16B Prof. Anant Sahai

Today: Finish up solving systems of nonlinear equations
Revisiting Classification.
Think about "losses"

Announcement: $\boxed{\text{Final Fri Dec 17 8AM}}$
We're slowing down and triaging course scope: DFT material will be dropped this term to permit less HW problems per assignment & more review at the end of the term.

How can we find a $\vec{u}_0$ s.t. $\vec{f}(\vec{x}_0, \vec{u}_0) = \vec{0}$?

i.e. How can I solve a system of nonlinear equations?

Key Idea: Iteration: Newton's Method.

To simplify notation: Let $\vec{g}(\vec{u}) = \vec{f}(\vec{x}_0, \vec{u})$ — $\vec{x}_0$ fixed; $\vec{u}$ what I want to fix

Want to solve $\vec{g}(\vec{u}) = \vec{0}$

Use linearization. $\vec{g}(\vec{u}) \approx \vec{g}(\vec{u}_0) + \left.\frac{\partial \vec{g}}{\partial \vec{u}}\right|_{\vec{u}_0} (\vec{u} - \vec{u}_0)$

Can I solve this?? Let $A_0 = \left.\frac{\partial \vec{g}}{\partial \vec{u}}\right|_{\vec{u}_0}$

Want: $A_0(\vec{u} - \vec{u}_0) \approx -\vec{g}(\vec{u}_0)$

Call $\vec{\tilde{u}} = \vec{u} - \vec{u}_0$ $\qquad A_0 \vec{\tilde{u}} \approx -\vec{g}(\vec{u}_0)$

If A_0 invertible (square-case), can invert it.

Can solve using least-squares or min-norm approaches depending on whether A_0 is tall or wide.

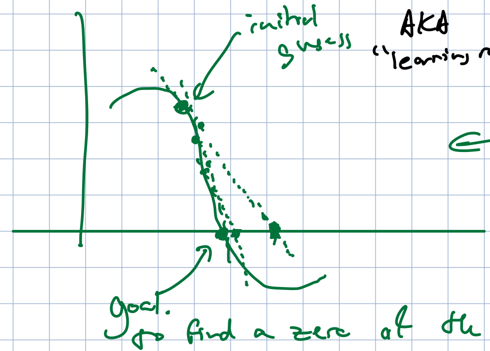
Suppose it were like least-squares. So A_0 is tall or square.

After I solve this, I'll get some $\vec{\tilde{u}}_1$.

I'll actually set $\vec{u}_1 = \vec{u}_0 + \alpha \cdot \tilde{u}_1$.

"step-size" AKA "learning rate"

Move a little bit in the direction since approximation is local.



Iterate... taking little steps

Can combine with the "transpose heuristic" ← From the Homework to iteratively solve least-squares.

$$\vec{u}_{i+1} = \vec{u}_i - \alpha A_i^T g(\vec{u}_i)$$

"Treat A^T as a cheap proxy for A^{-1}"

$$\text{where } A_i = \left.\frac{\partial \vec{g}}{\partial \vec{u}}\right|_{\vec{u}_i}.$$

Often works to get you to a solution in the neighborhood of initial guess.

If we have a way of solving systems of nonlinear equations, then we can find local minima & maxima.

Recall 16A: Module 1: Solve Linear Equations. Module 3: Do least-squares by Solving Linear Equations.

Find places where no matter which direction you move in, nothing gets better.

⇒ Gives rise to a system of equations.

So we can bring the same approach in 16B...

Aside:

What do we understand?

1) Solving systems of linear equations
2) Linear Systems
3) Least-squares & Min-norm problems

What are we faced with in real-life?

A) Solving systems of nonlinear equations
B) Nonlinear Systems
C) Non-quadratic cost functions to minimize

How to deal with this?

We do an approximate reduction.

Use "linearisation" ideas (Taylor Expansion) to <u>locally</u> convert problems into things we know how to solve.

We take the approximation error $\longrightarrow$ call it a disturbance

Make sure we're "stable".

If needed, we iterate: Turns the algorithm into a dynamic system.

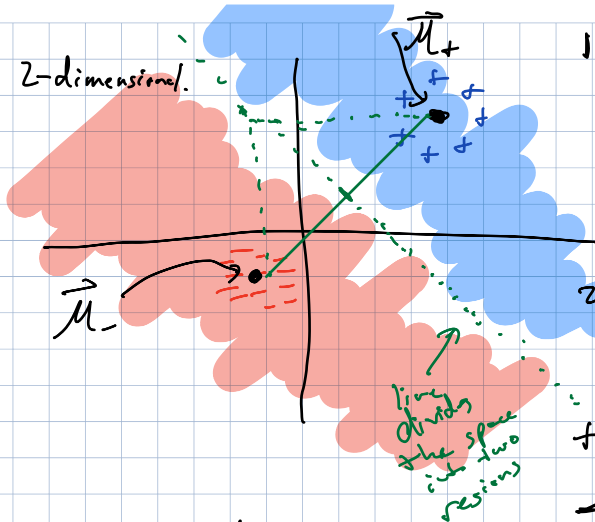
<u>Moral:</u> When you don't know what to do, just try to make a little progress. And repeat.

<u>Revisit Classification</u>: Focus on binary classification:

We have data: $(\vec{x}_i, l_i)$ where $l_i = +1$ one kind of thing or -1 A different thing

Most Intuitive Approach: (Lab approach)

Suppose $\vec{x}_i$ are 2-dimensional.



1. Compute means
$\vec{\mu}_+$ for + points
$\vec{\mu}_-$ for − points

2. Given a new point $\vec{x}$, we can classify it as

+ : if $\|\vec{x}-\vec{\mu}_+\| < \|\vec{x}-\vec{\mu}_-\|$

− : otherwise.

Question 1: What is the boundary between when we deem a point $\vec{x}$ as being + vs −??

Question 2: How do we express this decision rule so it is more clearly something related to a line?

$$\|\vec{x}-\vec{\mu}_+\|^2 < \|\vec{x}-\vec{\mu}_-\|^2$$

$$(\vec{x}-\vec{\mu}_+)^T(\vec{x}-\vec{\mu}_+) < (\vec{x}-\vec{\mu}_-)^T(\vec{x}-\vec{\mu}_-)$$

$$\cancel{\vec{x}^T\vec{x}} - 2\vec{\mu}_+^T\vec{x} + \vec{\mu}_+^T\vec{\mu}_+ < \cancel{\vec{x}^T\vec{x}} - 2\vec{\mu}_-^T\vec{x} + \vec{\mu}_-^T\vec{\mu}_-$$

$$2(\vec{\mu}_- - \vec{\mu}_+)^T\vec{x} + \|\vec{\mu}_+\|^2 - \|\vec{\mu}_-\|^2 < 0 \implies \text{classify point } \vec{x} \text{ as } +$$

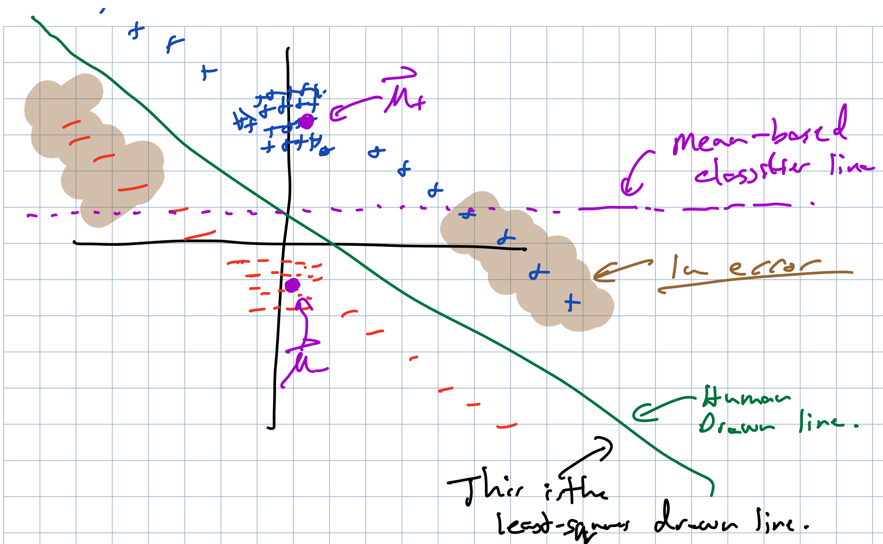
Alternatively $$(\vec{\mu}_+ - \vec{\mu}_-)^T\vec{x} + \frac{\|\vec{\mu}_-\|^2 - \|\vec{\mu}_+\|^2}{2} > 0 \implies$$

Classify as the $$\text{sign}\Big(\underbrace{(\vec{\mu}_+ - \vec{\mu}_-)^T}_{\text{some row}}\vec{x} + \underbrace{\frac{\|\vec{\mu}_-\|^2 - \|\vec{\mu}_+\|^2}{2}}_{\text{some constant.}}\Big)$$

i.e. $[w_1\ w_2]$ i.e. w_0

Question: How else could we learn something of this form?

Motivation: When does mean-based classification get things wrong??



Can we use least-squares to learn the line instead?

$$\underbrace{\begin{bmatrix}1 & \vec{x}_1^T\\ 1 & \vec{x}_2^T \\ \vdots & \vdots \\ 1 & \vec{x}_m^T\end{bmatrix}}_{A} \underbrace{\begin{bmatrix} w_0\\ w_1\\ w_2\end{bmatrix}}_{\vec{w}} \simeq \underbrace{\begin{bmatrix}l_1\\ l_2\\ \vdots \\ l_m\end{bmatrix}}_{\vec{l}}$$

Vector is filled with ± 1s.

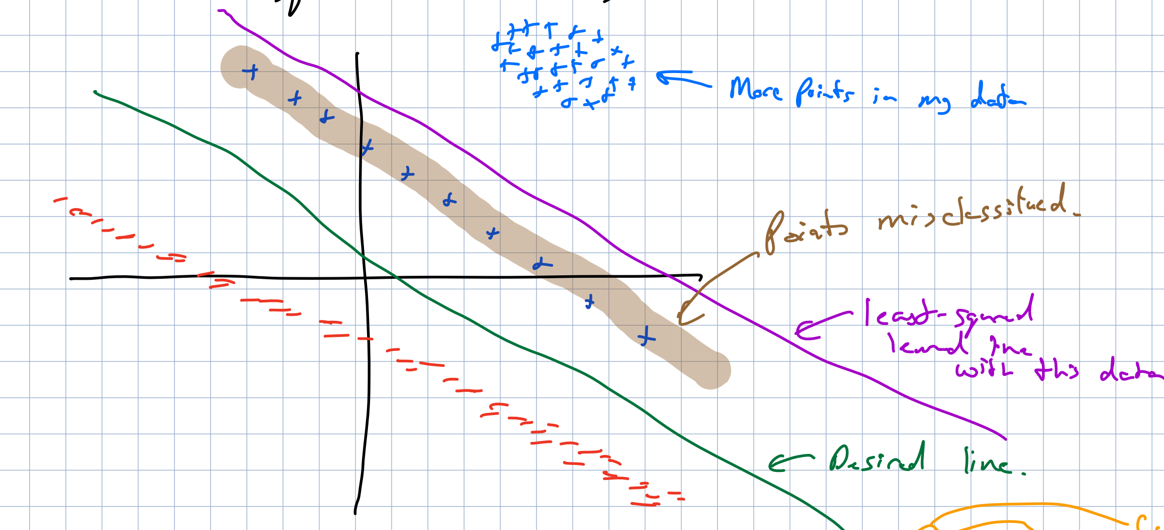
Given data points in 2d, view them in 3d instead with a constant 1 as the first entry.

Solving $A\vec{w} \simeq \vec{l}$ $\qquad \hat{\vec{w}} = (A^TA)^{-1}A^T\vec{l}$

To classify a new point $\vec{x} \in$ 2d point.

Classify as $\text{sign}\left(\hat{\vec{w}}^T \begin{bmatrix}1\\ \vec{x}\end{bmatrix}\right)$.

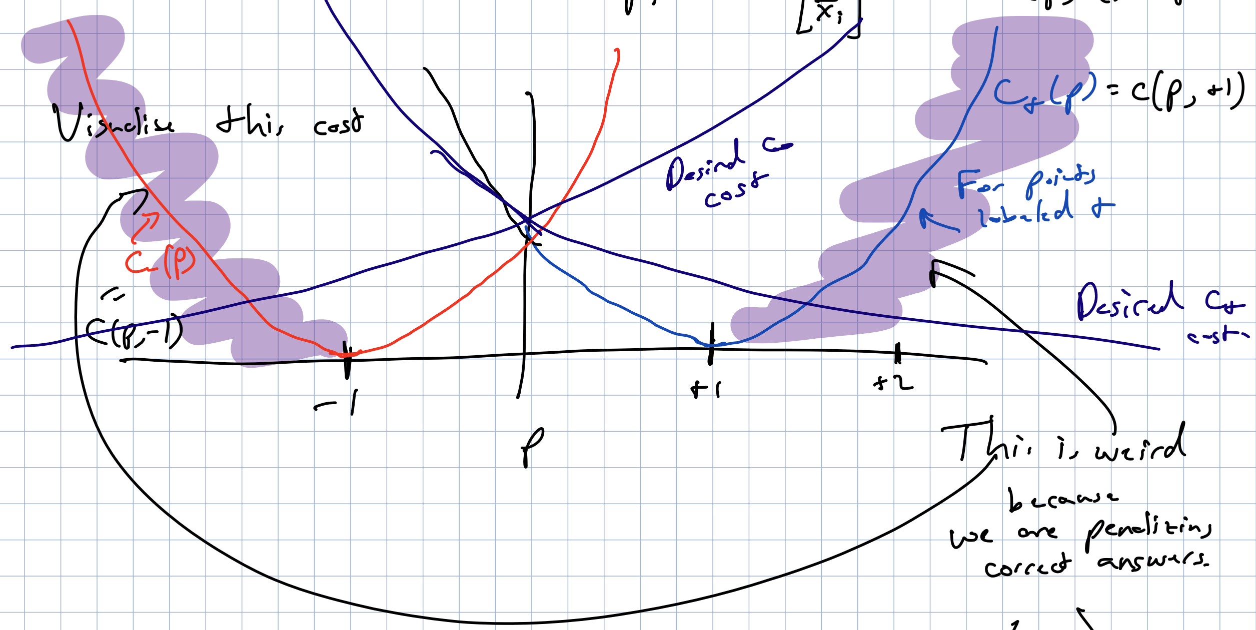
When does least-squares based learning of classification not work?



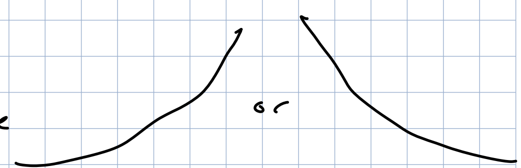
Least squares minimizes $\|A\vec{w} - \vec{l}\|^2 = \sum_{i=1}^{m} \overbrace{\left(\vec{w}^T\begin{bmatrix}1\\ \vec{x}_i\end{bmatrix} - l_i\right)^2}^{\text{Call this the cost associated with } i^{th} \text{ pt.}}$

What went wrong?! Least-squares is minimizing $\sum_{i=1}^{m} c(p_i, l_i)$

where $p_i = \vec{w}^T \begin{bmatrix} 1 \\ \vec{x}_i \end{bmatrix}$ and $c(p, l) = (p-l)^2$



What kind of cost looks like



e.g. exponential. Can we use a cost $c_+(p) = e^{-p}$ & $c_-(p) = e^{+p}$?

$$c(p, l) = e^{-lp}$$

We want $\underset{\vec{w}}{\arg\min} \sum_{i=1}^{m} \exp\left(-l_i \vec{w}^T \begin{bmatrix} 1 \\ \vec{x}_i \end{bmatrix}\right)$

From now on assume $\vec{x}_i$ has 0^{th} component as 1

$$\underset{\vec{w}}{\arg\min} \underbrace{\sum_{i=1}^{m} \underbrace{\exp(-l_i \vec{x}_i^T \vec{w})}_{c_i}}_{C(\vec{w})}$$

How do I optimize this??

3 approaches:

Ⓐ Pretend this is locally quadratic and massage it into least-squares form to solve.

Ⓑ Look for a local min/max by seeing where the system of nonlinear equations $\frac{\partial}{\partial \vec{w}} C(\vec{w}) = \vec{0}^T$ is solved.

(Use Newton's Method)

It turns out that these two give the same algorithm.

Ⓒ Pretend it is linear and just iterate
$\Longrightarrow$ go down the slope a little.

SUPER IMPORTANT

Computers can take derivatives automatically,

e.g. pytorch. (also tensorflow, JAX, etc...)

Will leverage this in discussion.