Machine Learning

Linear Models

Fabio Vandin

October 27^{th}, 2023

Back to Our Linear Classification Problem

- binary classification problem: $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{-1, 1\}$
- with linear models
- with loss $\ell(\mathbf{w}, (\mathbf{x}, y)) = \max\{0, -y\langle \mathbf{w}, \mathbf{x}\rangle\}$.

How to find the ERM solution? SGD!

SGD for Linear Classification

SGD: take i uniformly at random from $\{1,\dots,m\}$. Let $(\vec{x}', y')$ be the corresponding point in the training set, and consider the vector $\nabla \ell(\vec{w}, (\vec{x}', y'))$

Note that GD considers (as gradient of the function to minimize):

$$\nabla L_S(\vec{w}) = \frac{1}{m} \sum_{i=1}^{m} \nabla \ell(\vec{w}, (\vec{x}_i, y_i))$$

and for SGD we have:

$$\mathbb{E}[\nabla \ell(\vec{w}, (\vec{x}', y'))] = \sum_{i=1}^{m} \underbrace{\Pr[(\vec{x}', y') = (\vec{x}_i, y_i)]}_{\frac{1}{m}\ \forall i \in \{1,\dots,m\}\ \text{(uniform distribution)}} \cdot \nabla \ell(\vec{w}, (\vec{x}_i, y_i))$$

$$= \frac{1}{m} \sum_{i=1}^{m} \nabla \ell(\vec{w}, (\vec{x}_i, y_i))$$

$$= \nabla L_S(\vec{w})$$

SGD algorithm:

$\vec{w}^{(0)} \leftarrow \vec{0}$;

for $t \leftarrow 0$ to $T-1$ do {

pick i uniformly at random from $\{1, \dots, m\}$;

$\vec{w}^{(t+1)} \leftarrow \vec{w}^{(t)} - \eta \underbrace{\nabla \ell(\vec{w}^{(t)}, (\vec{x}_i, y_i))}_{?}$; ($\star$)

}

return $\bar{w} = \left(\sum_{t=1}^{T} \vec{w}^{(t)} \right) / T$;

To simplify the notation, we are going to compute

$\nabla \ell(\vec{w}, (\vec{x}_i, y_i))$

$$\nabla \ell(\vec{w}, (\vec{x}_i, y_i)) = \begin{cases} \vec{0} & \text{if } y_i \langle \vec{w}, \vec{x}_i \rangle > 0 \\ \nabla(-y_i \langle \vec{w}, \vec{x}_i \rangle) & \text{otherwise} \end{cases}$$

if $\vec{x}_i$ is correctly classified by $\vec{w}$

if $\vec{x}_i$ is wrongly classified by $\vec{w}$

Assume that $y_i < \vec{w}, \vec{x}_i > < 0$

$$\nabla\left(-y_i < \vec{w}, \vec{x}_i >\right) = \left[\frac{\partial(-y_i < \vec{w}, \vec{x}_i >)}{\partial w_1}, \dots, \frac{\partial(-y_i < \vec{w}, \vec{x}_i >)}{\partial w_d} \right]^T$$

Let $\vec{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{bmatrix}$. Since $-y_i < \vec{w}, \vec{x}_i > = -y_i \cdot \sum_{j=1}^{d} (w_j x_{ij})$

$$\Rightarrow \frac{\partial(-y_i < \vec{w}, \vec{x}_i >)}{\partial w_j} = -y_i x_{ij}$$

$$\Rightarrow \nabla \ell(\vec{w}, (\vec{x}_i, y_i)) = \left[-y_i x_{i1}, -y_i x_{i2}, \dots, -y_i x_{id} \right]^T$$

$$= -y_i \vec{x}_i$$

Therefore, in the pseudocode, (★) is replaced by

if $y_i < \vec{w}^{(t)}, \vec{x}_i > < 0$ then {

$\vec{w}^{(t+1)} \leftarrow \vec{w}^{(t)} + \eta\, y_i \vec{x}_i$;

}

Comparison :

perceptron vs SGD perceptron

1) choose a missclassified point | choose a point at random (not only a missclassified one)

2) $\eta = 1$ | η is a parameter

3) return "best" $\vec{w}^{(t)}$ | return $\bar{w}$

Main difference: 1), but we can "speed up" the SGD perceptron by, at each iteration, pick a missclassified point at random

$\Rightarrow$ SGD perceptron is the perceptron

Linear Regression

$\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$

Linear Regression

$\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$

Hypothesis class:

$$\mathcal{H}_{reg} = L_d = \{\mathbf{x} \to \langle \mathbf{w}, \mathbf{x} \rangle + b : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

Linear Regression

$\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$

Hypothesis class:

$$\mathcal{H}_{reg} = L_d = \{\mathbf{x} \rightarrow \langle \mathbf{w}, \mathbf{x} \rangle + b : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

Note: $h \in \mathcal{H}_{reg} : \mathbb{R}^d \rightarrow \mathbb{R}$

Commonly used loss function: *squared-loss*

$$\ell(h, (\mathbf{x}, y)) \stackrel{def}{=} (h(\mathbf{x}) - y)^2$$

ERM for regression with linear models
and squared loss

Linear Regression

$\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$

Hypothesis class:

$$\mathcal{H}_{reg} = L_d = \{\mathbf{x} \to \langle \mathbf{w}, \mathbf{x} \rangle + b : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

Note: $h \in \mathcal{H}_{reg} : \mathbb{R}^d \to \mathbb{R}$

Commonly used loss function: *squared-loss*

$$\ell(h, (\mathbf{x}, y)) \stackrel{def}{=} (h(\mathbf{x}) - y)^2$$

$\Rightarrow$ empirical risk function (training error): *Mean Squared Error*

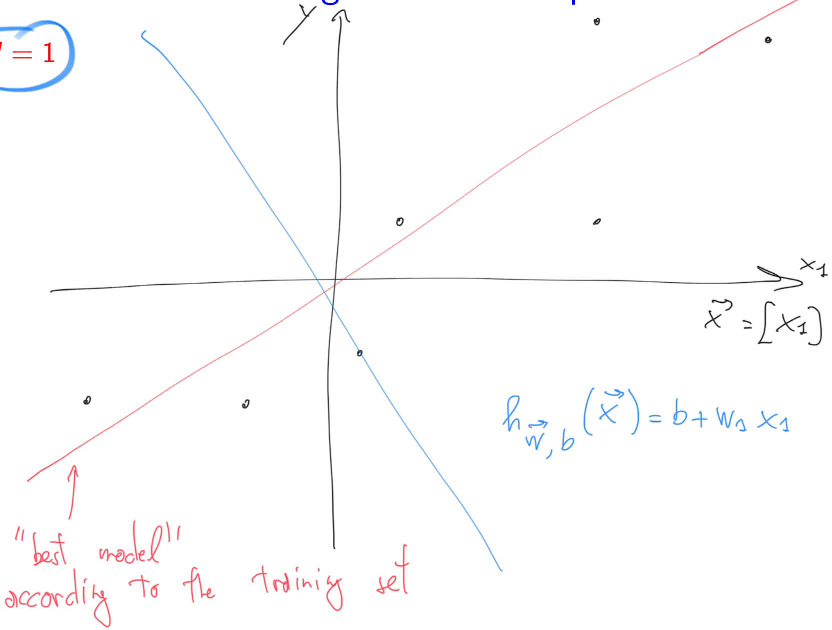
training set

$S = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_m, y_m)\}$

$$L_S(h) = \frac{1}{m} \sum_{i=1}^{m} \underbrace{(h(\mathbf{x}_i) - y_i)^2}_{\ell(h, (\vec{x}_i, y_i))}$$

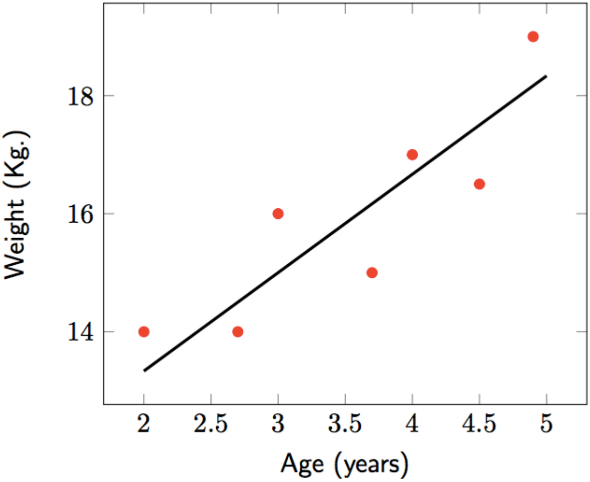
Linear Regression - Example

$d = 1$



Linear Regression - Example

$d = 1$



Least Squares

How to find a ERM hypothesis? *Least Squares* algorithm

$\vec{w} = [b, w_1, \dots, w_d]^T$

Best hypothesis:

$$\arg\min_{\mathbf{w}} L_S(h_{\mathbf{w}}) = \arg\min_{\mathbf{w}} \cancel{\frac{1}{m}} \sum_{i=1}^{m} (\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i)^2$$

Least Squares

How to find a ERM hypothesis? *Least Squares* algorithm

Best hypothesis:

$$\arg\min_{\mathbf{w}} L_S(h_{\mathbf{w}}) = \arg\min_{\mathbf{w}} \frac{1}{m}\sum_{i=1}^{m}(\langle \mathbf{w}, \mathbf{x}_i\rangle - y_i)^2$$

Equivalent formulation: $\mathbf{w}$ minimizing *Residual Sum of Squares* (RSS), i.e.

$$\arg\min_{\mathbf{w}} \sum_{i=1}^{m}(\langle \mathbf{w}, \mathbf{x}_i\rangle - y_i)^2$$

RSS: Matrix Form

Let

$$\mathbf{X} = \begin{bmatrix} \cdots & \mathbf{x}_1 & \cdots \\ \cdots & \mathbf{x}_2 & \cdots \\ \cdots & \vdots & \cdots \\ \cdots & \mathbf{x}_m & \cdots \end{bmatrix}$$

$S = \{ (\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_m, y_m) \}$

$\mathbf{X}$: *design matrix*

RSS: Matrix Form

Let

$$\mathbf{X} = \begin{bmatrix} \cdots & \mathbf{x}_1 & \cdots \\ \cdots & \mathbf{x}_2 & \cdots \\ \cdots & \vdots & \cdots \\ \cdots & \mathbf{x}_m & \cdots \end{bmatrix}$$

$\mathbf{X}$: *design matrix*

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

$\Rightarrow$ we have that RSS is

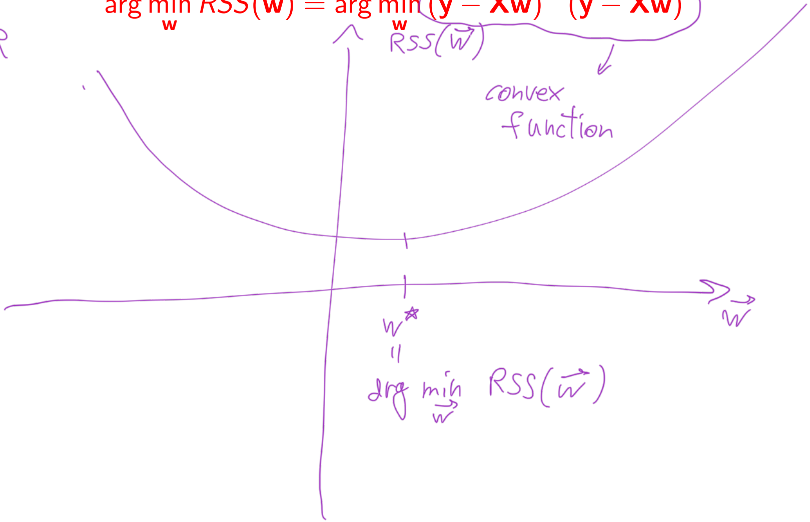
HW: check that it is true

$$\sum_{i=1}^{m} (\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i)^2 = (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w})$$

Want to find **w** that minimizes RSS (=*objective function*):

$$\arg\min_{\mathbf{w}} RSS(\mathbf{w}) = \arg\min_{\mathbf{w}} (\mathbf{y}-\mathbf{X}\mathbf{w})^T (\mathbf{y}-\mathbf{X}\mathbf{w})$$

$\vec{w} \in \mathbb{R}$



Want to find $\mathbf{w}$ that minimizes RSS (=*objective function*):

$$\arg\min_{\mathbf{w}} RSS(\mathbf{w}) = \arg\min_{\mathbf{w}} \left(\mathbf{y} - \mathbf{X}\mathbf{w}\right)^T \left(\mathbf{y} - \mathbf{X}\mathbf{w}\right)$$

How?

Compute gradient $\frac{\partial RSS(\mathbf{w})}{\partial \mathbf{w}}$ of objective function w.r.t $\mathbf{w}$ and compare it to 0.

$$\frac{\partial RSS(\mathbf{w})}{\partial \mathbf{w}} = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w})$$

Then we need to find $\mathbf{w}$ such that

$$-2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) = 0$$

$$-2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) = 0$$

$$-2X^T\vec{y} + 2X^TX\vec{w} = 0$$

$$\cancel{2}X^TX\vec{w} = \cancel{2}X^T\vec{y}$$

$$X^TX\vec{w} = X^T\vec{y}$$

$$(X^TX)^{-1}X^TX\vec{w} = (X^TX)^{-1}X^T\vec{y}$$

$$\vec{w} = (X^TX)^{-1}X^Ty$$

$$-2\mathbf{X}^T(\mathbf{y}-\mathbf{X}\mathbf{w}) = 0$$

is equivalent to

$$\mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{y}$$

If $\mathbf{X}^T\mathbf{X}$ is invertible $\Rightarrow$ solution to ERM problem is:

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Complexity Considerations

We need to compute

$$(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$S = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_m, y_m)\}$

$\vec{x}_i \in \mathbb{R}^{d+1}$

Complexity Considerations

We need to compute

$$(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Algorithm:

1. compute $\mathbf{X}^T\mathbf{X}$: product of $(d+1) \times m$ matrix and $m \times (d+1)$ matrix
2. compute $(\mathbf{X}^T\mathbf{X})^{-1}$ inversion of $(d+1) \times (d+1)$ matrix
3. compute $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$: product of $(d+1) \times (d+1)$ matrix and $(d+1) \times m$ matrix
4. compute $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$: product of $(d+1) \times m$ matrix and $m \times 1$ matrix

Most expensive operation? Inversion!

$\Rightarrow$ done for $(d+1) \times (d+1)$ matrix

$\mathbf{X}^T\mathbf{X}$ not invertible?

How do we get $\mathbf{w}$ such that

$$\mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{y}$$

if $\mathbf{X}^T\mathbf{X}$ is not invertible?
Let

$$\mathbf{A} = \mathbf{X}^T\mathbf{X}$$

Let $\mathbf{A}^+$ be the *generalized inverse* of $\mathbf{A}$, i.e.:

$$\mathbf{A}\mathbf{A}^+\mathbf{A} = \mathbf{A}$$

$\mathbf{X}^T\mathbf{X}$ not invertible?

How do we get $\mathbf{w}$ such that

$$\mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{y}$$

if $\mathbf{X}^T\mathbf{X}$ is not invertible?
Let

$$\mathbf{A} = \mathbf{X}^T\mathbf{X}$$

Let $\mathbf{A}^+$ be the *generalized inverse* of $\mathbf{A}$, i.e.:

$$\mathbf{A}\mathbf{A}^+\mathbf{A} = \mathbf{A}$$

Proposition

If $\mathbf{A} = \mathbf{X}^T\mathbf{X}$ is not invertible, then $\hat{w} = \mathbf{A}^+\mathbf{X}^T\mathbf{y}$ is a solution to $\mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{y}$.

Computing the Generalized Inverse of **A**

Note $\mathbf{A} = \mathbf{X}^T\mathbf{X}$ is symmetric $\Rightarrow$ eigenvalue decomposition of **A**:

$$\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^T$$

$$D = \begin{bmatrix} \square & & & & 0 \\ & \square & & & \\ & & \square & & \\ & & & \ddots & \\ 0 & & & & \square \end{bmatrix}$$

with

- **D**: diagonal matrix (entries = eigenvalues of **A**)
- **V**: orthonormal matrix ($\mathbf{V}^T\mathbf{V} = \mathbf{I}_{d\times d}$)

$$I_{d\times d} = \left.\overbrace{\begin{bmatrix} 1 & & & 0 \\ & 1 & & \\ & & \ddots & \\ 0 & & & 1 \end{bmatrix}}^{d}\right\} d$$

$$A\vec{x} = \lambda\vec{x}$$

eigenvector eigenvalue

Computing the Generalized Inverse of $\mathbf{A}$

Note $\mathbf{A} = \mathbf{X}^T\mathbf{X}$ is symmetric $\Rightarrow$ eigenvalue decomposition of $\mathbf{A}$:

$$\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^T$$

with

- $\mathbf{D}$: diagonal matrix (entries = eigenvalues of $\mathbf{A}$)
- $\mathbf{V}$: orthonormal matrix ($\mathbf{V}^T\mathbf{V} = \mathbf{I}_{d \times d}$)

$$D = \begin{bmatrix} D_{11} & & & & 0 \\ & D_{22} & & & \\ & & \ddots & & \\ & & & 0 & \\ 0 & & & & 0 \end{bmatrix}$$

Define $\mathbf{D}^+$ diagonal matrix such that:

$$\mathbf{D}^+_{i,i} = \begin{cases} 0 & \text{if } \mathbf{D}_{i,i} = 0 \\ \frac{1}{\mathbf{D}_{i,i}} & \text{otherwise} \end{cases}$$

$$DD^+ = \begin{bmatrix} 1 & & & & 0 \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & 0 \\ 0 & & & & 0 \end{bmatrix} \qquad D^+ = \begin{bmatrix} \frac{1}{D_{1,1}} & & & & 0 \\ & \frac{1}{D_{22}} & & & \\ & & \ddots & & \\ & & & 0 & \\ 0 & & & & 0 \end{bmatrix}$$

Let $\mathbf{A}^+ = \mathbf{VD}^+\mathbf{V}^T$

Is it the generalized inverse for A?

Show: $AA^+A = A$

$$AA^+A = \underbrace{VDV^T}_{A}\,\underbrace{VD^+V^T}_{A^+}\,\underbrace{VDV^T}_{A}$$

since V is orthonormal: $V^TV = I$

$$= V\underbrace{DD^+D}_{D}V^T$$

$$= VDV^T = A$$

Let $\mathbf{A}^+ = \mathbf{V}\mathbf{D}^+\mathbf{V}^T$

Then

$$\begin{aligned} \mathbf{A}\mathbf{A}^+\mathbf{A} &= \mathbf{V}\mathbf{D}\mathbf{V}^T\mathbf{V}\mathbf{D}^+\mathbf{V}^T\mathbf{V}\mathbf{D}\mathbf{V}^T \\ &= \mathbf{V}\mathbf{D}\mathbf{D}^+\mathbf{D}\mathbf{V}^T \\ &= \mathbf{V}\mathbf{D}\mathbf{V}^T \\ &= \mathbf{A} \end{aligned}$$

$\Rightarrow$ $\mathbf{A}^+$ is a generalized inverse of $\mathbf{A}$.

Let $\mathbf{A}^+ = \mathbf{V}\mathbf{D}^+\mathbf{V}^T$

Then

$$\begin{aligned}\mathbf{A}\mathbf{A}^+\mathbf{A} &= \mathbf{V}\mathbf{D}\mathbf{V}^T\mathbf{V}\mathbf{D}^+\mathbf{V}^T\mathbf{V}\mathbf{D}\mathbf{V}^T \\ &= \mathbf{V}\mathbf{D}\mathbf{D}^+\mathbf{D}\mathbf{V}^T \\ &= \mathbf{V}\mathbf{D}\mathbf{V}^T \\ &= \mathbf{A}\end{aligned}$$

$\Rightarrow$ $\mathbf{A}^+$ is a generalized inverse of $\mathbf{A}$.

In practice: the Moore-Penrose generalized inverse $\mathbf{A}^\dagger$ of $\mathbf{A}$ is used, since it can be efficiently computed from the Singular Value Decomposition of $\mathbf{A}$.

Exercise

Consider a linear regression problem, where $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}$, with mean squared loss. The hypothesis set is the set of *constant* functions, that is $\mathcal{H} = \{h_a : a \in \mathbb{R}\}$, where $h_a(\mathbf{x}) = a$. Let $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$ denote the training set.

- Derive the hypothesis $h \in \mathcal{H}$ that minimizes the training error.
- Use the result above to explain why, for a given hypothesis $\hat{h}$ from the set of all linear models, the coefficient of determination $R^2 = 1 - \frac{\sum_{i=1}^m (\hat{h}(\mathbf{x}_i) - y_i)^2}{\sum_{i=1}^m (y_i - \bar{y})^2}$ where $\bar{y}$ is the average of the y_i,$i = 1, \ldots, m$ is a measure of how well $\hat{h}$ performs (on the training set).

Exercise

Your friend has developed a new machine learning algorithm for binary classification (i.e., $y \in \{-1, 1\}$) with 0-1 loss and tells you that it achieves a generalization error of only 0.05. However, when you look at the learning problem he is working on, you find out that $\Pr_{\mathcal{D}}[y = 1] = 0.95$...

- Assume that $\Pr_{\mathcal{D}}[y = \ell] = p_\ell$. Derive the generalization error of the (dumb) hypothesis/model that *always* predicts ℓ.
- Use the result above to decide if your friend's algorithm has learned something or not.

Exercise

Assume we have the following training set S, where $\mathbf{x} = [x_1, x_2] \in \mathbb{R}^2$ and $\mathcal{Y} = \{-1, 1\}$:

$S = \{([-3, 4], 1), ([2, -3], -1), ([-3, -4], -1), ([1, 1.5], 1)\}$.

Assume you decide to use $\mathcal{H} = \{h_1, h_2, h_3, h_4\}$ with

$h_1 = sign(-x_1 - x_2)$

$h_2 = sign(-x_1 + x_2)$

$h_3 = sign(x_1 - x_2)$

$h_4 = sign(x_1 + x_2)$

Your algorithm uses the ERM rule and the 0-1 loss.

- What model h_S is produced in output by your ML algorithm?
- Assume the realizability assumption holds. What can you say about the generalization error $L_{\mathcal{D}}(h_S)$ of h_S?