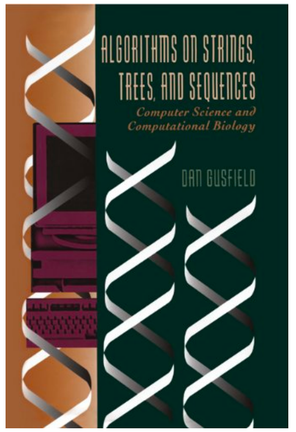
Parametric Sequence Alignment



Based on Chapter 13 from *Algorithms on Strings, Trees, and Sequences* by Dan Gusfield

pairwise sequence alignment

Given

- a pair of sequences S_1,S_2 with lengths m and n, and
- an alignment objective function

find an 2 x L matrix

- where max(m,n) < L < m+n,
- each row represents one sequence from the set with inserted gaps, and
- is optimal under the objective function.

Edit Distance

A simplified version of the the alignment problem, where the objective function is simply the count of operations:

- *replace* one character with another -- `R`
- *delete* a character from the first string -- `D`
- *insert* a character from the second string -- `I`

Example: S_1 = **`baseball`** & S_2 = **`ballcap`**.

```
   RRR DR
baseball
ballca p
```

- 5 operations: change `s→l`, `e→l`, `b→c`, delete `l`, `l→p`

Alignment

In general, associate a similarity score with each pair of aligned characters:

- for characters $x,y \in \Sigma \cup \{-\}$, let $\lambda(x,y)$ be the similarity of x and y

Let the score, Λ, of an alignment, $A = (S'_1, S'_2)$, be defined as

$$\Lambda(A) =: \sum_{1 \le i \le |S'_1|} \lambda(S'_1[i], S'_s[i])$$

Goal of alignment is to maximize that sum

	-	a	b	c	e	l	p	s
-		-1	-1	-1	-1	-1	-1	-1
a	-1	0	-1	-1	-1	-1	-1	-1
b	-1	-1	0	-1	-1	-1	-1	-1
c	-1	-1	-1	0	-1	-1	-1	-1
e	-1	-1	-1	-1	0	-1	-1	-1
l	-1	-1	-1	-1	-1	0	-1	-1
p	-1	-1	-1	-1	-1	-1	0	-1
s	-1	-1	-1	-1	-1	-1	-1	0

```
baseball
ballca-p
```

Alignment

In general, associate a similarity score with each pair of aligned characters:

- for characters $x,y \in \Sigma \cup \{-\}$, let $\lambda(x,y)$ be the similarity of x and y

Let the score, Λ, of an alignment, $A = (S'_1, S'_2)$, be defined as

$$\Lambda(A) =: \sum_{1 \le i \le |S'_1|} \lambda(S'_1[i], S'_s[i])$$

Goal of alignment is to maximize that sum

How can we compute an alignment that optimizes Λ?

	-	a	b	c	e	l	p	s
-		0	0	0	0	0	0	0
a	0	2	-1	-1	-1	-1	-1	-1
b	0	-1	2	-1	-1	-1	-1	-1
c	0	-1	-1	2	-1	-1	-1	-1
e	0	-1	-1	-1	2	-1	-1	-1
l	0	-1	-1	-1	-1	2	-1	-1
p	0	-1	-1	-1	-1	-1	2	-1
s	0	-1	-1	-1	-1	-1	-1	2

```
baseball---
----ballcap
```

Needleman-Wunsch

brute-force: compute all possible alignments and score them

- would take exponential time to compute the optimal alignment

using dynamic programming Needleman and Wunsch [1970] found that the optimal alignment can be computed in $O(mn)$-time.

Lets talk about gaps!

Needleman-Wunsch considers all gap character as constant cost operations

Consider two possible alignments:

- one has one group of 12 gap characters next to each other
- the other has two groups of 6 gap characters.

In real life (i.e. in biology) any group of gap characters causes a disruption

- the length (within reason) is secondary.

So we want to score alignments with this in mind.

Generally, for a gap (set of contiguous insertions or deletions) of length *k,* we will assign a score that is some function *f(k)* of the length

Affine Gap Costs

Using an abitrary gap cost function increases the running time to *O(mn(m+n)).*

Define the function $g_{a,b}(k) =: a + b * k$

- where *a* and *b* are tunable parameters.
- When *a=0*, this equivalent to the original definition.

Can still be solved in *O(mn)*-time and *O(mn)*-space

- The algorithm is attributed to Gotoh [1982]

Alignment Scores

In all of the examples we have been given λ, a, and b

- using fixed values we are able to find the optimal alignment in $O(mn)$-time

Since different values of those parameter induce different alignments, how do we know which parameter values are best?

What if we look at all possible (optimal) alignments, and pick one?

- This is a problem known as *inverse parametric alignment* (or just parametric alignment).

How many *optimal* alignments do you think there are? (notice that optimal is emphasized)

Alignment objective function

$$f_{\alpha,\beta,\gamma,\delta}(\mathbb{A}) = \alpha \cdot \mathbf{mt}_{\mathbb{A}} - \beta \cdot \mathbf{ms}_{\mathbb{A}} - \gamma \cdot \mathbf{id}_{\mathbb{A}} - \delta \cdot \mathbf{gp}_{\mathbb{A}}$$

- $\mathrm{mt}_{\mathbb{A}}$ -- number of columns where both characters match
- $\mathrm{ms}_{\mathbb{A}}$ -- number of columns where there characters are different (mismatches)
- $\mathrm{id}_{\mathbb{A}}$ -- number of gap characters (indels)
- $\mathrm{gp}_{\mathbb{A}}$ -- number of gaps

Equivalent to a λ that only has two distinct values: one for matches and one for mismatches.

An example

s_1 = AACCCG
s_1 = AAGGCC

$\mathbb{A}_1$

```
AA--CCCG
AAGGCC--
```

	$\mathbb{A}_1$
mt	4
ms	0
id	4
gp	2

Question: what values of α,β,ɣ, and δ should we choose to get the “best” alignment?

An example

s_1 = AACCCG
s_1 = AAGGCC

$\mathbb{A}_1$
AA--CCCG
AAGGCC--

$\mathbb{A}_2$
AA-CCCG
AAGGCC-

$\mathbb{A}_3$
AACCCG
AAGGCC

$\mathbb{A}_4$
AAC-CCG
AAGGCC-

	$\mathbb{A}_1$	$\mathbb{A}_2$	$\mathbb{A}_3$	$\mathbb{A}_4$
mt	4	4	3	4
ms	0	1	3	1
id	4	2	0	2
gp	2	2	0	2

Question: what values of α,β,ɣ, and δ should we choose to get the “best” alignment?

An example

s_1 = AACCCG
s_1 = AAGGCC

$\mathbb{A}_1$ **AA--CCCG**
AAGGCC--

$\mathbb{A}_2$ **AA-CCCG**
AAGGCC-

$\mathbb{A}_3$ **AACCCG**
AAGGCC

$\mathbb{A}_4$ **AAC-CCG**
AAGGCC-

	$\mathbb{A}_1$	$\mathbb{A}_2$	$\mathbb{A}_3$	$\mathbb{A}_4$
mt	4	4	3	4
ms	0	1	3	1
id	4	2	0	2
gp	2	2	0	2

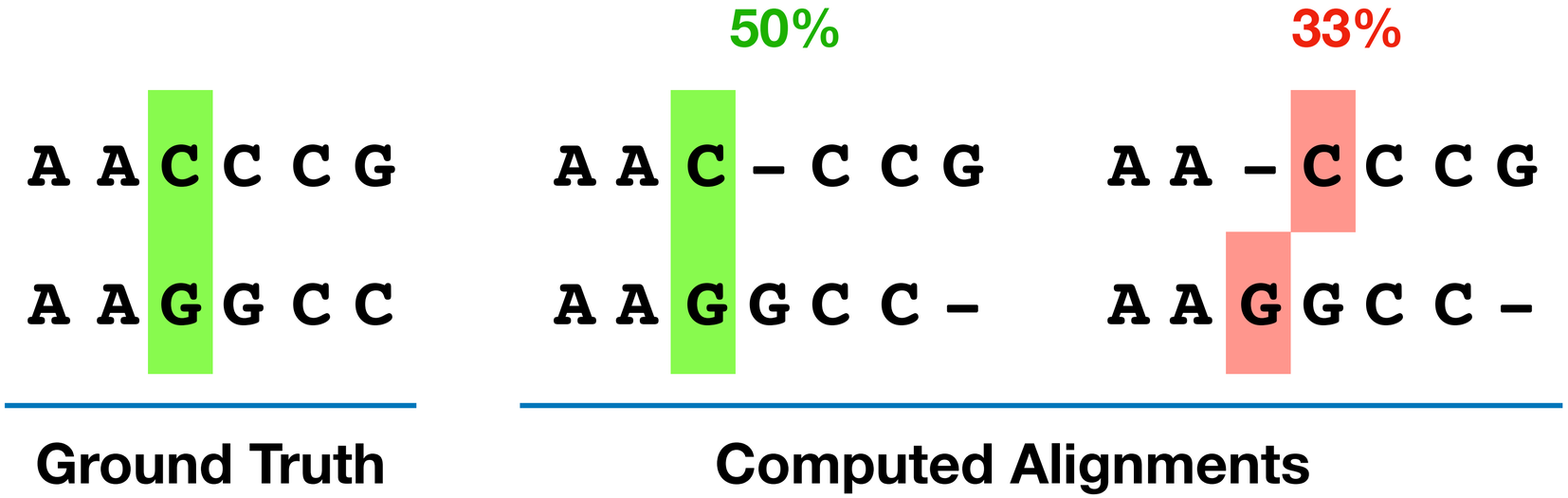
What do we even mean by "best"?

Question: what values of α,β,ɣ, and δ should we choose to get the “best” alignment?

A Digression on Accuracy

How would we know how accurate an alignment was if we knew the right answer?

The **sum-of-pairs** accuracy measures the fraction of substitutions from the ground truth alignment that are recovered in a computed alignment



An example

s_1 = AACCCG

s_1 = AAGGCC

$\mathbb{A}_1$ **AA--CCCG**
AAGGCC--

$\mathbb{A}_2$ **AA-CCCG**
AAGGCC-

$\mathbb{A}_3$ **AACCCG**
AAGGCC

$\mathbb{A}_4$ **AAC-CCG**
AAGGCC-

	$\mathbb{A}_1$	$\mathbb{A}_2$	$\mathbb{A}_3$	$\mathbb{A}_4$
mt	4	4	3	4
ms	0	1	3	1
id	4	2	0	2
gp	2	2	0	2

Can we enumerate all possible alignments?

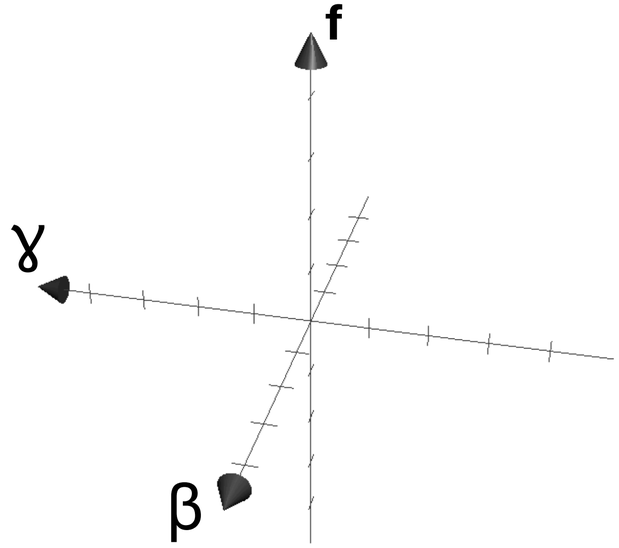
How many are there?

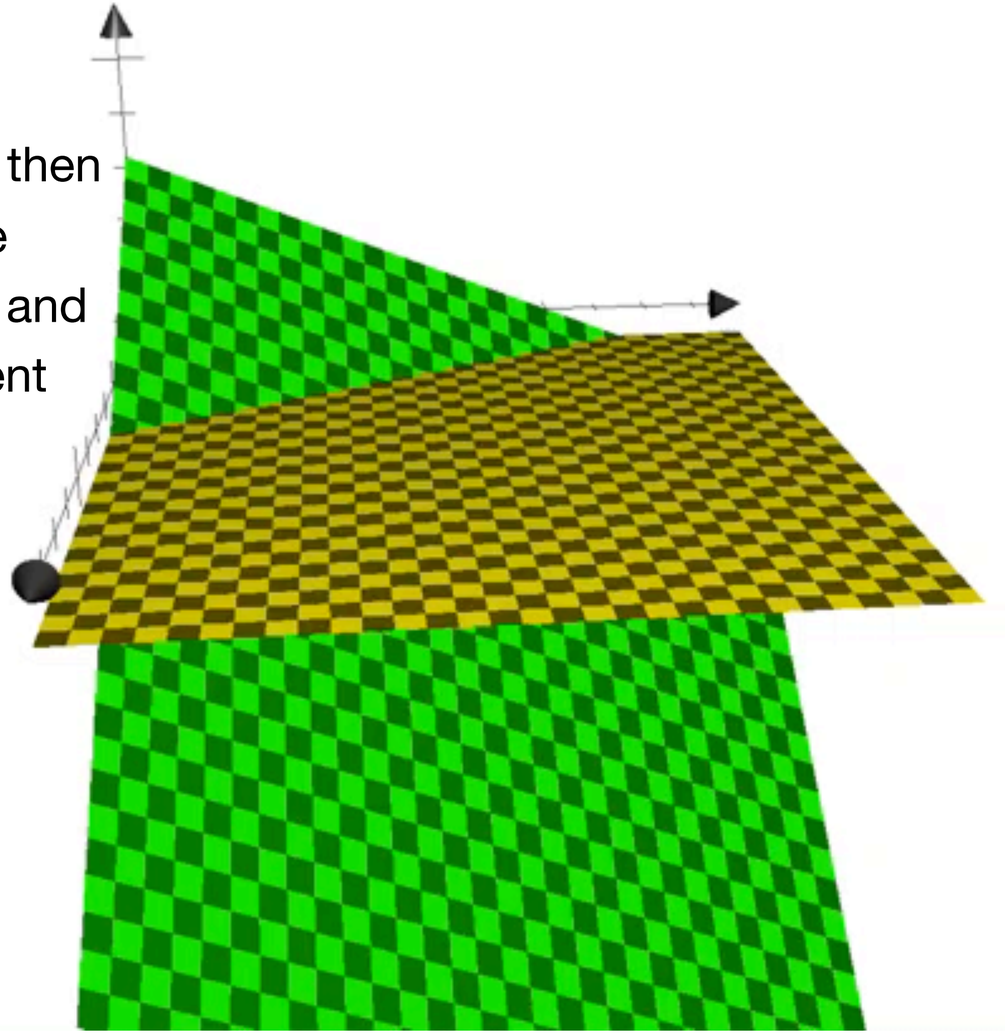
Question: what values of ~~α,β~~,γ, and δ should we choose to get the “best” alignment?

Pairs of alignments in parameter space

Each alignment can be represented as a plane in the (ɣ, δ, f)-space.

If the planes of alignments 𝔸 & 𝔸’ intersect, and are distinct, then there is a line L in (ɣ, δ, f)-space along which 𝔸 & 𝔸’ have the same objective value; 𝔸 has a larger value on one half plane and 𝔸’ on he other. If the planes don’t intersect then one alignment had a larger objective value at all assignments of ɣ & δ.



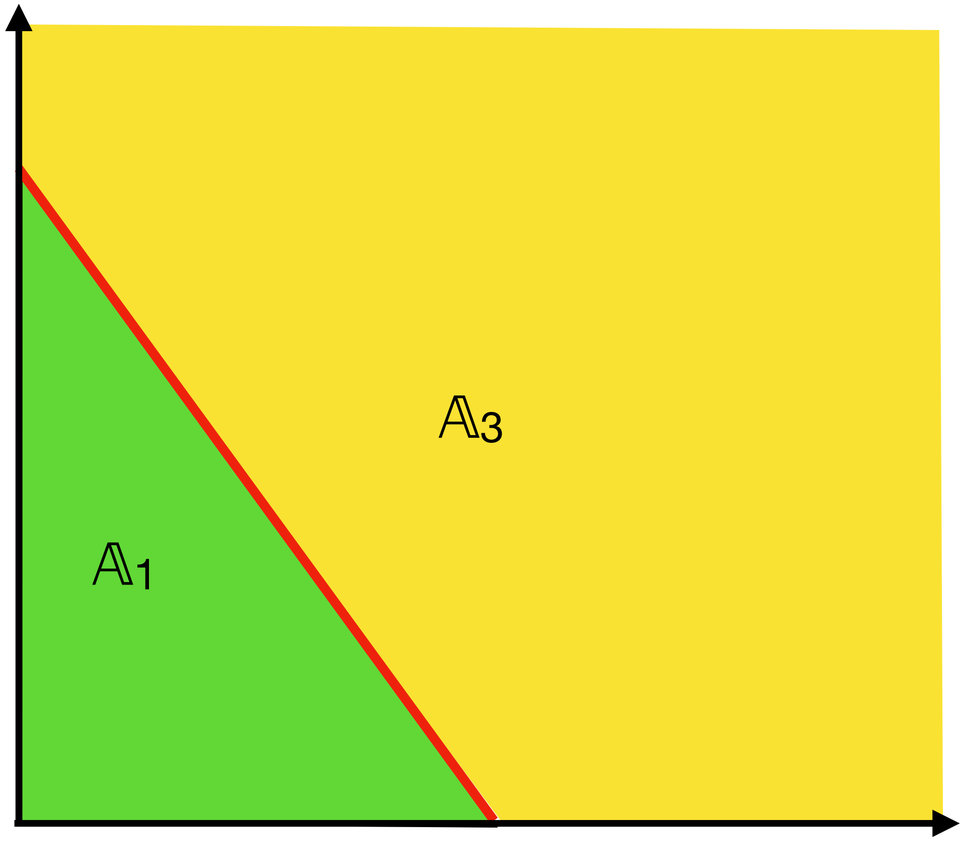


Pairs of alignments in parameter space

Each alignment can be represented as a plane in the (ɣ, δ, f)-space.

If the planes of alignments $\mathbb{A}$ & $\mathbb{A}$’ intersect, and are distinct, then there is a line L in **(ɣ, δ)-space** along which $\mathbb{A}$ & $\mathbb{A}$’ have the same objective value; $\mathbb{A}$ has a larger value on one half plane and $\mathbb{A}$’ on he other. If the planes don’t intersect then one alignment had a larger objective value at all assignments of ɣ & δ.

When projected to the (ɣ, δ)-plane, we can designate regions for which f($\mathbb{A}$)>f($\mathbb{A}$’) and vice versa

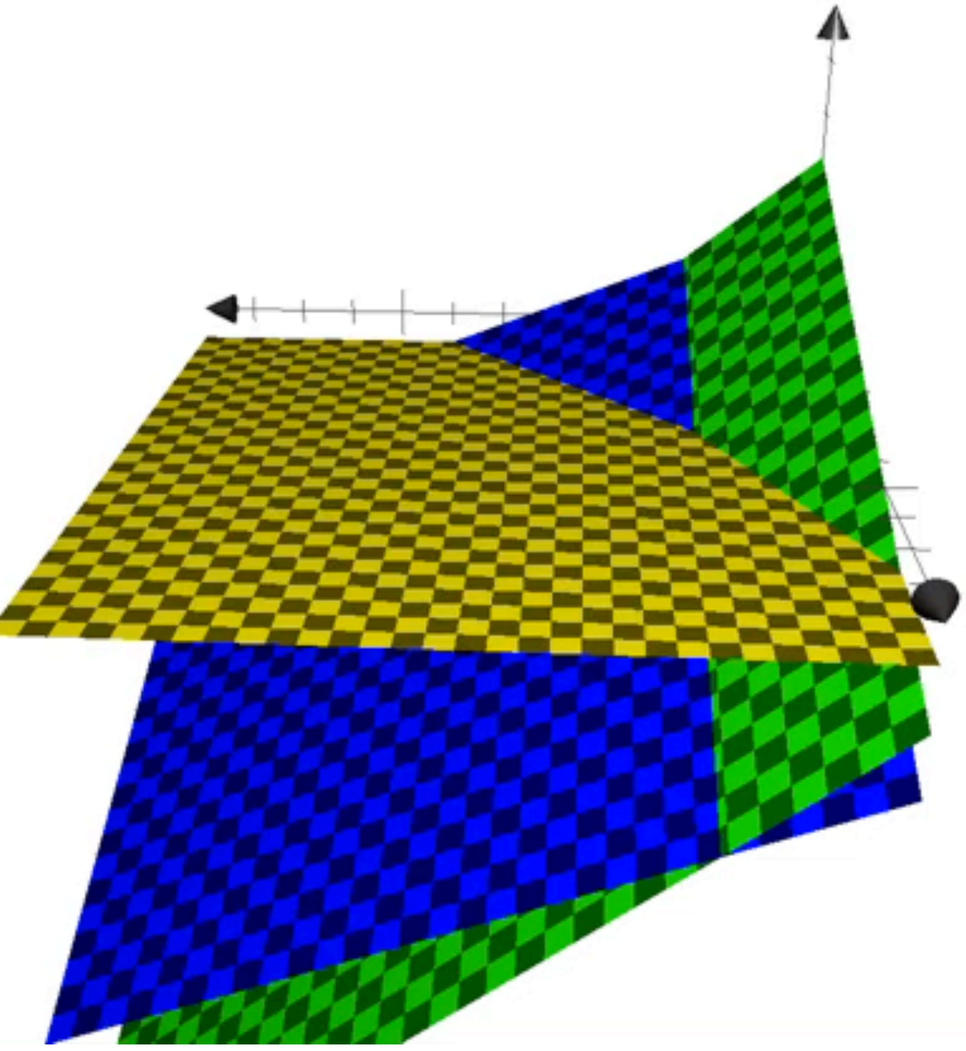


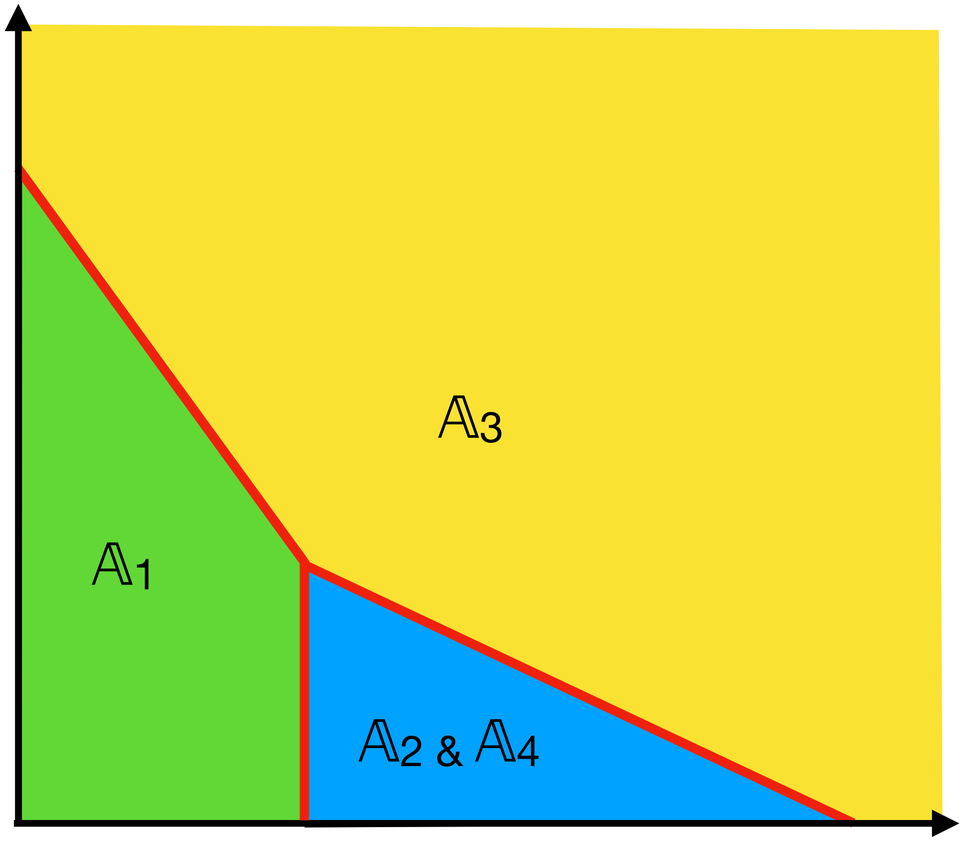
Things we know so far

For a parameter setting, we can find the optimal alignment.

Two alignments will have a line in (ɣ, δ)-space where they are co-optimal*.

Alignments in parameter space





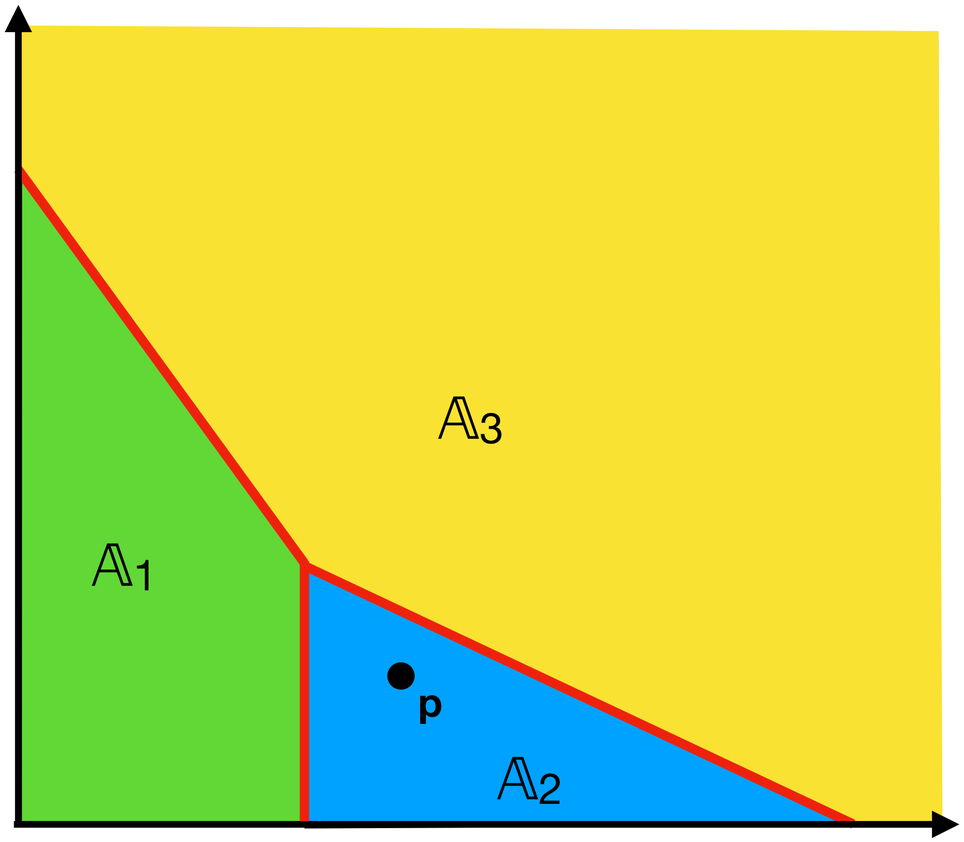
Alignments in parameter space

If $\mathbb{A}$ is optimal at some point p, it is on the correct side of the line that separates $\mathbb{A}$ from all $\mathbb{A}$'.

If $\mathbb{A}$ is optimal for at least 1 point p in the (ɣ, δ)-space then it is optimal for:

(1) only point p,
(2) only a line segment that contains p, or
(3) a convex polygon that contains p

Given two string s_1 and s_2 the (ɣ, δ)-space decomposes into convex polygons such that any point in the interior of the polygon P is optimal for all points in P



slides: deblasiolab.org/parametric-sequence-alignment/

Things we know so far

For a parameter setting, we can find the optimal alignment.

Two alignments will have a line in (ɣ, δ)-space where they are co-optimal*.

For any point, the optimal alignment is optimal for a point, a line, or a region.

There are a limited number of regions for a fixed input.

How many regions are there?
Can we find them?

Newton's ray search algorithm

Given a random point p, choose a ray h that extends to the boundary.

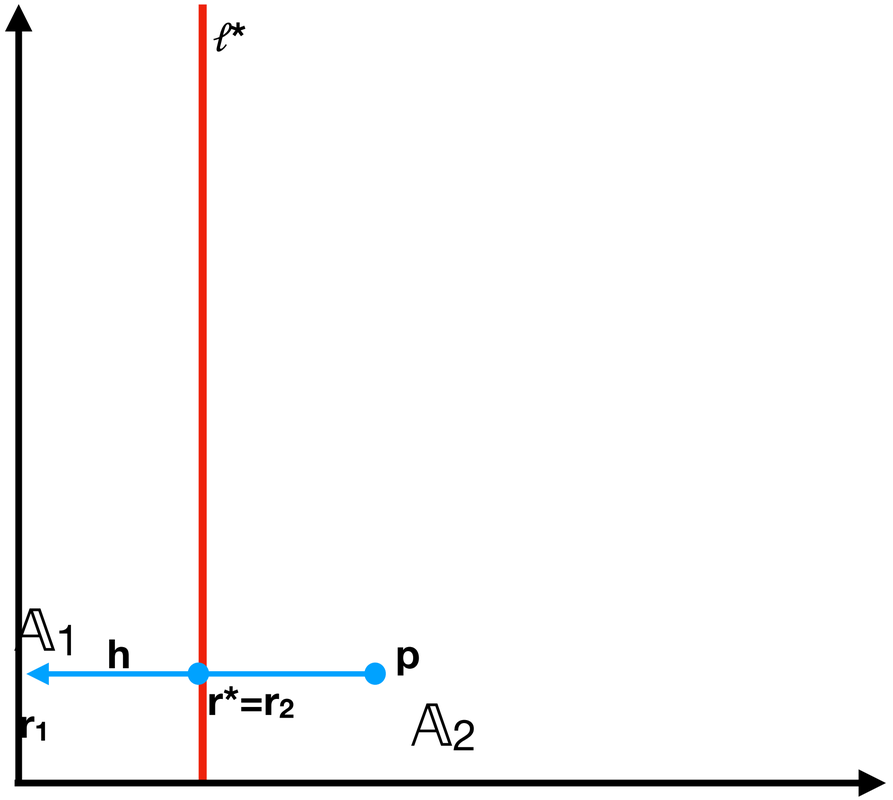
Let r_1 be the point at that boundary.

Find alignment $\mathbb{A}'$ that is optimal at r_1.

If $\mathbb{A}!=\mathbb{A}'$ let the point r_{i+1} be the parameter choice that is on the line that divides $\mathbb{A}$ and $\mathbb{A}'$, and is also on h.

Repeat until $\mathbb{A}$ and $\mathbb{A}'$ are co-optimal (i.e. stop when $\mathbb{A}$ is optimal) at r_{i+1}, and set $r^*= r_{i+1}$.

The boundary for the polygon in which p resides is a segment of that line.



Newton's ray search algorithm

- Newton's ray search algorithm finds r* exactly
- Unless $\mathbb{A}$ is optimal at the initial setting of r, the last computed alignment $\mathbb{A}^*$ is cooptimal with $\mathbb{A}$ at r* and it is also optimal on h for some non-zero distance beyond r*.
- When Newton's ray search algorithm computes an alignment at a point r on h, none of the alignments computed previously (in this execution of Newton's algorithm) are optimal at r.

note: it follows that any polygon P intersected by h, a single ray search computes alignments at no more than 2 points of P.

Things we know so far

For a parameter setting, we can find the optimal alignment.

Two alignments will have a line in (ɣ, δ)-space where they are co-optimal*.

For any point, the optimal alignment is optimal for a point, a line, or a region.

There are a limited number of regions for a fixed input.

Given a point and a ray, we can find a point (and a line) that is at the boundary for the polygon p is in (if it is inside a polygon).

How many regions are there?
Can we find them?

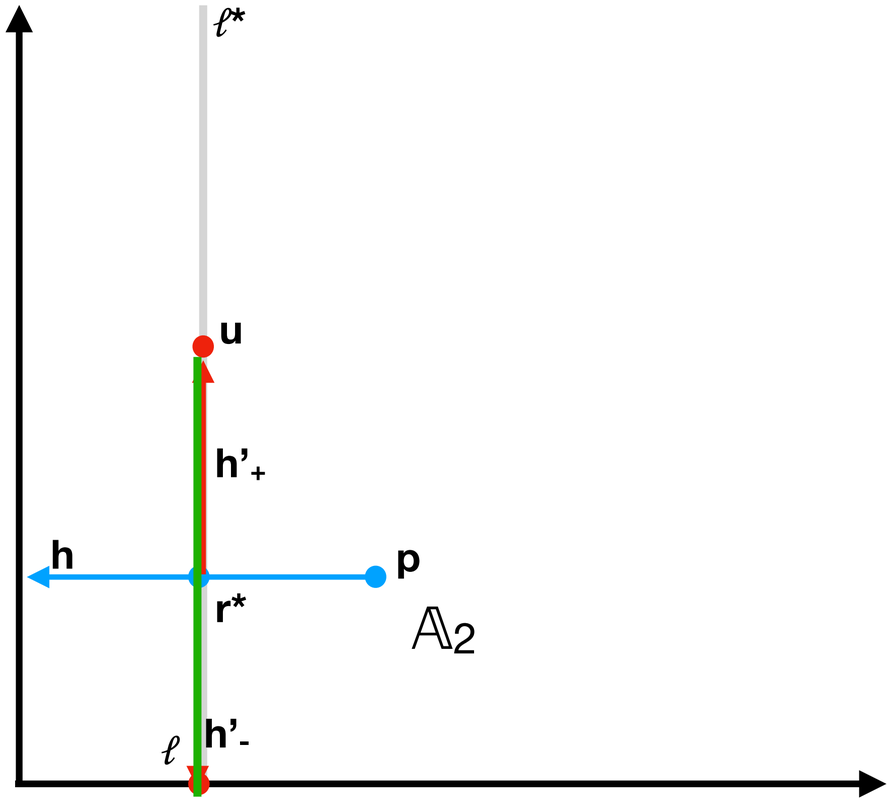
Finding an edge of the polygon

Given p, h, r*, and the line that separates $\mathbb{A}$ from the next optimal alignment on h. (assume p is inside a polygon)

Perform a ray search in both directions from r*, along the line.

Let l and u be the boundaries where $\mathbb{A}$ is still optimal.

The line segment (l,u) is a face of the polygon for $\mathbb{A}$.



Things we know so far

For a parameter setting, we can find the optimal alignment.

Two alignments will have a line in (ɣ, δ)-space where they are co-optimal*.

For any point, the optimal alignment is optimal for a point, a line, or a region.

There are a limited number of regions for a fixed input.

Given a point and a ray, we can find a point (and a line) that is at the boundary for the polygon p is in (if it is inside a polygon).

Given a point, a ray, we can find a face of the polygon (if it is inside a polygon).

How many regions are there?
Can we find them?

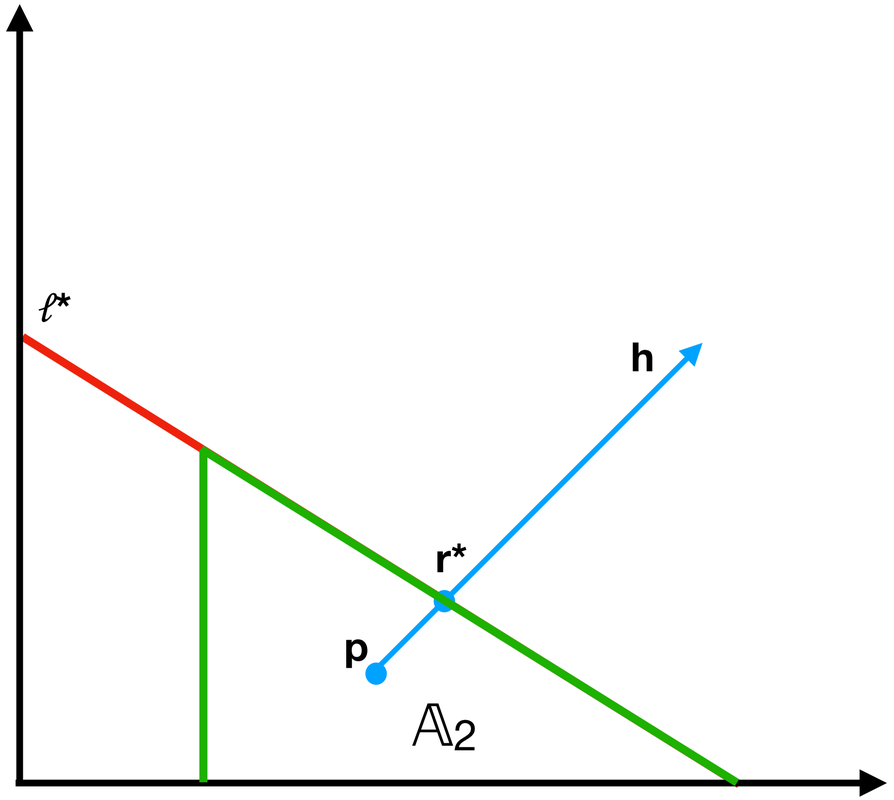
Finding other faces

Given a point p, and a subset of the faces of the polygon in which p resides. (assume p is inside a polygon).

Find a new h that does not intersect any existing faces.

Apply the ray finding algorithm to find r*.

Apply the ray finding twice to find the face of the polygon that intersects r*.



Completing the polygon

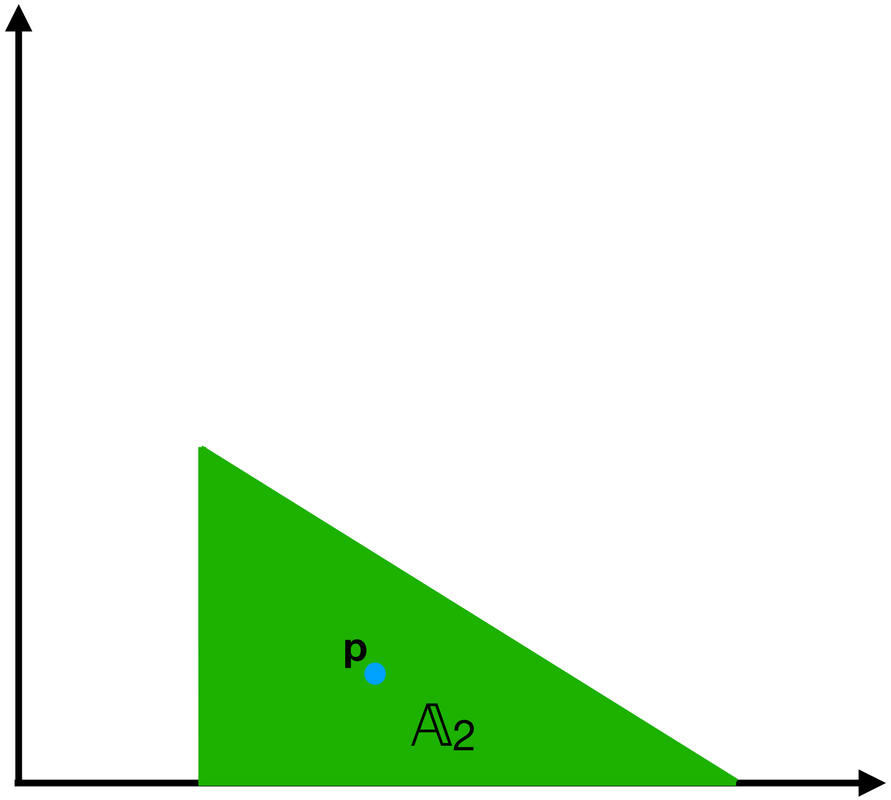
Given a point p, and a subset of the faces of the polygon in which p resides. (assume p is inside a polygon).

Find a new h that does not intersect any existing faces.

Apply the ray finding algorithm to find r*.

Apply the ray finding twice to find the face of the polygon that intersects r*.

Repeat until no additional rays can be placed from p.



The degenerate cases

r* is on the border of the parameter space

- one of the edges of the polygon is the edge of the space.
- use the border line as l* to find edge.

r* is a vertex of the polygon

- one of the ray searches along l* will not find any point beyond r*.
- Stop and use another h that avoids the current r*.

Things we know so far

For a parameter setting, we can find the optimal alignment.

Two alignments will have a line in (ɣ, δ)-space where they are co-optimal*.

For any point, the optimal alignment is optimal for a point, a line, or a region.

There are a limited number of regions for a fixed input.

Given a point and a ray, we can find a point (and a line) that is at the boundary for the polygon p is in (if it is inside a polygon).

Given a point, a ray, we can find a face of the polygon (if it is inside a polygon).

Given a point, find the polygon that it resides in (if it is inside a polygon).

How many regions are there?
Can we find them?

Finding a starting point

How do we ensure that a point is on the interior of a polygon?

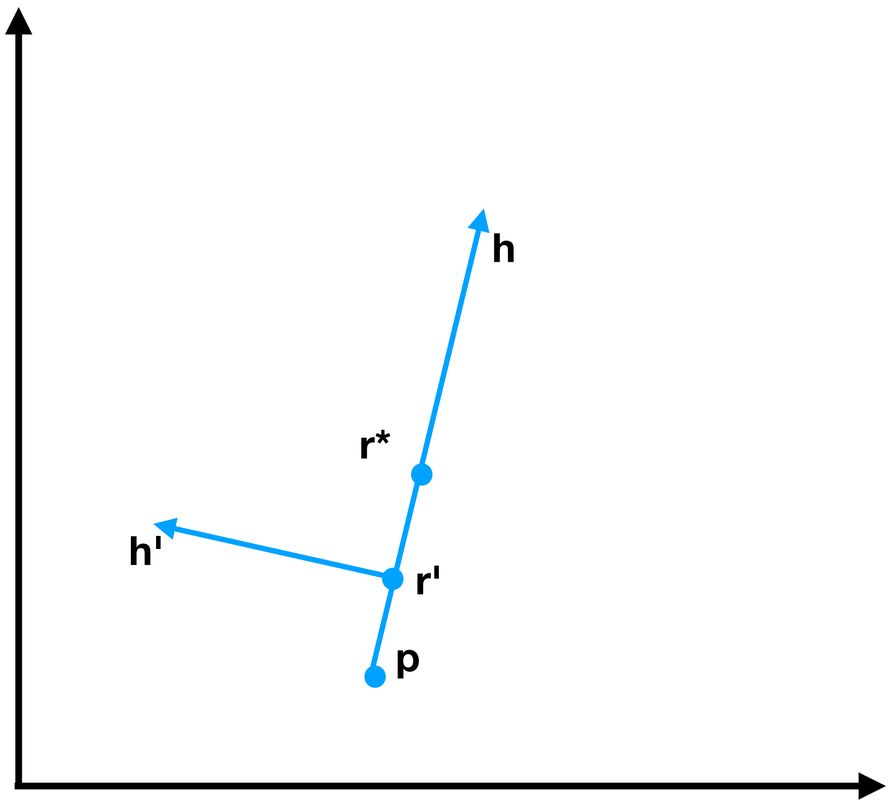
If r*=p --- $\mathbb{A}^*$ is optimal for some non-zero distance along h.

If r*!=p --- $\mathbb{A}$ is optimal for some non-zero distance along h.

Could be optimal only at a line.

Choose a point in the range where $\mathbb{A}/\mathbb{A}^*$ is optimal, perform a ray search in a perpendicular direction.

$\mathbb{A}/\mathbb{A}^*$ is either optimal for some distance along h', or some alignment that is optimal for some distance is returned.



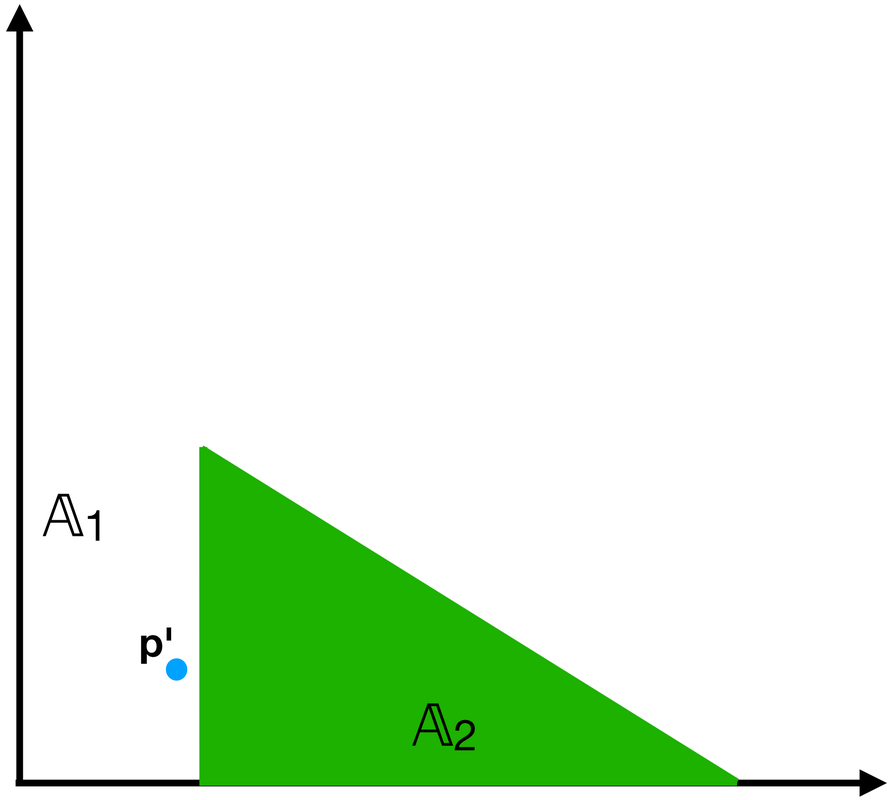
Completing the decomposition

Find a new point p' outside any existing polygon, but internal to another polygon.

Each time ray-search is run, for each alignment seen, insert into a list of alignments if its not already there.

We know these alignments are internal to some polygon.

Use an unmkarked point from this list, and mark it.



Completing the decomposition

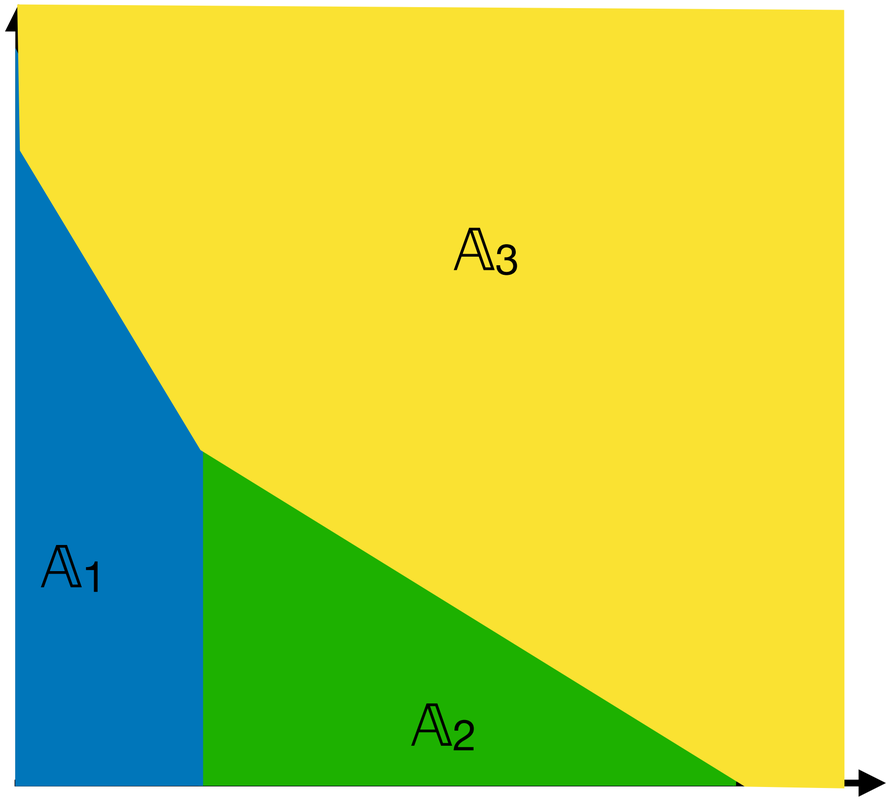
Find a new point p' outside any existing polygon, but internal to another polygon.

Each time ray-search is run, for each alignment seen, insert into a list of alignments if its not already there.

We know these alignments are internal to some polygon.

Use an unmkarked point from this list, and mark it.

When entire list is marked, the decomposition is complete.



Things we know so far

For a parameter setting, we can find the optimal alignment.

Two alignments will have a line in (ɣ, δ)-space where they are co-optimal*.

For any point, the optimal alignment is optimal for a point, a line, or a region.

There are a limited number of regions for a fixed input.

Given a point and a ray, we can find a point (and a line) that is at the boundary for the polygon p is in (if it is inside a polygon).

Given a point, a ray, we can find a face of the polygon (if it is inside a polygon).

Given a point, find the polygon that it resides in (if it is inside a polygon).

In the process of ray search, we can list all representative alignments.

How many regions are there?

✓ **Can we find them?**

Bounding the number of regions

Theorem:
No matter which two of the four parameters are chosen to be variable, the polygon decomposition can contain at most **O(m²)** polygons.

Proof:

Without loss of generality, let $\alpha=\alpha_0$ and $\beta=\beta_0$.

For any alignment $\mathbb{A}$, let $C_{\mathbb{A}} = \alpha_0 \mathbf{mt}_{\mathbb{A}} - \beta_0 \mathbf{ms}_{\mathbb{A}}$.

Then each alignment is represented by the tuple ($C_{\mathbb{A}}$, $\mathbf{id}_{\mathbb{A}}$, $\mathbf{gp}_{\mathbb{A}}$).

If some $\mathbb{A}'$ has $\mathbf{id}_{\mathbb{A}}$ indels, $\mathbf{gp}_{\mathbb{A}}$ gaps, and $C_{\mathbb{A}} < C_{\mathbb{A}'}$ then $\mathbb{A}'$ cannot be optimal for any (χ, δ).

For all triples with the last two variables $\mathbf{id}_{\mathbb{A}}$, $\mathbf{gp}_{\mathbb{A}}$, at most 1 can be optimal at some point.

If $n \leq m$ are the lengths of the string, there can be at most m+n gaps, and m+n indels.

Any two alignments with the same triple are optimal at exactly the same points.

Decomposition Speed

How many regions can a single ray intersect with?

- as many regions as there are, $O(m^2)$

How much work needs to be done at each intersection?

- optimal alignment at that point, compare the alignments, $O(m^2)$

How much work needs to be done to find the ends of a boundary?

- two ray searches at the found r^*, which is $O(m^4)$ work once you get there

Therefore a boundary can be found in $O(m^4)$-time

Total decomposition can be found in that proportional to the number of edges in the decomposition E, which is bounded by $O(n^2)$, so the total time is $O(n^6)$

Can be $O(m^4)$!

Decomposition Speed

Keep a list *L* of optimal alignments found along the way (the values of **mt**, **ms**, **id**, & **gp**)

Any time a new *h* is chosen, find the place where the line intersecting the optimal alignment and every alignment in *L* intersect *h*

Start the ray search at the closest to *p* rather than the boundary.

The size of *L* is bounded by sum of the number of vertices (V), edges (E), and polygons (R) in the decomposition.

O(E) ray searches to find all edges, therefore $O(E(V+E+R)) = O(m^4)$ extra work to use *L.*

When doing a ray search, we only compute an alignment for points not in *L*, then they are added to *L*, so the number of alignments is bounded by the same size as *L*, therefore the alignment running time is $O((V+E+R)m^2) = O(m^4)$

Parametric sequence alignment

For a fixed input:

- there are $O(m^2)$ optimal alignments when two parameters are free
- the regions can be found by repeated ray-search

More free parameters

$$f_{\alpha,\beta,\gamma,\delta}(\mathbb{A}) = \alpha \cdot \mathbf{mt}_{\mathbb{A}} - \beta \cdot \mathbf{ms}_{\mathbb{A}} - \gamma \cdot \mathbf{id}_{\mathbb{A}} - \delta \cdot \mathbf{gp}_{\mathbb{A}}$$

- $mt_{\mathbb{A}}$ -- number of columns where both characters match
- $ms_{\mathbb{A}}$ -- number of columns where there characters are different (mismatches)
- $id_{\mathbb{A}}$ -- number of gap characters (indels)
- $gp_{\mathbb{A}}$ -- number of gaps

Using the same argument as with 2 parameters, how many polygons are possible when all 4 parameters are free?

- how many values of $\mathbf{mt}_{\mathbb{A}}$ and $\mathbf{ms}_{\mathbb{A}}$ can there be for a single input?
- $O(m^4)$

Later work shows that this bound can be reduced to $O(m^{d+1/d-1})$

How does our work contribute

Several projects in my group involve finding optimal parameters for tools

- specifically using alignment as a base problem

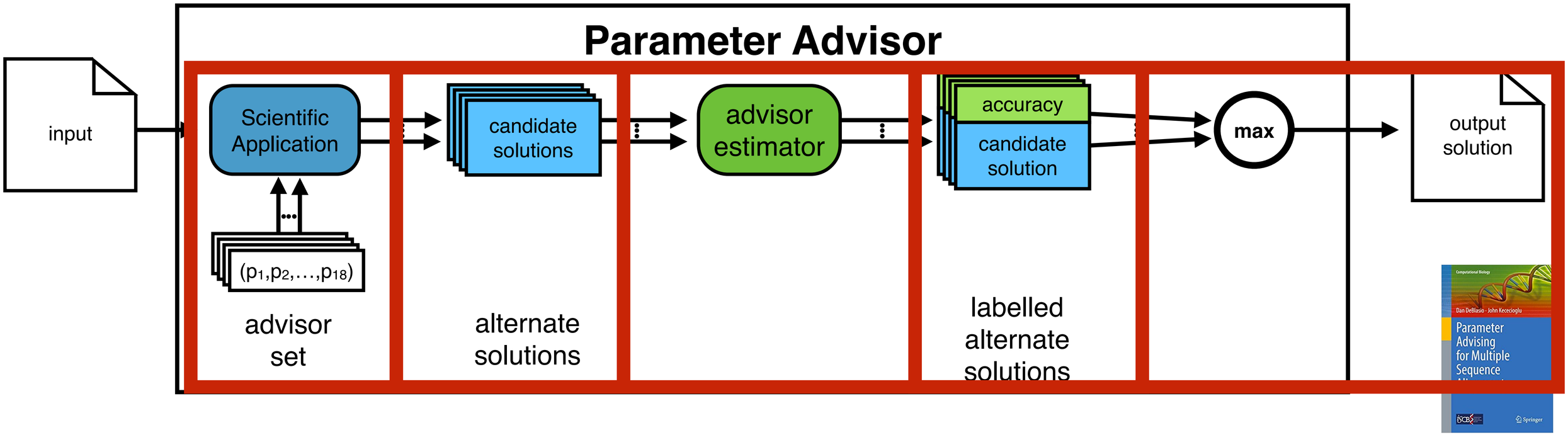
We call this *algorithm configuration*

- an open problem in most domains
- we work not only in alignment but other biological and non-biology applications

Parameter advising framework

Steps of advising:

- An advisor set of parameter choice vectors is used to obtain candidates.
- Solutions are ranked based on the accuracy estimation.
- The highest ranked candidate is returned.



[DK, Springer International, 2017]

Parameter advising framework

Steps of advising:

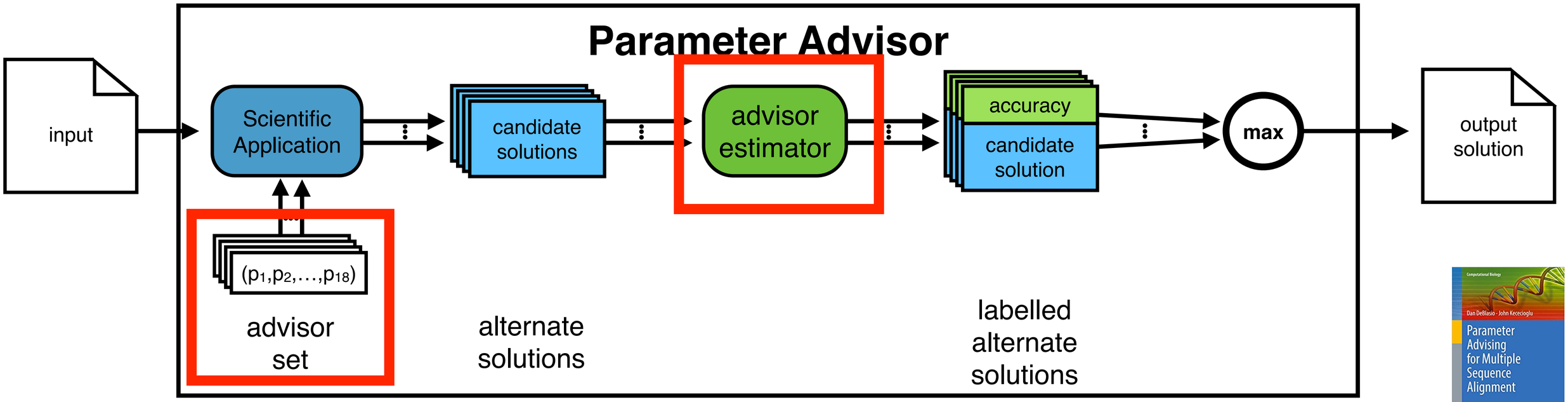
- An advisor set of parameter choice vectors is used to obtain candidates.
- Solutions are ranked based on the accuracy estimation.
- The highest ranked candidate is returned.

[DK, Springer International, 2017]

Parameter advising framework

Components of an advisor:

- An advisor set of parameter choice vectors.
- An advisor estimator to rank solutions.



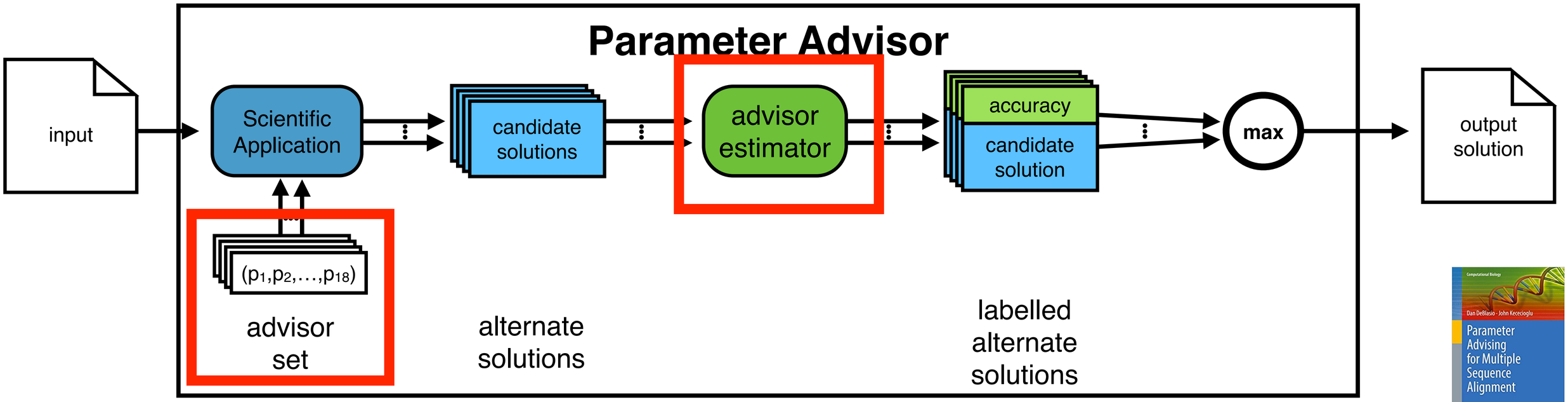
[DK, Springer International, 2017]

Parameter advising framework

Components of an advisor:

- An advisor set of parameter choice vectors.
- An advisor estimator to rank solutions.

A good advisor set:
- Small
- Representative



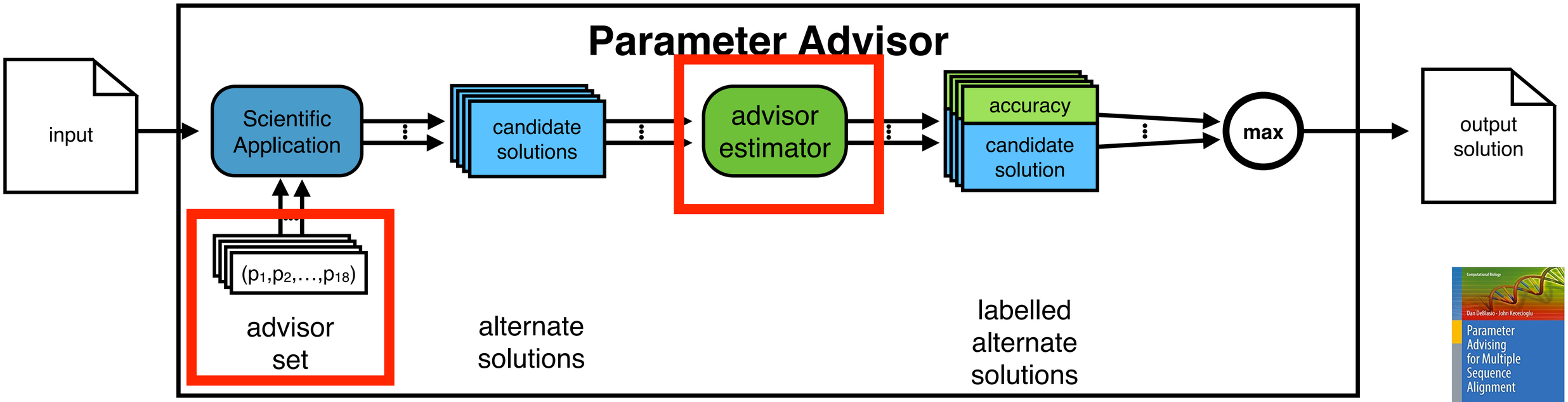
[DK, Springer International, 2017]

Parameter advising framework

Components of an advisor:

- An advisor set of parameter choice vectors.
- An advisor estimator to rank solutions.

A good advisor estimator:
- Efficient
- Rank Solutions Well

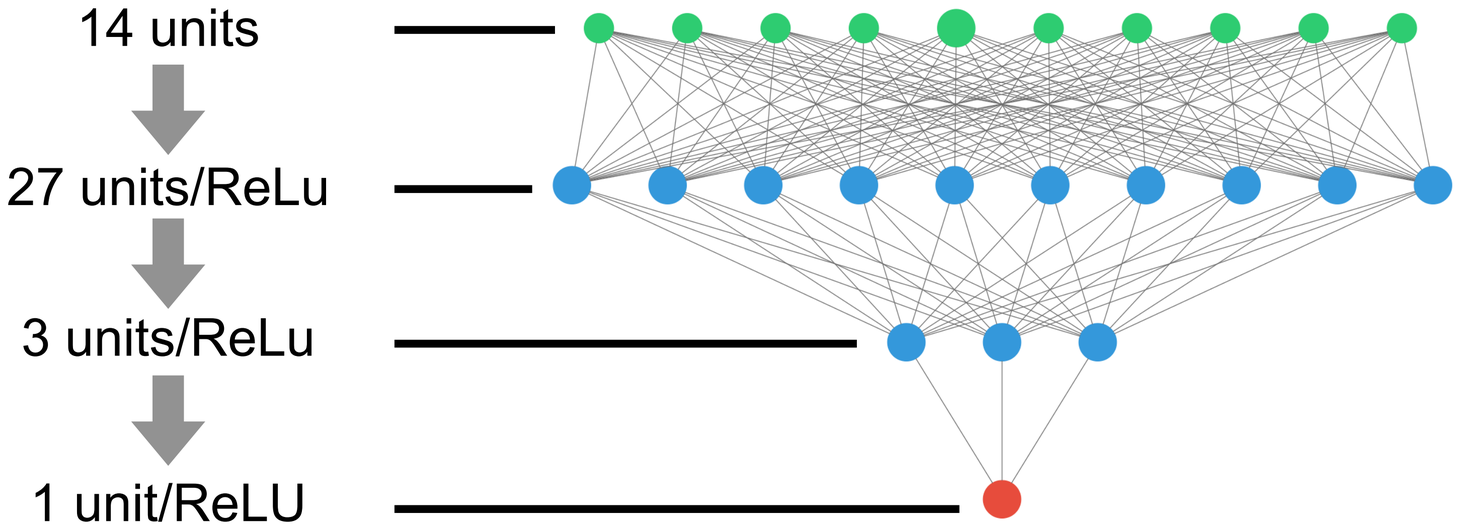


[DK, Springer International, 2017]

Exploiting non-linearity

While we designed the features to scale linearly with accuracy, some show some non-linear behavior when plotted.

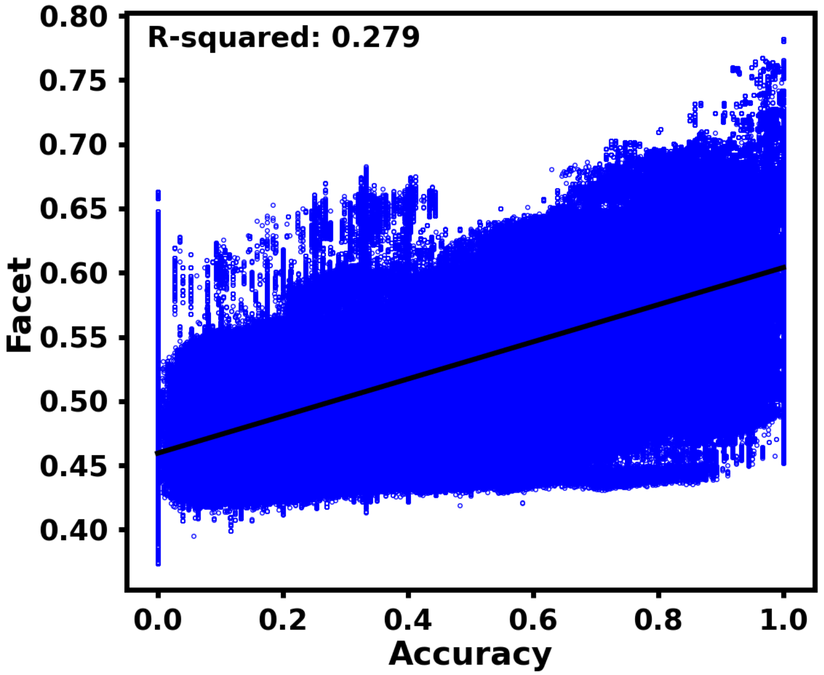
- Advanced machine learning allowed for the use of a neural network predictor.
- We also produced a much larger training set (now >14M alignments).



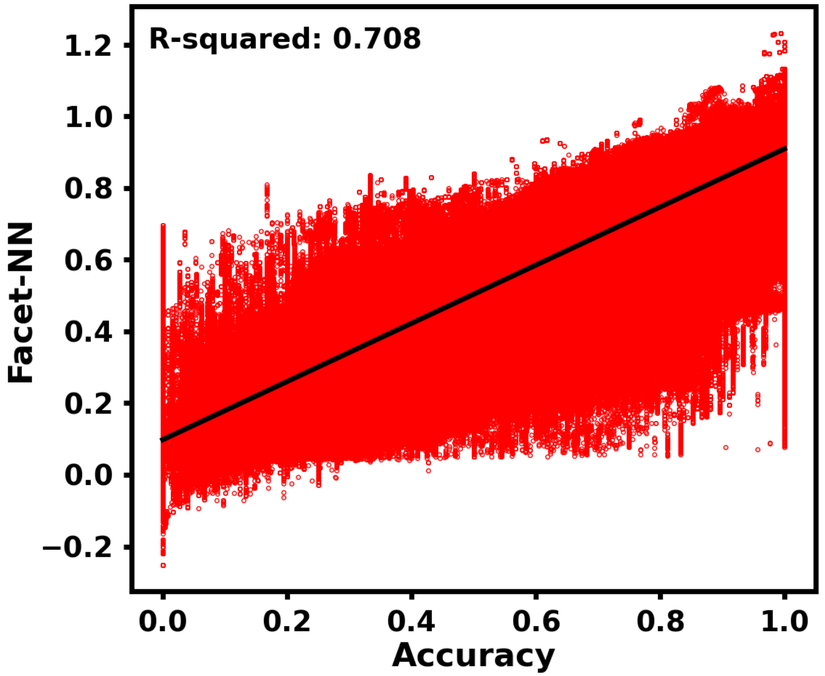
Cedillo, Richart Ruiz, DeBlasio. **Exploiting Large Datasets Improves Accuracy Estimation for Multiple Sequence Alignment.** bioRxiv:10.1101/2022.05.22.493004

Exploiting non-linearity

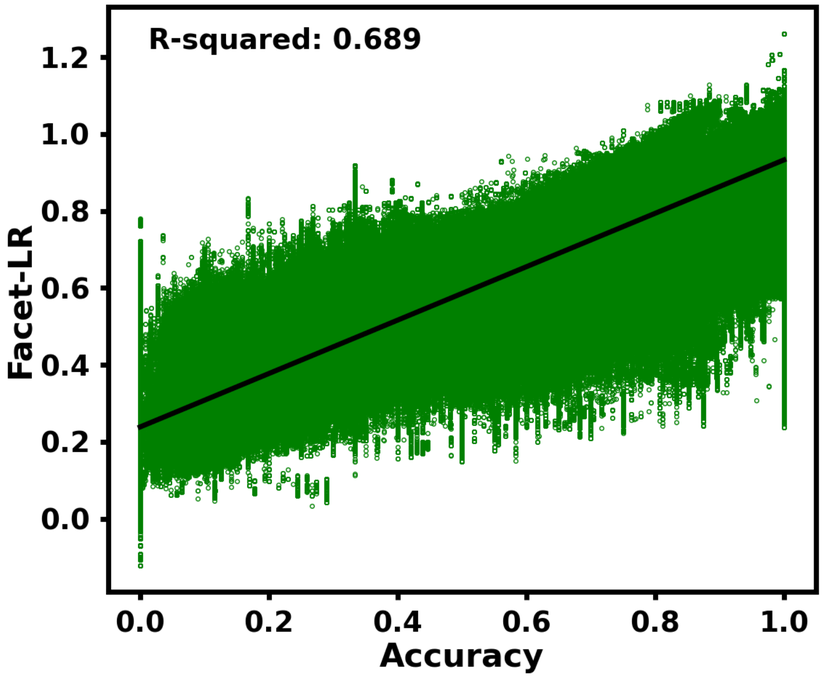
Previous Result



Neural Network



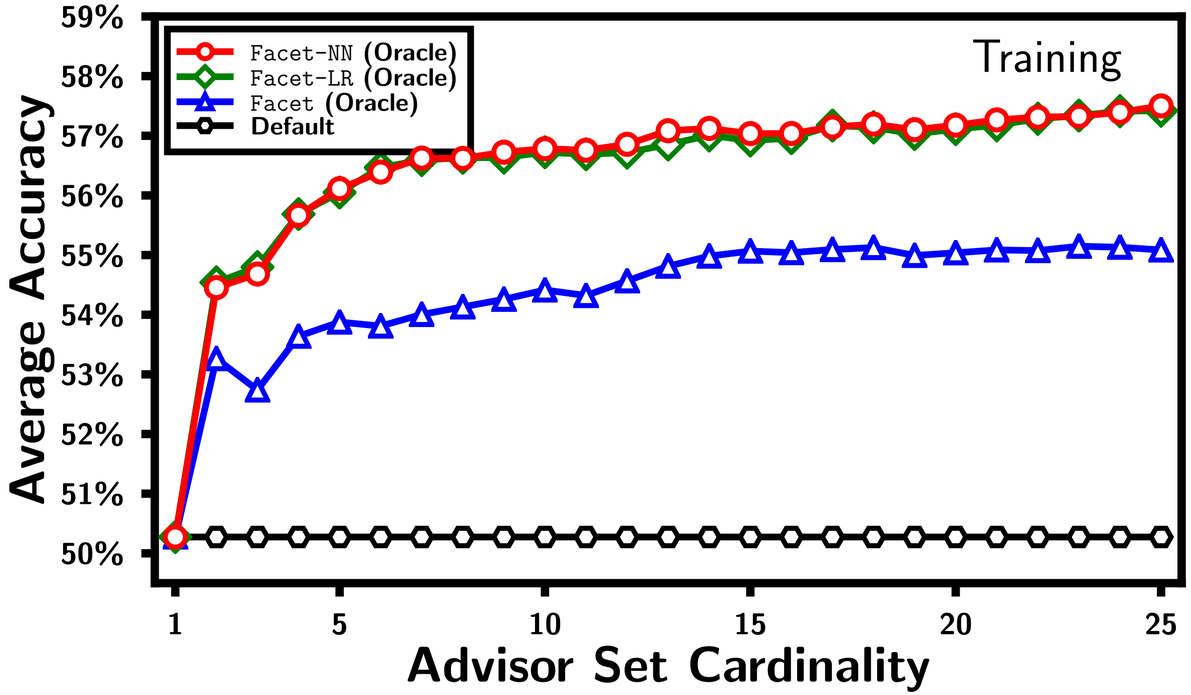
Linear Regression

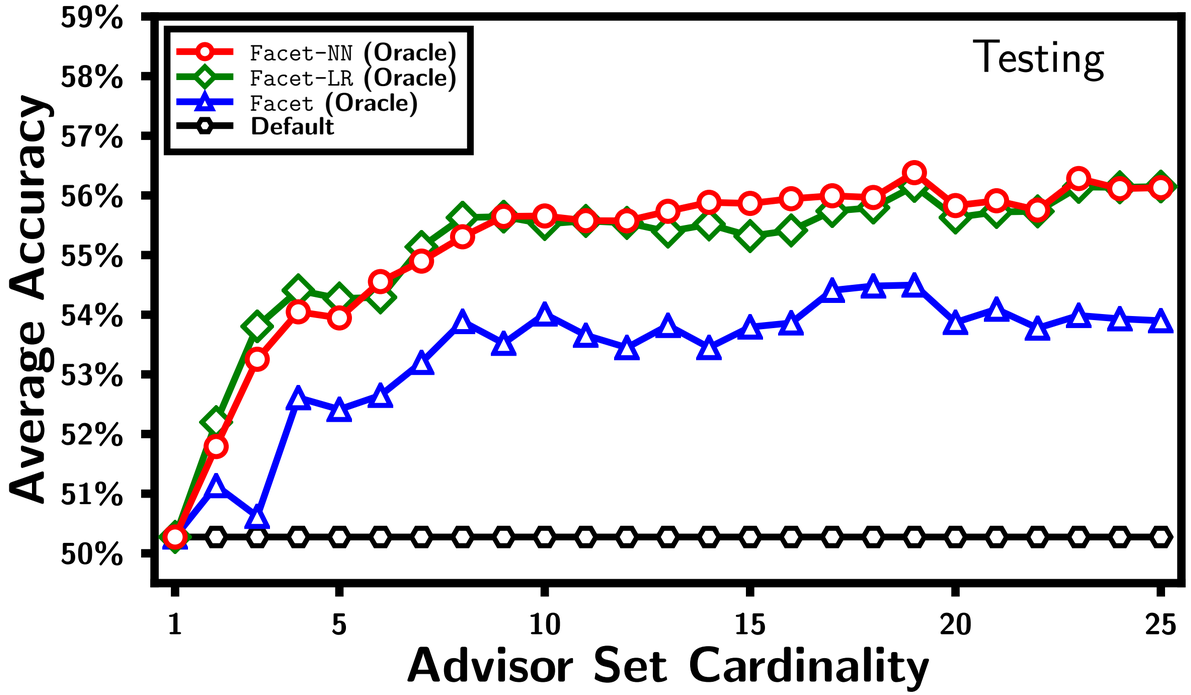


Modern techniques and larger training also lead to a more accurate linear model.

Cedillo, Richart Ruiz, DeBlasio. **Exploiting Large Datasets Improves Accuracy Estimation for Multiple Sequence Alignment.** bioRxiv:10.1101/2022.05.22.493004

Advising for Multiple Sequence Alignment



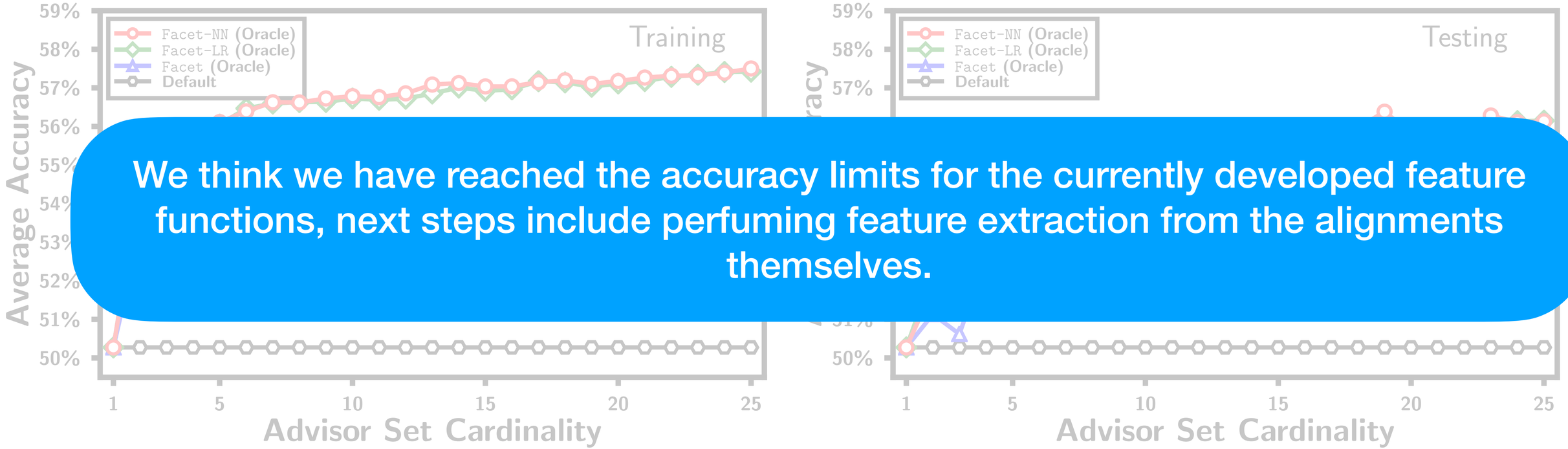


Facet-NN and Facet-LR outperform original Facet on the advising task.

slides: deblasiolab.org/parametric-sequence-alignment/

Cedillo, Richart Ruiz, DeBlasio. **Exploiting Large Datasets Improves Accuracy Estimation for Multiple Sequence Alignment.** bioRxiv:10.1101/2022.05.22.493004

Advising for Multiple Sequence Alignment



We think we have reached the accuracy limits for the currently developed feature functions, next steps include perfuming feature extraction from the alignments themselves.

Facet-NN and Facet-LR outperform original Facet on the advising task.

slides: deblasiolab.org/parametric-sequence-alignment/

Cedillo, Richart Ruiz, DeBlasio. **Exploiting Large Datasets Improves Accuracy Estimation for Multiple Sequence Alignment.** bioRxiv:10.1101/2022.05.22.493004

Other ongoing/continuing projects

Multiple sequence alignment benchmark set bias identification

- Fransisco Parra (Senior/REU), Luis Cedillo (Sophomore)

Trying to use unsupervised learning to help stratify the training benchmarks
There is a high number of easy-to-align benchmarks, want to compensate in training.

Other ongoing/continuing projects

Multiple sequence alignment benchmark set bias identification

- Fransisco Parra (Senior/REU), Luis Cedillo (Sophomore)

Using neural networks to improve genome assembly speed by encoding minimizer schemes

- Demetrius Hernandez (Senior)

Currently working toward encoding existing schemes, with plans to expand to learning *local* schemes.

Other ongoing/continuing projects

Multiple sequence alignment benchmark set bias identification

- Fransisco Parra (Senior/REU), Luis Cedillo (Sophomore)

Using neural networks to improve genome assembly speed by encoding minimizer schemes

- Demetrius Hernandez (Senior)

Automatic parameter configuration for additive manufacturing

- Fernando Sepulveda (Freshman)
- Collaboration with faculty in Electrical and Computer Engineering

Identifying unresolved space objects using ground-based hyperspectral imaging

- Taposh Sarker (Graduate Student)
- Collaboration with faculty in Electrical and Computer Engineering

Acknowledgments

The DeBlasio Lab

Luis Cedillo (UG, Facet-NN/LR)

Demetrius Hernandez (UG, Minimizers)

Taposh Sarkar (PhD, USROs)

Fernando Sepulveda (UG, Additive Manufacturing)

Md. Easin Hasan (former MS, BWA)

Hector Richart-Ruiz (former UG)

The Current Collaborators

Miguel Velez-Reyes

Arizbe Najera

Previous Collaborators

The University of Arizona | Computer Science

John Kececioglu

Travis Wheeler (Montana)

Jen Wisecaver (Purdue)

Computational Biology Department | Carnegie Mellon University School of Computer Science

Carl Kingsford

Fiyinfoluwa Gbosibo (visiting UG)

Kwanho Kim (MS)

Guillaume Marçais

Funding

Contact

deblasiolab.org

danfdeblasio

slides: deblasiolab.org/parametric-sequence-alignment/