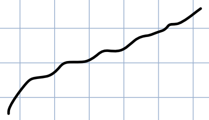
16B Prof. Anant Sahai (OH: Thu 2-3pm)

Today: Orthonormalization



Midterm last night, but now redo it like HW.
Can collaborate. Go to HW Party. Due Friday.
Read solutions after redo due, and self-grade redo.
Can resubmit after seeing solutions to get back redo pts.
If redo complete (i.e. hit 80% after resubmit), unlocks clobber.

Our Goal: Your Mastery

Motivating Example:

Related to HW 6 DemoSystemId problem Example 4.

$$\underset{\text{Scalar}}{u[i]} \longrightarrow \boxed{\text{System}} \longrightarrow y[i]$$

Want to model

The model could be: $y[i+1] = b\,u[i]$

Called auto-regressive models or Markov of order k.

or $y[i+1] = b\,u[i] + a_0\,y[i]$

or $y[i+1] = b\,u[i] + a_0\,y[i] + a_1\,y[i-1]$

or $y[i+1] = b\,u[i] + a_0\,y[i] + a_1\,y[i-1] + a_2\,y[i-2]$

or
$\vdots$ $y[i+1] = b\,u[i] + \sum_{j=0}^{k-1} a_j\,y[i-j]$

Want to do System-ID.

a) Try to fit all the models

b) Test them to see which one works best.

Gold Standard: Works in practice

Silver Standard: predicts well on test data.
"hold-out set"

To do ⓐ, need to solve a family of nested least-squares problems

Approach: $\underset{\text{data}}{D}\,\underset{\text{unknown parameters}}{P} \simeq \underset{\text{data.}}{S}$ ← Set up approximate equations

$$S = \begin{bmatrix} \vdots \\ y[7] \\ y[8] \\ \vdots \end{bmatrix} \qquad P = \begin{bmatrix} b \end{bmatrix} \text{ or } \begin{bmatrix} b \\ a_0 \end{bmatrix} \text{ or } \begin{bmatrix} b \\ a_0 \\ a_1 \end{bmatrix} \cdots\cdots$$

$$D = \underbrace{\begin{bmatrix} \vec{d}_1 & \vec{d}_2 & \cdots & \vec{d}_{k+1} \\ & & & \end{bmatrix}}_{\#\text{ matches dim } p}$$

LS Problem 0:

$$\begin{bmatrix} \vec{d}_1 \end{bmatrix} \vec{p}_1 \approx \vec{s}$$

LS Problem 1

$$\begin{bmatrix} \vec{d}_1 & \vec{d}_2 \end{bmatrix} \vec{p}_2 \approx \vec{s}$$

LS Problem 2

$$\begin{bmatrix} \vec{d}_1 & \vec{d}_2 & \vec{d}_3 \end{bmatrix} \vec{p}_3 \approx \vec{s}$$

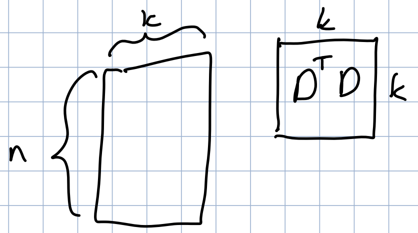
Other places this can happen:

1) Polynomial fitting.

2) <u>OMP</u>

How do you solve least-squares?

$$D\vec{p} \approx \vec{s} \quad \longrightarrow \quad \hat{\vec{p}} = \underset{k \times k}{(D^T D)}^{-1} D^T \vec{s}$$



Keeps growing.

Inverting takes $O(k^3)$ steps.

<u>Gets painful.</u> –

Can we make things faster?

To do ℓ nested least-sq problems

$$1 + 2^3 + 3^3 + \cdots + \ell^3 = \frac{\ell^2(\ell+1)^2}{4} = O(\ell^4)$$

Can we solve a whole nested family of LS problems faster.

<u>Thoughts</u>: (1) $(D^T D)^{-1}$ is the painful step. When is this fast?

(2) Can we change coordinates somehow?

$$D^T D = \begin{bmatrix} \vec{d}_1^{\,T} \\ \vec{d}_2^{\,T} \\ \vdots \\ \vec{d}_k^{\,T} \end{bmatrix} \begin{bmatrix} \vec{d}_1 & \vec{d}_2 & \cdots & \vec{d}_k \end{bmatrix} = \underbrace{\begin{bmatrix} \vec{d}_1^{\,T} \vec{d}_1 & \vec{d}_1^{\,T} \vec{d}_2 & \cdots & \vec{d}_1^{\,T} d_k \\ \vdots & & \ddots & \\ & & & \vec{d}_k^{\,T} \vec{d}_k \end{bmatrix}}_{\text{Matrix of pairwise inner products}}$$

$\vec{d}_i \in \mathbb{R}^n$ ← n-dim vectors

$\langle \vec{d}_i, \vec{d}_j \rangle = \vec{d}_j^{\,T} \vec{d}_i$

When is $D^T D$ the identity?

$\vec{d}_i^{\,T} \vec{d}_i = 1$ ← diagonal entries of

$\vec{d}_i^{\,T} \vec{d}_j = 0$ if $i \neq j$ aka. $\langle \vec{d}_j, \vec{d}_i \rangle = 0$

i.e. orthogonal or perpendicular $\perp$

$\|\vec{d}_i\|^2 = 1$

$\|\vec{d}_i\| = 1$ ← normalized $= \sqrt{(d_i[1])^2 + (d_i[2])^2 + \cdots (d_i[n])^2}$

Desired property: Orthonormal columns.

$Q = [\vec{q}_1 \ \vec{q}_2 \ \cdots \ \vec{q}_k]$ Know if Q is orthonormal matrix then $Q^T Q = I$ ← $k \times k$

↑ favored letter for a matrix with orthonormal columns.

"orthonormal matrix"

How can we project onto the span of the columns of Q

Want to find $\vec{x}$ s.t. $\underbrace{Q\vec{x}}_{\hat{y}}$ is as close as possible to $\vec{y}$

$\hat{y}$ projection of $\vec{y}$ onto span(Q)

$$\hat{\vec{x}} = \cancel{(Q^T Q)^{-1}}^{\,I} Q^T \vec{y} = Q^T \vec{y}$$

$$\hat{\vec{y}} = Q \hat{\vec{x}} = \underbrace{Q Q^T}_{n \times n \text{ Projection operator for } Q.} \vec{y}$$

e.g

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1/\sqrt{2} \\ 0 & 1/\sqrt{2} \end{bmatrix} \qquad Q^T Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad Q Q^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 1/2 \\ 0 & 1/2 & 1/2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ \frac{1}{2}(y_2+y_3) \\ 1/2\,(y_2+y_3) \end{bmatrix}$$

Given a basis expressed as an orthonormal Q
for a subspace

Projection onto it is very fast.

Goal (Subgoal of making things fast):

Given a sequence of vectors $\vec{d}_1, \vec{d}_2, \dots, \vec{d}_\ell$

Return another sequence of vectors $\vec{q}_1, \vec{q}_2, \dots, \vec{q}_\ell$

s.t. $Q = [\vec{q}_1, \dots, \vec{q}_\ell]$ is orthonormal

and $\forall k \leq \ell \quad \text{span}\{\vec{d}_1, \dots, \vec{d}_k\} = \text{span}\{\vec{q}_1, \dots, \vec{q}_k\}$.

Orthonormalization is the name given to this process.

Start with simplest case. $\{\vec{d}_1\}$ ← just one vector.

Need $\vec{q}_1 = \alpha \vec{d}_1$ ← so it spans the same subspace.

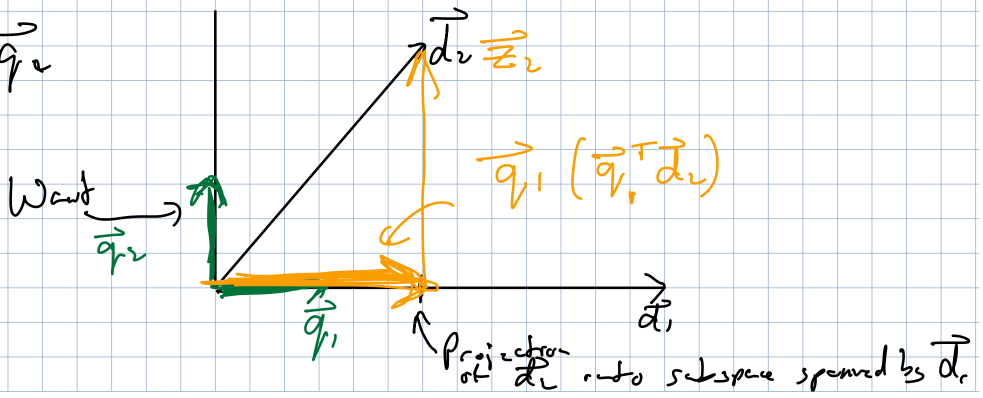
Q: How to pick α?

Guess: Divide by norm?

$\vec{q}_1 = \frac{\vec{d}_1}{\|\vec{d}_1\|}$ Compute $\vec{q}_1^T \vec{q}_1 = \frac{\vec{d}_1^T \vec{d}_1}{\|\vec{d}_1\| \cdot \|\vec{d}_1\|} = \frac{\|\vec{d}_1\|^2}{\|\vec{d}_1\|^2} = 1$

First nontrivial case: $\{\vec{d}_1, \vec{d}_2\}$

Want $\vec{q}_2$


$\vec{d}_2$ $\vec{z}_2$
$\vec{q}_1(\vec{q}_1^T \vec{d}_2)$
Want
$\vec{q}_2$
$\vec{q}_1$
$\vec{d}_1$
Projection of $\vec{d}_2$ onto subspace spanned by $\vec{d}_1$

$\vec{z}_2 = \vec{d}_2 - \vec{q}_1(\vec{q}_1^{\,T} \vec{d}_2)$

↖ Parallel to desired $\vec{q}_2$

So let $\vec{q}_2 = \vec{z}_2 / \|\vec{z}_2\|$

Steps

1) Project $\vec{d}_2$ onto $\vec{d}_1$

2) Subtract this projection to get a residual that is $\perp$

3) Normalize it.

Breaks if $\|\vec{z}_2\| = 0$ or $\vec{z}_2 = \vec{0}$

Happens only if $\vec{d}_2$ is a multiple of $\vec{d}_1$.

Generalize Given $\{\vec{d}_1, \dots, \vec{d}_k, \vec{d}_{k+1}\}$

How can we get $\vec{q}_{k+1}$ given that we already have $Q_k = [\vec{q}_1, \dots, \vec{q}_k]$ with colspan(Q_k) = colspan(D_k)

Let $D_k = [\vec{d}_1, \dots, \vec{d}_k]$

Want $\vec{z}_{k+1} = \vec{d}_{k+1} -$ Projection of $\vec{d}_{k+1}$ onto subspace spanned by columns of D_k

$$= \vec{d}_{k+1} - \underbrace{D_k (D_k^T D_k)^{-1} D_k^T}_{\text{Projection Operator for colspan } D_k} \vec{d}_{k+1}$$

Since Projection onto colspan(D_k) is the same as projection onto colspan(Q_k)

$$= \vec{d}_{k+1} - Q_k Q_k^T \vec{d}_{k+1}$$

$$= \vec{d}_{k+1} - \sum_{j=1}^{k} \vec{q}_j (\vec{q}_j^{\,T} d_{k+1})$$

Can normalize to get $\vec{q}_{k+1} = \vec{z}_{k+1} / \|\vec{z}_{k+1}\|$

Is $Q_{k+1} = [Q_k \,;\, \vec{q}_{k+1}]$ orthonormal?

Need to check: $\|\vec{q}_{k+1}\| = 1$? Yes because

Is $\vec{q}_{k+1} \perp q_i$ for $i = 1, \dots, k$

Check: $\vec{q}_i^{\,T} \vec{q}_{k+1} = \dfrac{\vec{q}_i^{\,T} \vec{z}_{k+1}}{\|\vec{z}_{k+1}\|} = \dfrac{1}{\|\vec{z}_{k+1}\|} \left(\vec{q}_i^{\,T} \vec{d}_{k+1} - \sum_{j=1}^{k} \vec{q}_i^{\,T} \vec{q}_j (\vec{q}_k^{\,T} d_{k+1}) \right)$

$$= \frac{1}{\|\vec{z}_{k+1}\|} \left(\boxed{\vec{q}_i^{\,T} \vec{d}_{k+1}} - 0 - 0 - 0 - 1 \cdot \boxed{\vec{q}_i^{\,T} \vec{d}_{k+1}} - 0 - 0 - \dots 0 \right)$$

$= 0$ So $\vec{q}_{k+1} \perp \vec{q}_i$ for $i = 1 \dots k$.

☺

<u>Subgoal Achieved</u>: <u>Gram-Schmidt Orthonormalization</u>

Is this actually any faster??!

<u>New Way</u>

How many operations to get $\vec{q}_k$?

$\boxed{n} + \boxed{kn}$ ← to get $\vec{z}_k$

↑ For normalization

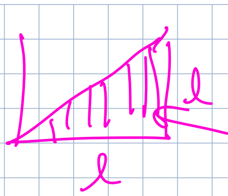
For $k = 1, \dots, \ell$

So total cost of getting $\vec{q}_1, \dots \vec{q}_\ell$ is $O(\ell^2 n)$

$$= \vec{d}_{k+1} - \sum_{j=1}^{k} \vec{q}_j (\vec{q}_j^{\,T} \vec{d}_{k+1})$$

to get $\vec{q}_{k+1} = \vec{z}_{k+1} / \|\vec{z}_{k+1}\|$

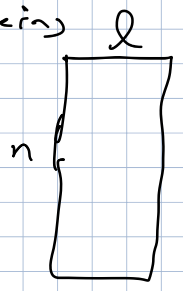
Recall $\vec{d}_i$ are n-dim as are $\vec{q}_i$

Because 

Area is $\frac{1}{2}l^2$

What about the old way? $(D^TD)^{-1}D^T\vec{y}$

For D being



$$l^3 + l^2 n + l n$$

From $(D^TD)^{-1}$ to compute D^TD

$n > l$. So Both ways for the l-dim case alone cost the same.

Savings exist relative to the l^4 term in computing all the LS. problem

Get significant speedup.

$(100)^4 = 10^8 \leftarrow$ 100 million

$(100)^3 = 10^6 \leftarrow$ 1 million.

Final Approach: To solve nested least squares problems:

1) Orthonormalize the columns $[\vec{d}_1, \dots, \vec{d}_l]$ to get $Q = [\vec{q}_1, \dots, \vec{q}_l]$

2) Compute $Q^T\vec{y}$ to get solutions to all nested problems at once.

So in the new coordinates: to see the solution for just k cols,
look at $\vec{q}_1^T \vec{y}, \ldots, \vec{q}_k^T \vec{y}$.

This gives you a projection $\sum_{i=1}^{k} \vec{q}_i \left(\vec{q}_i^T \vec{y}\right)$
that best approximates $\vec{y}$ in the subspace spanned by
$\vec{d}_1, \ldots, \vec{d}_k$.