Perspective & connection to DS/ML:

In data science / machine learning we often need to optimize functions of the form

$$\boxed{\underbrace{\min_{w}}_{\text{model parameters}} \frac{1}{N}\sum_{i=1}^{N} \underbrace{f}_{\text{loss function}}(w; \underbrace{(x_i, y_i)}_{\text{training examples or data}}) + \lambda \underbrace{R}_{\text{regularization}}(w)}$$

Examples / applications:

* linear regression
* logistic regression
* least squares fit to a model
* PCA
* Neural network loss
* Support vector machines
* (K-means) clustering

...

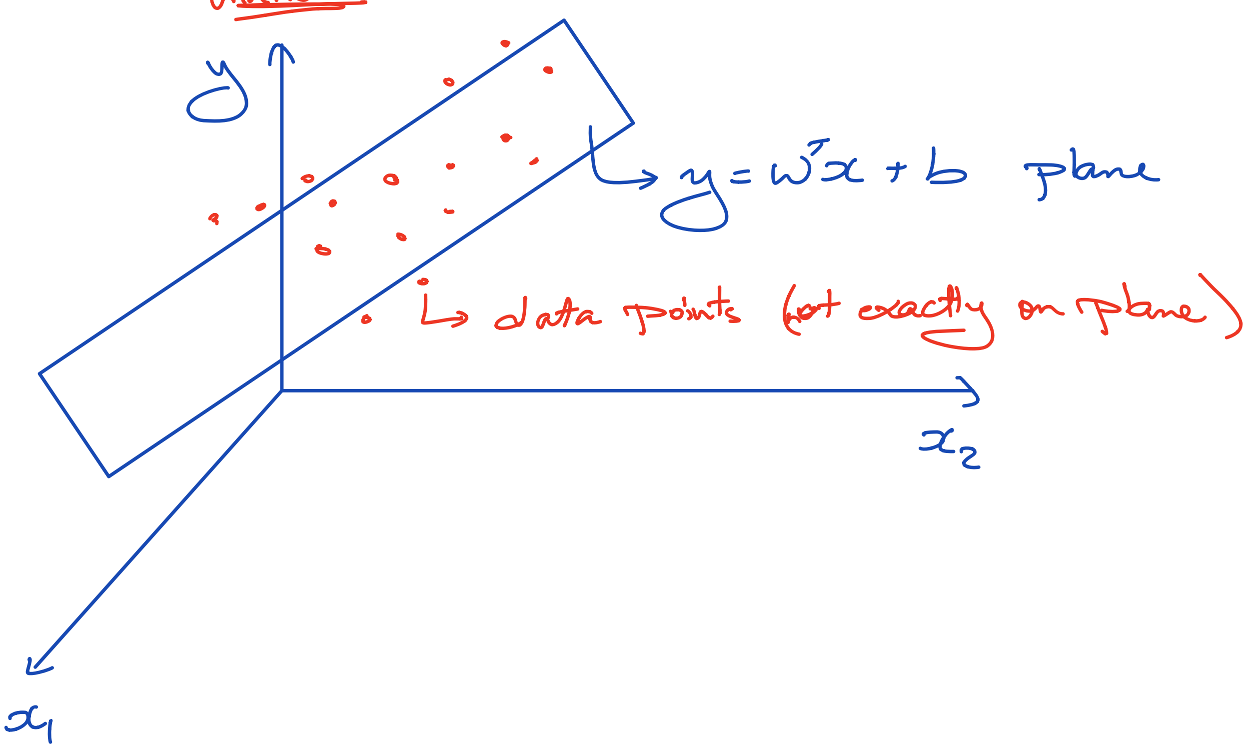
× **Linear regression:**

Given data $(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, $i = 1, \dots, N$

"regressors" → x_i ; y_i ← dependent variable

linear regression assumes that the relationship between y_i & x_i is linear, up to an error or noise term so

$$y_i = \underbrace{w}_{\in \mathbb{R}^d \text{ unknown}}{}^{T} \underbrace{x_i}_{\in \mathbb{R}^d} + \overbrace{b}^{\in \mathbb{R} \text{ unknown}} + \underbrace{\varepsilon_i}_{\in \mathbb{R},\ \text{noise},\ \text{unknown}}$$



Goal: estimate w & b

under some assumptions (e.g. ε_i is Gaussian random variable), it makes sense to solve

$$\boxed{\min_{w,b} \frac{1}{N} \sum_{i=1}^{N} |w^T x_i + b - y_i|^2}$$

(linear least squares, has closed form solution)

Sometimes we would like our model to be "parsimonious", i.e., we would like most entries of w to be zero, so we introduce the regularizer

$$R(w) = \|w\|_1 = \sum_{i=1}^{d} |w_i|$$

and we solve

$$\boxed{\min_{w,b} \frac{1}{N} \sum_{i=1}^{N} |w^T x_i + b - y_i|^2 + \lambda \underline{\|w\|_1}}$$

"Lasso", no closed form sol'n

not differentiable

* Logistic regression (Classification)

Data : $(x_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}$, $i = 1, \dots, N$

Logistic regression assumes that the relationship between y_i & x_i is

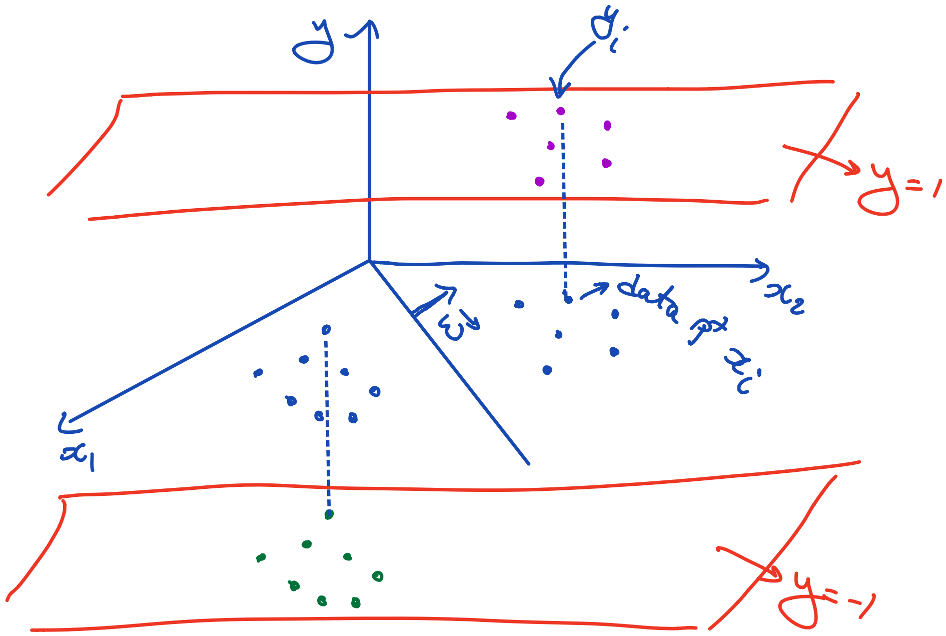
$$y_i = \text{sign}(w^T x_i + b + \varepsilon_i)$$
$$= \text{sign}(\tilde{w}^T \tilde{x}_i + \varepsilon_i)$$

"noise", "error"

↳ CDF $\frac{1}{1+e^{-z}}$

where $\tilde{w} = (w, b) \in \mathbb{R}^{d+1}$
$\tilde{x} = (x, 1) \in \mathbb{R}^{d+1}$

So $w^T x + b = \tilde{w}^T \tilde{x}$



Subject of a lot of our assignments

$$\boxed{\min_{w} \frac{1}{N} \sum_{i=1}^{N} \log\left(1 + e^{-w^T x_i y_i}\right)}$$

* Least squares fit to a model

Example: HW 2 where we fit the parameters of a hyperbola based on data.

* Nonlinear least squares:

Data: (x_i, y_i) as before

Model: $y_i = h_w(x_i)$

nonlinear function parametrized by w

Example: $y = b e^{w^T x}$

$$\boxed{\min_{\substack{b \in \mathbb{R} \\ w \in \mathbb{R}^d}} \frac{1}{N} \sum_{i=1}^{N} |y_i - b e^{w^T x_i}|^2}$$

Can lead to non-convex opt. problems

(training) Neural Networks:

Neural nets are functions of the form

$$y = g_L \circ g_{L-1} \circ \cdots \circ g_1(x) =: G_{W,b}(x)$$

composition

$$= g_L(g_{L-1}(\cdots g_2(g_1(x))))$$

where $g_j(z) = \rho(W^{(j)} z + b^{(j)})$

non-linear function

matrix

vector

e.g. relu $\rho(z) = \max(0, z)$

Given data (x_i, y_i), $i = 1, \ldots, N$

want to solve for $(W^{(j)}, b^{(j)})$, $j = 1, \ldots, L$

$\Rightarrow$ Solve

$$\min_{\substack{W^{(j)}, b^{(j)} \\ j=1,\dots,L}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left((W^{(j)}, b^{(j)}); (x_i, y_i) \right)$$

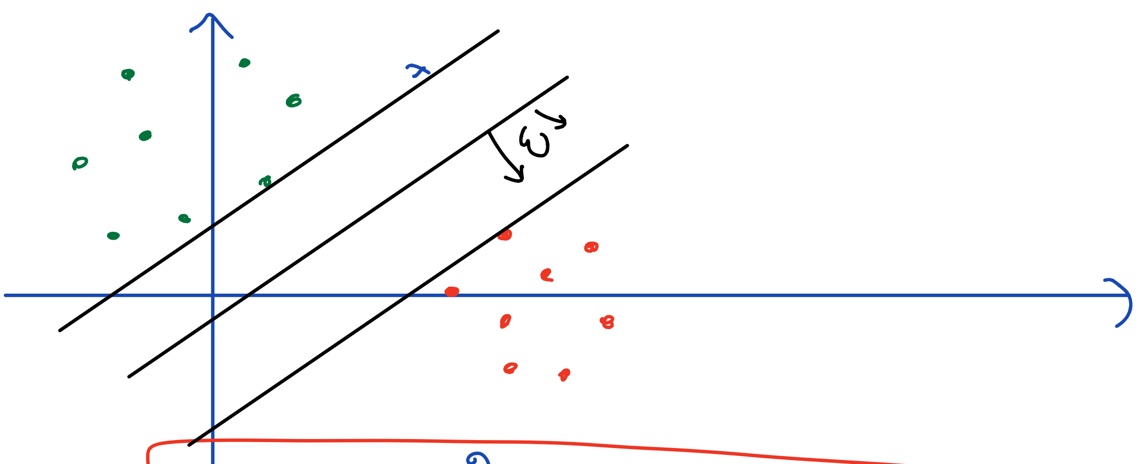
some loss function

No closed form, can be hard to optimize even with methods from this course

$\Rightarrow$ Stochastic GD (173 B)

Support Vector Machines (Classification)

Data: $(x_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}$, $i = 1, \dots, N$



$$\max_{w, b} \frac{2}{\|w\|}$$

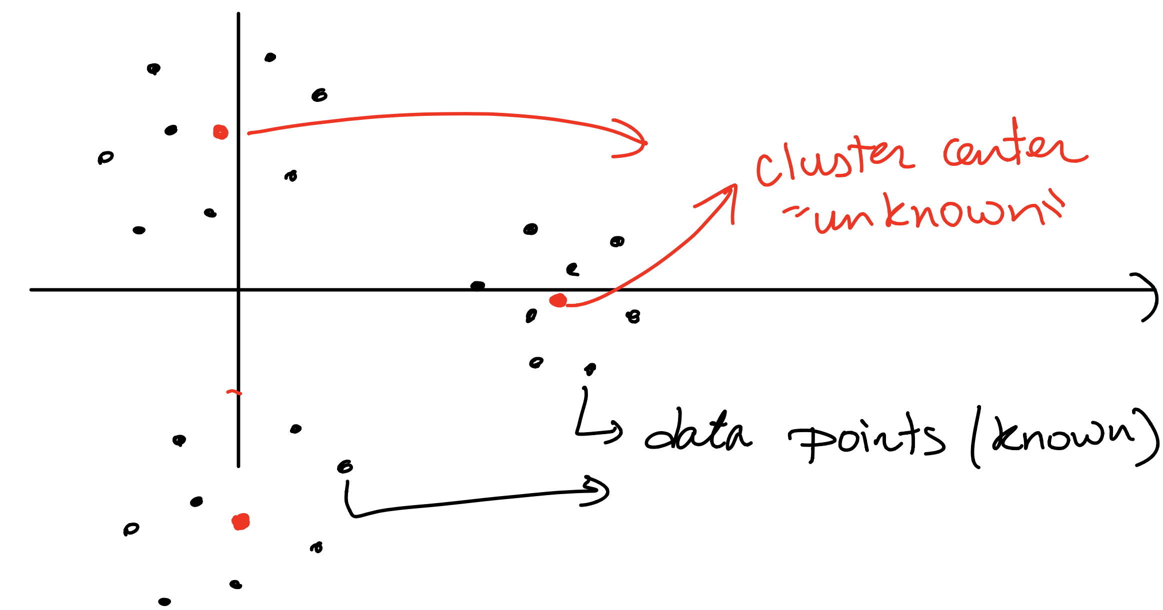
s.t. $\begin{cases} \langle w, x_i \rangle + b \geq 1 & \forall \text{ red point} \\ \langle w, x_j \rangle + b \leq -1 & \forall \text{ green point} \end{cases}$

constrained problem!

much more on this in math 173 B

K-means Clustering:

Given data $x_i \in \mathbb{R}^d$ (no y_i's)



the goal is to "cluster" the data into groups where x_i's in the same cluster are closer to their cluster "center" than to other cluster centers.

clusters

$$\underset{S}{\text{minimize}} \sum_{i=1}^{k} \sum_{x \in S_i} \| x - \mu_i \|^2$$

k → number of clusters

μ_i → cluster centers

S_i → the clusters (sets of x's)

→ turns out to be NP-hard (tough to find algorithms guaranteed to solve this)

Lots of good heuristics algorithms.

...