1^\top \\ \vdots \\ \mathbf{a}_m^\top \end{bmatrix} \text{ (rows of samples)}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} \text{ (vector of labels } \pm 1\text{)}.$$

and \mathbf{w} is vector of weights. We find weights by solving:

$$\begin{aligned} \min_w \|\mathbf{A}\mathbf{w} - \mathbf{b}\|_2^2 + \underbrace{\lambda \|\mathbf{w}\|_2^2}_{\text{regularize if needed}} &= \min_w \sum_{i=1}^m (\mathbf{a}_i^\top \mathbf{w} - b_i)^2 + \lambda \|\mathbf{w}\|_2^2. \\ &= \min_w \sum_{i=1}^m (1 - b_i \mathbf{a}_i^\top \mathbf{w})^2 + \lambda \|\mathbf{w}\|_2^2 \end{aligned}$$

Then error is # of incorrect classifications:

error if b_i and $\mathbf{a}_i^\top \mathbf{w}$ have different sign. i.e. if $\text{sign}(b_i \mathbf{a}_i^\top \mathbf{w}) = -1$.

$$\text{nerr}(\mathbf{w}) = \sum_{i=1}^m \frac{1}{2} (1 - \text{sign}(b_i \mathbf{a}_i^\top \mathbf{w})) = \# \text{ of misclassifications.}$$

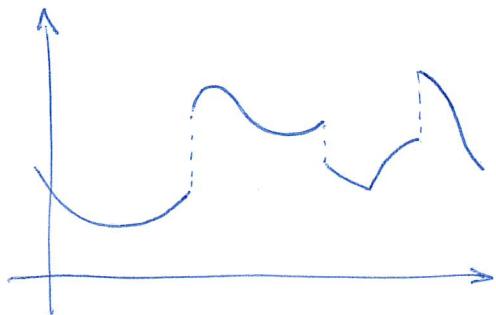
why not penalize this directly? i.e.

$$\min_w \text{nerr}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$$

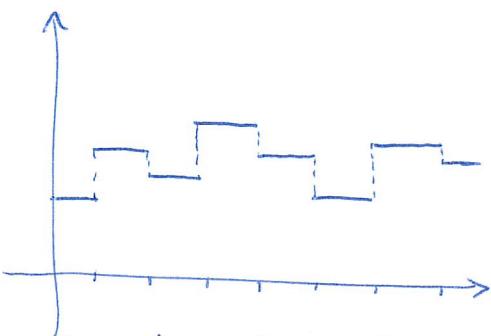
(2)

because these functions are difficult to optimize.

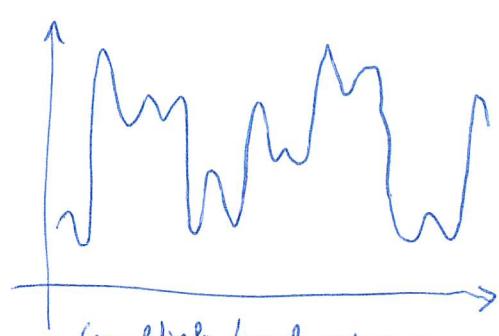
other examples of difficult functions:



(discontinuous, non-differentiable)

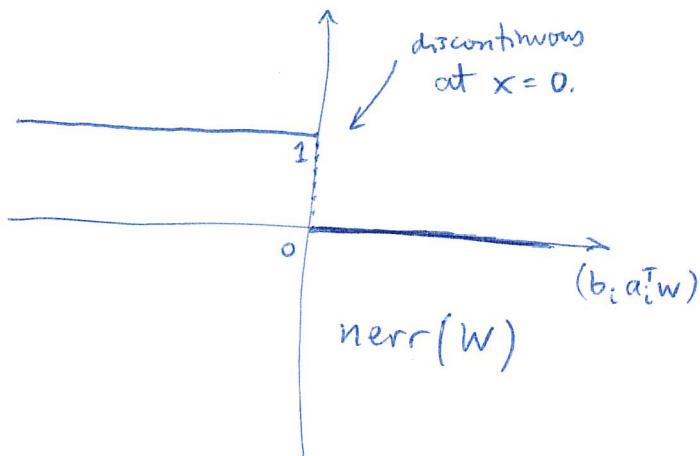
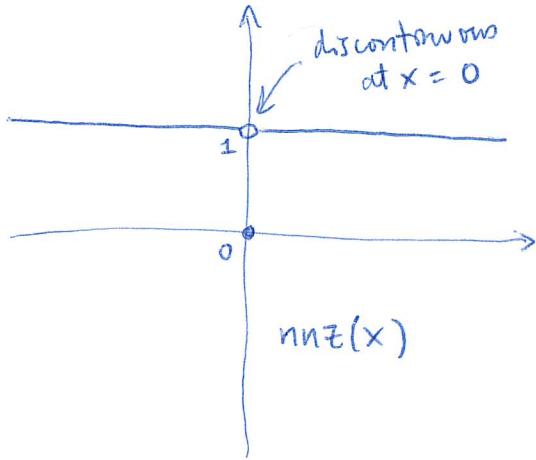


(discrete-valued function)



(multiple local minima,
despite being smooth).

the only way to ensure you have the minimum is to check entire function!



the only way to guarantee you have the optimal solution in these cases:

★ $nnz(x)$: which entries are zero? For each 2^n possible subset, solve a LS problem. Very costly!

★ $nerr(w)$: enumerate and check every possible classifier. For a finite set of points, there are finitely many (very large number) classifiers. Also very costly.

(3)

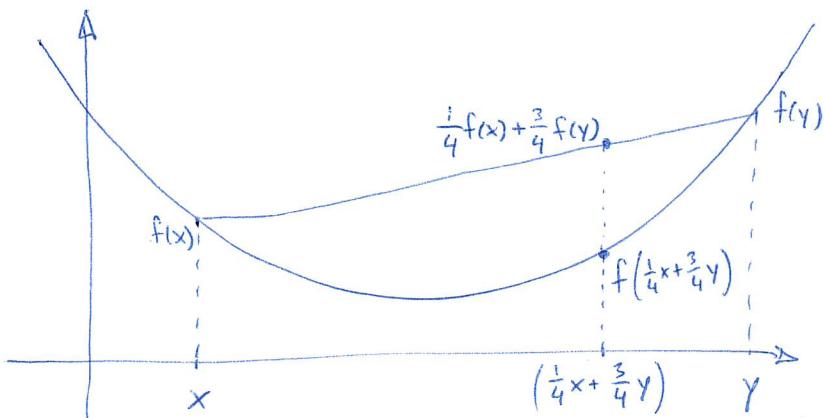
Convex functions

$f: \mathbb{R}^n \rightarrow \mathbb{R}$ is convex if for any $x, y \in \mathbb{R}^n$ and $0 \leq \alpha \leq 1$

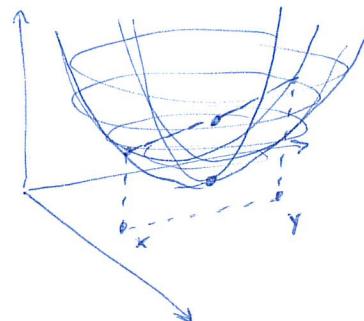
$$\underbrace{\alpha f(x) + (1-\alpha)f(y)}_{\text{convex combination}} \geq f(\underbrace{\alpha x + (1-\alpha)y}_{\text{linear combination}})$$

this is called a "convex combination".
similar to linear combination, but the
coefficients sum to 1

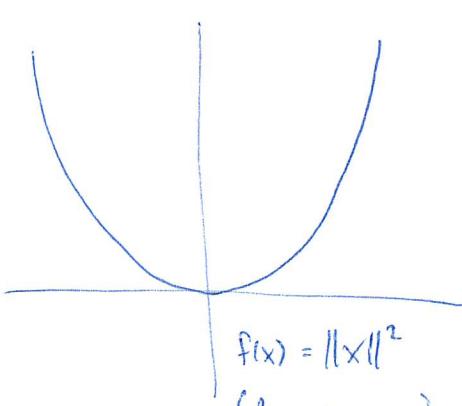
curve always lies below
any intersecting secants.



similar in higher dimensions:

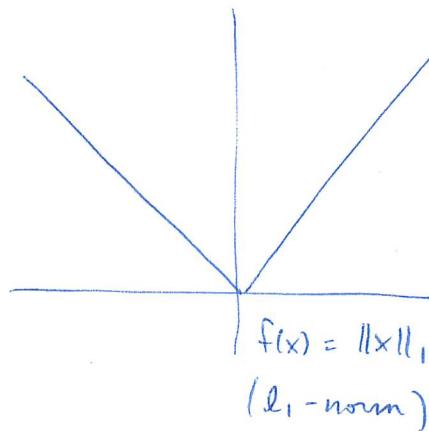


examples :



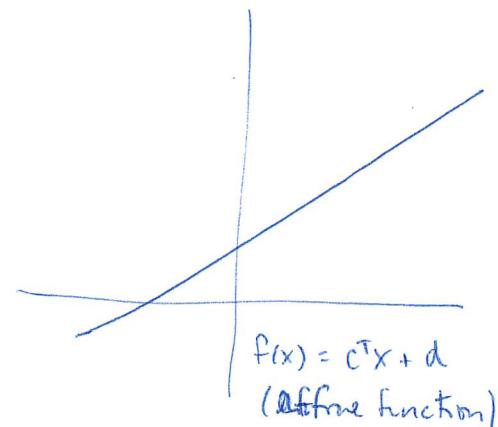
$$f(x) = \|x\|^2$$

(L₂-norm)



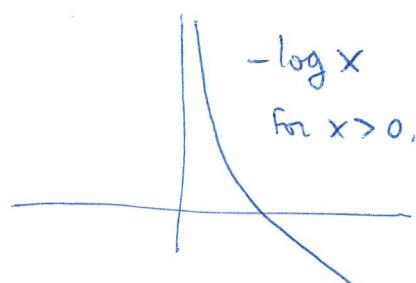
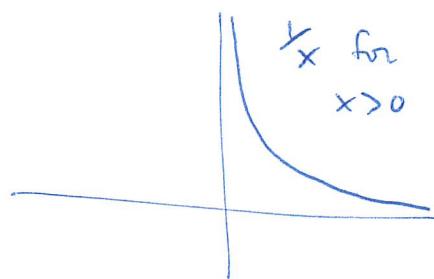
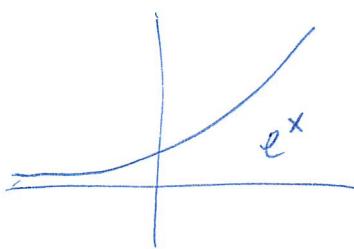
$$f(x) = \|x\|_1$$

(L₁-norm)



$$f(x) = c^T x + d$$

(affine function)



Also: sums of cvx functions are convex, max of cvx functions is convex.

benefits of convex functions

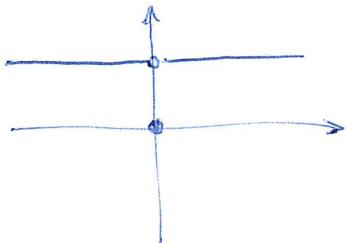
(4)

★ easy to optimize. (e.g. gradient descent).

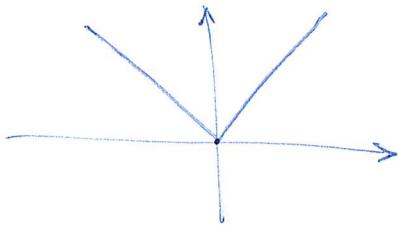
★ local optima are global optima.

★ bounds to optimality.

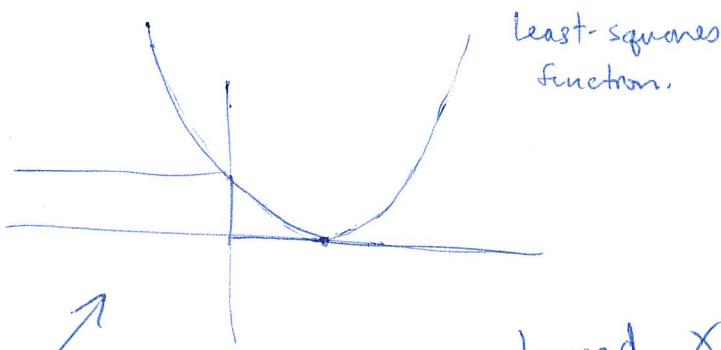
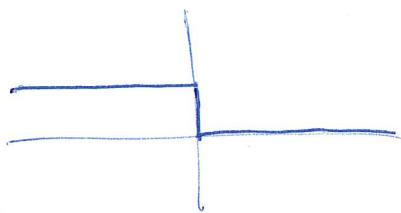
$\text{nnz}(x)$ not convex.



$\|x\|_1$ convex



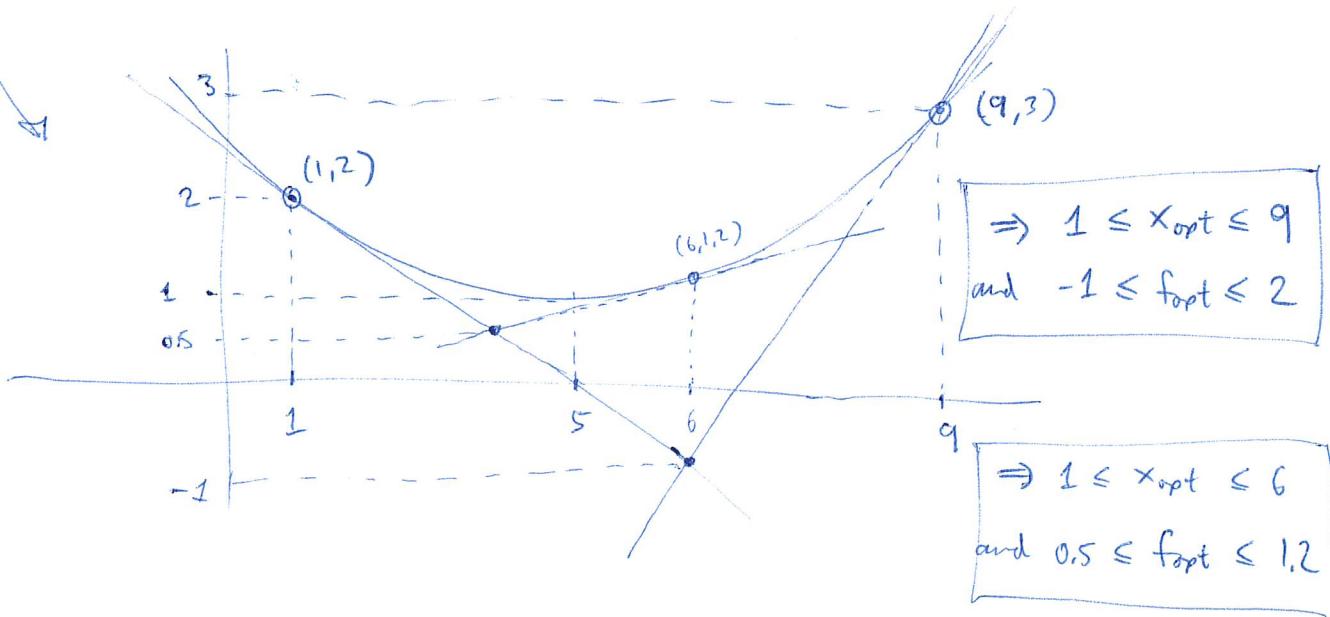
$\text{err}(w)$ not convex.



least-squares function.

more about this...

can bound x_{opt} and f_{opt} .



Classification

model:

$(a^T w)$ determines plane

(5)

what we want: minimize error.

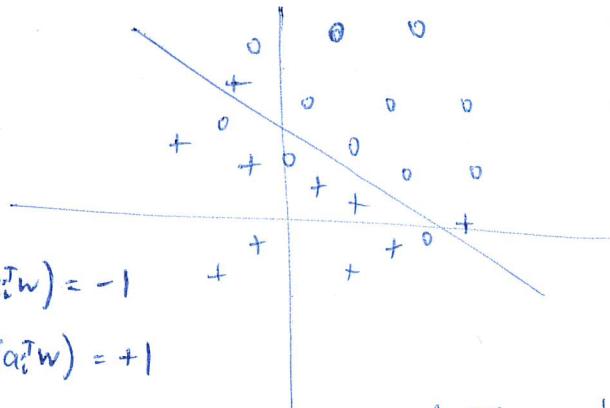
if $y_i = \text{label}$, $a_i^T w = \text{line}$,

count +1 if different sign \Rightarrow if $y_i \text{sign}(a_i^T w) = -1$

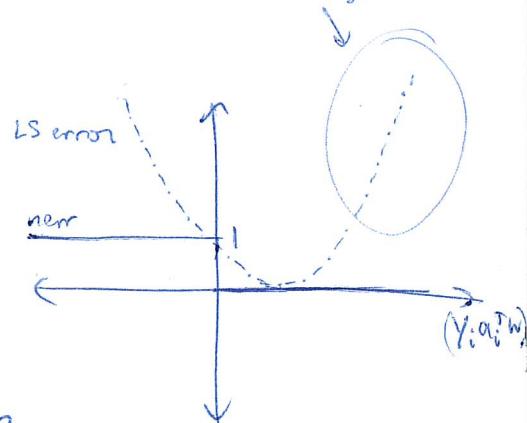
count +0 if same sign. \Rightarrow if $y_i \text{sign}(a_i^T w) = +1$

$$\Leftrightarrow \begin{cases} 1 & \text{if } \text{sign}(y_i a_i^T w) = -1 \\ 0 & \text{if } \text{sign}(y_i a_i^T w) = 1 \end{cases}$$

$$\Rightarrow \text{minimize}_{w \in \mathbb{R}} \sum_i \frac{1}{2} (1 - \text{sign}(y_i a_i^T w)).$$

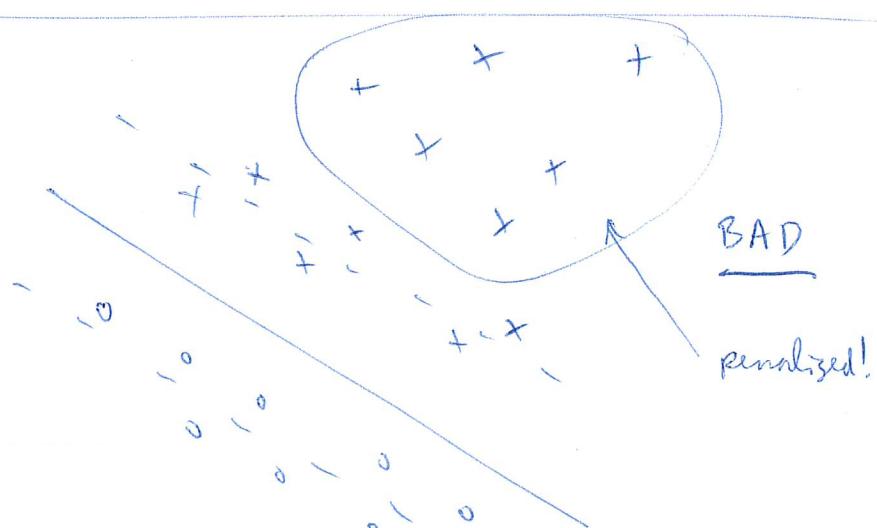
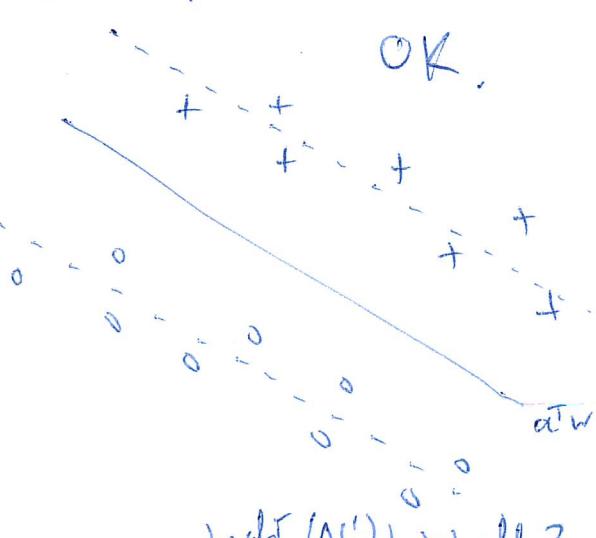


penalizes good classification



$$\text{LS error: } \sum_i (y_i - a_i^T w)^2 = \sum_i (1 - y_i a_i^T w)^2.$$

OK.



BAD

penalized!

example

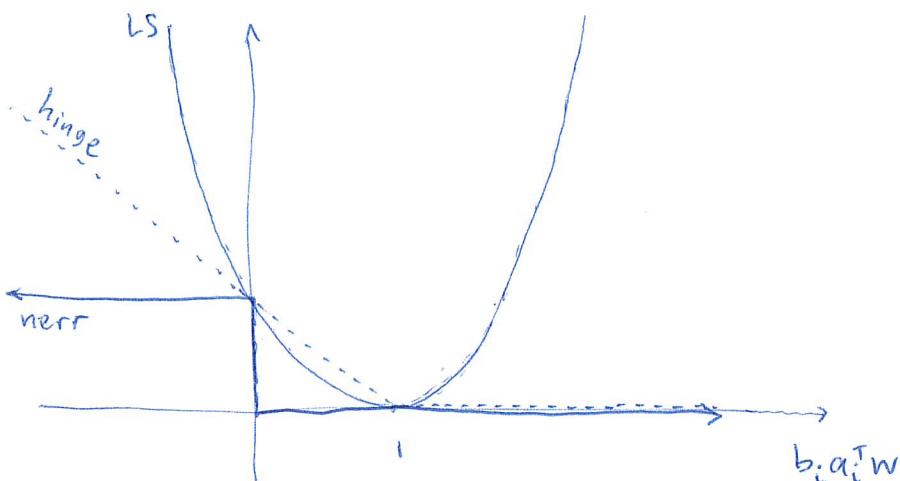
height (ft)	ball?
70	1
71	1
73	1
82	1
-2	-1
-1	-1
+1	1
+10	1

$$\begin{bmatrix} 70 & 1 \\ 71 & 1 \\ 73 & 1 \\ 82 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \approx \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix} \Rightarrow \hat{w} = \frac{1}{4} \begin{bmatrix} 7 \\ -518 \end{bmatrix}, \quad h = \frac{518}{7} = 74''$$

$$\Rightarrow [h \ 1] \hat{w} = 0 \Rightarrow h = \frac{518}{7} = 74''$$

(6)

Alternative way of doing classification



$$\text{nerr} : \frac{1}{2} (1 - \text{sign}(b_i a_i^T w))$$

$$\text{LS} : (1 - b_i a_i^T w)^2$$

$$\begin{aligned}\text{hinge loss} &: \max(0, 1 - b_i a_i^T w) \\ &= (1 - b_i a_i^T w)_+\end{aligned}$$

The idea is to minimize $\sum_{i=1}^m (1 - b_i a_i^T w)_+$ instead!

This is called the SVM ("support vector machine").

* it's a good idea to regularize (we'll see why later).

$$\Rightarrow \min_w \sum_{i=1}^m (1 - b_i a_i^T w)_+ + \lambda \|w\|^2$$

back to the basketball example:

let's shift heights to enter about 72":

$$A = \begin{bmatrix} -2 & 1 \\ -1 & 1 \\ 1 & 1 \\ 10 & 1 \end{bmatrix} \quad b = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \quad \text{classifies: } a_i^T w \approx b_i$$

if $w = \begin{bmatrix} x \\ y \end{bmatrix}$, prediction boundary

$$\text{is } [h \ 1] \begin{bmatrix} x \\ y \end{bmatrix} = 0 \Rightarrow h = -\frac{y}{x}$$

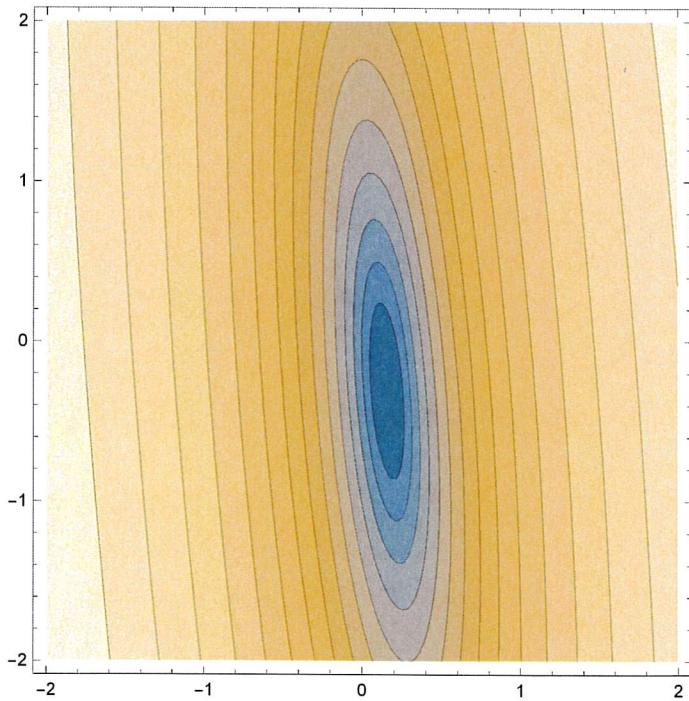
so if $\begin{cases} h > -y/x, \text{ classify } +1 \\ h < -y/x, \text{ classify } -1 \end{cases}$

(7)

$$\text{In[127]} = \mathbf{f} = \begin{pmatrix} (-1) & (-2x+y) \\ (-1) & (-1x+y) \\ (+1) & (1x+y) \\ (+1) & (10x+y) \end{pmatrix};$$

LS Classification

`In[128] = ContourPlot[Log[Sum[(1 - f[[i]])^2, {i, 1, 4}]], {x, -2, 2}, {y, -2, 2}, Contours -> 15]`



`In[129] = Sum[(1 - f[[i]])^2, {i, 1, 4}] [[1]]`

`Out[129]= (1 - 10x - y)^2 + (1 - x - y)^2 + (1 - 2x + y)^2 + (1 - x + y)^2`

`In[92] = Sum[(1 - f[[i]])^2, {i, 1, 4}] [[1]] // Expand`

`Out[92]= 4 - 28x + 106x^2 + 16xy + 4y^2`

`In[108] = Minimize[Sum[(1 - f[[i]])^2, {i, 1, 4}], {x, y}]`

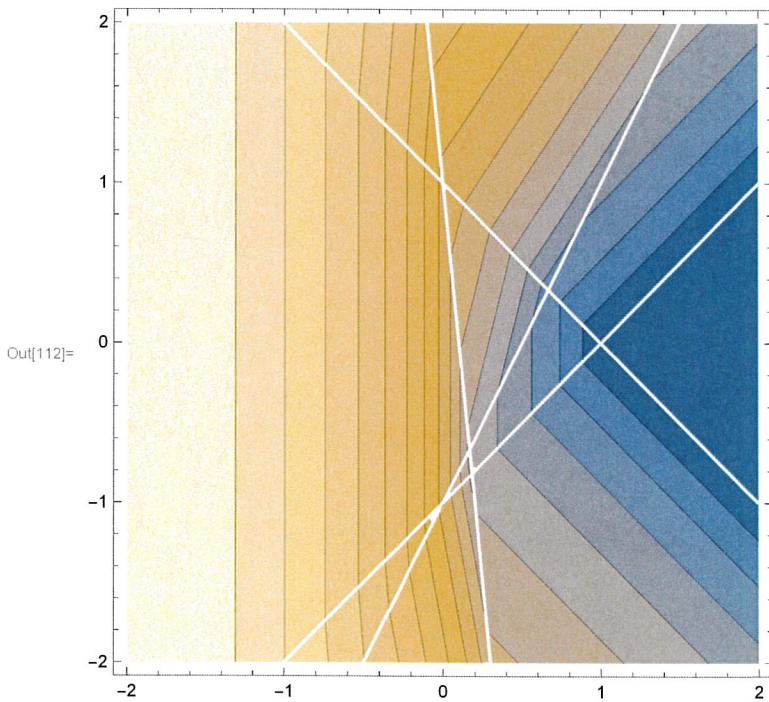
`Out[108]= {82/45, {x -> 7/45, y -> -14/45}}`

★ single optimal point

★ misclassification!

SVM classification

```
In[112]= ContourPlot[Log[Sum[Max[0, (1 - f[[i]])], {i, 1, 4}] + 1],
{x, -2, 2}, {y, -2, 2}, Contours -> 15, PlotPoints -> 25]
```



```
In[109]= Sum[Max[0, (1 - f[[i]])], {i, 1, 4}]
```

```
Out[109]= Max[0, 1 - 10 x - y] + Max[0, 1 - x - y] + Max[0, 1 - 2 x + y] + Max[0, 1 - x + y]
```

```
In[110]= Minimize[Sum[Max[0, (1 - f[[i]])], {i, 1, 4}], {x, y}]
```

```
Out[110]= {0, {x -> 1, y -> 0}}
```

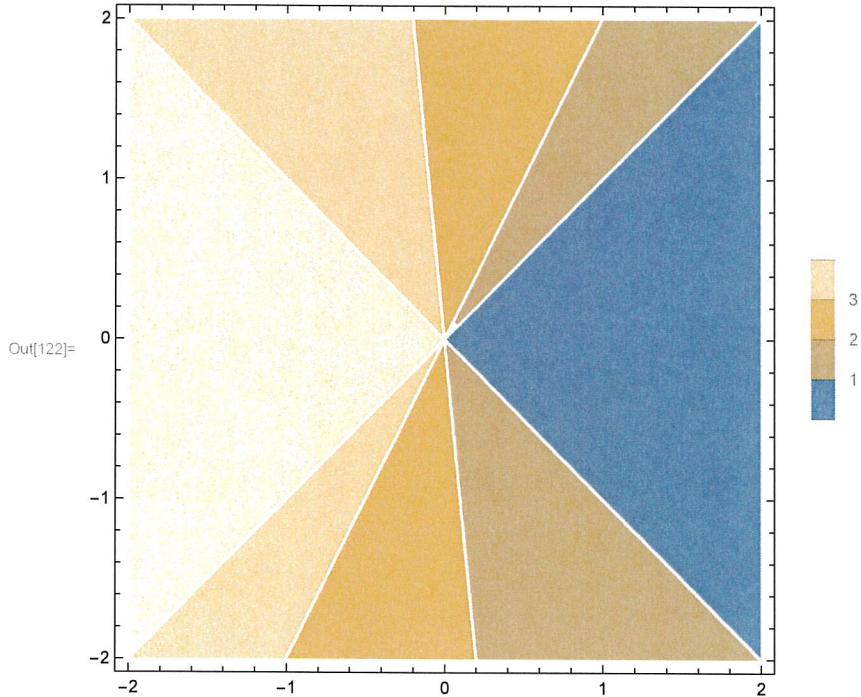
★ many optimal points (with zero loss)

★ if we regularize, we find $(1, 0)$, smallest norm point.

★ zero classification error.

Exact (non convex) classification

```
In[122]= ContourPlot[ $\frac{1}{2} \text{Sum}[(1 - \text{Sign}[f[[i]]]), \{i, 1, 4\}]$ , {x, -2, 2}, {y, -2, 2},  
Contours → {0, 1, 2, 3, 4}, PlotPoints → 50, PlotLegends → Automatic]
```



```
In[124]=  $\frac{1}{2} \text{Sum}[(1 - \text{Sign}[f[[i]]]), \{i, 1, 4\}] [[1]]$   
Out[124]=  $\frac{1}{2} (4 - \text{Sign}[x - y] - \text{Sign}[2x - y] - \text{Sign}[x + y] - \text{Sign}[10x + y])$ 
```