Predicting core columns of protein multiple sequence alignments for improved parameter advising

Dan DeBlasio
John Kececioglu

Workshop on Algorithms in Bioinformatics
August 24, 2016

Motivation

Multiple sequence alignment is a fundamental problem in bioinformatics.

- multiple sequence alignment is NP-Complete
- many popular aligners for multiple sequence alignment
- each aligner has many parameters whose values affect the alignment that is output

Motivation

Aligners often use *one* default parameter choice for *all* inputs.

- The default has good *average accuracy* across all benchmarks.
- The optimal default choice can be found by inverse alignment [Kececioglu and Kim 2007].
- The default may be a poor choice for specific inputs.

default

```
... yl-lhqflspssnqrtdqyggsvenrarlvlevvdavcnewsad-RIGIRVSPigtfq ...
... kP-LGVKLPPyf--dlvhfdimaeilnqfpltyvsnv-nsig----nglfidpeaesv ...
... yl-lnqfldphsntrtdeyggsienrarftlevvdalveaighe-KVGLRLSPygvfn ...
... yl-plqflnpyynkrtdkyggslenrarfwletlekvkhavgsdcAIATRF---GVdt ...
... kvPLYVKLSPnv-tdivpiakaveaagadgltmintl---------mgvrfdlktrqp ...
```

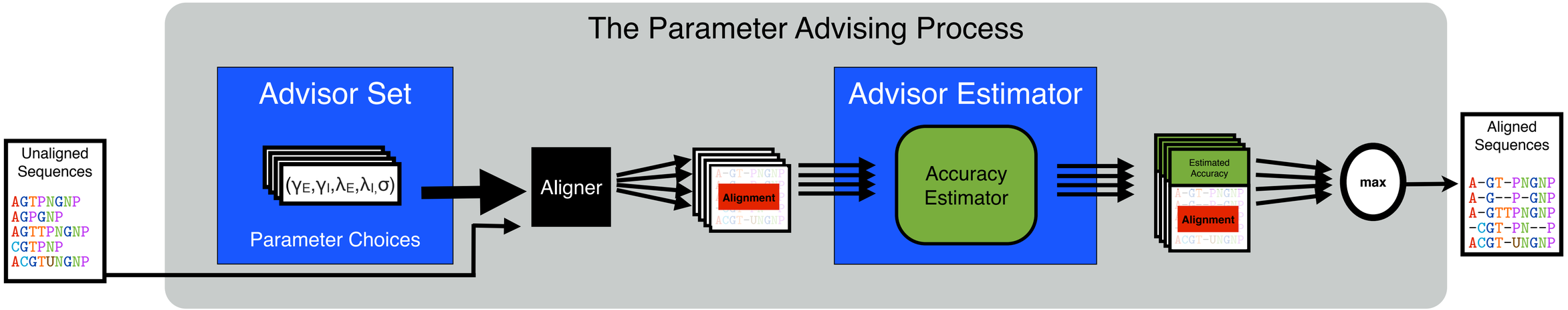
alternate

```
... gsvenrarlvlevvdavcnewsad-RIGIRVSPigtfqnvdngpnee--adalyl--- ...
... ydfeatekllke-----vftfftk-PLGVKLPPyf---------------dlvhfdim ...
... gsienrarftlevvdalveaighe-KVGLRLSPygvfnsmsggaetgivaqyayvage ...
... gslenrarfwletlekvkhavgsdcAIATRFGV-----------------dtvygpgq ...
... tdpevaaalvka-----ckavskv-PLYVKLSPnvt-------------divpiaka ...
```

Parameter advising

Advising for unaligned input sequences

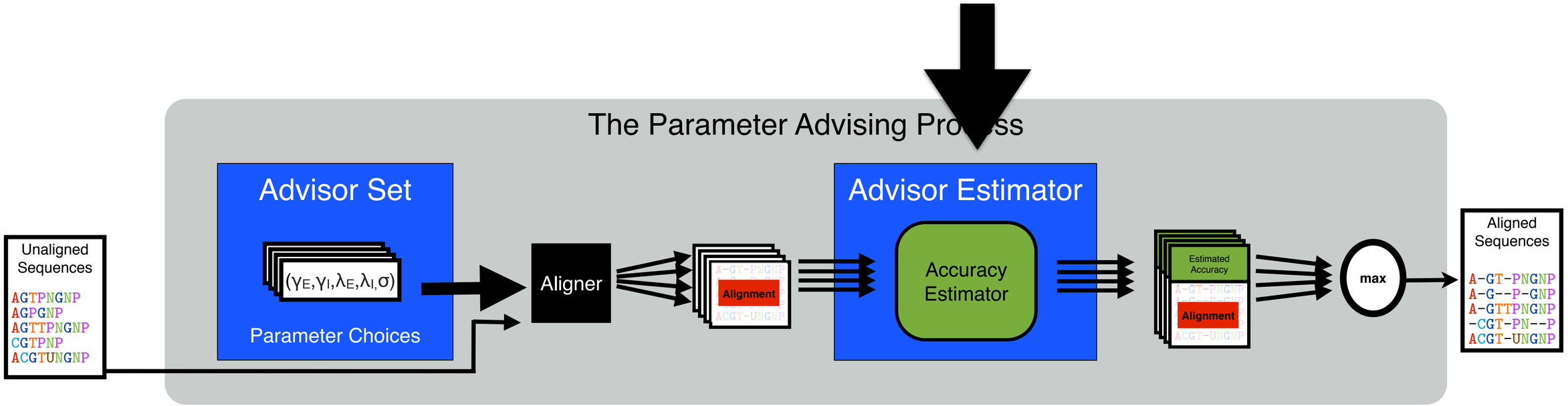
- aligns sequences using each parameter choice from a set,
- assigns an estimated accuracy to each alignment, and
- selects the alignment with the highest estimated accuracy.



Parameter advising

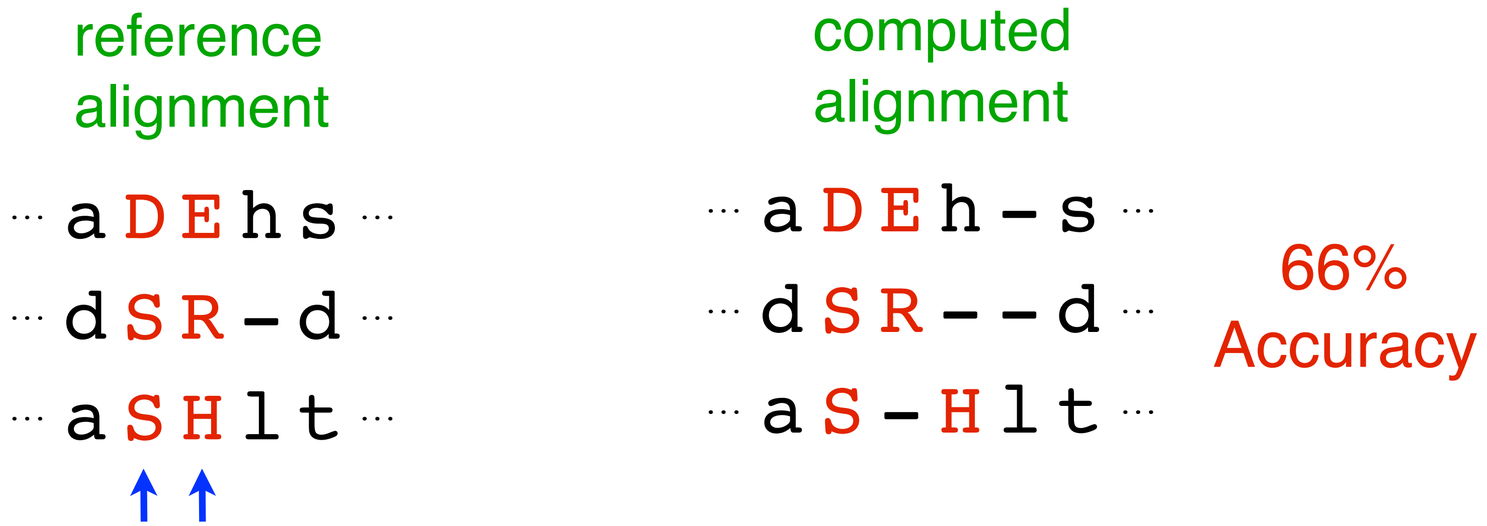
A parameter advisor has two components:

- an accuracy estimator, and
- a set of candidate parameter choices.



Accuracy estimation

Alignment accuracy is measured with respect to a reference alignment.



- accuracy is the fraction of substitutions from the reference that are in the computed alignment,
- measured on the core columns of the reference.

Accuracy estimation

Our estimator `Facet` (“**F**eature-based **AC**curacy **E**s**T**imator”)

- estimates accuracy by a polynomial on feature functions,
- efficiently learns the polynomial coefficients from examples,
- uses novel features that are efficient to evaluate.

Accuracy estimation

The estimator $E(A)$ is a polynomial in the feature functions $f_i(A)$.

linear estimator

$$E(A) \quad := \quad \sum_i c_i \; f_i(A)$$

quadratic estimator

$$E(A) \quad := \quad \sum_i c_i \; f_i(A) + \sum_i \sum_j c_{ij} \; f_i(A) \; f_j(A)$$

Learning the estimator

We learn the estimator using examples consisting of

- an alignment, and
- its associated true accuracy.

Learning finds optimal coefficients that either fit

- accuracy values of the examples, or
- accuracy differences on pairs of examples,
- by solving a linear or quadratic program.

Feature functions

We use protein alignment feature functions that

- are fast to evaluate,
- measure novel properties,
- use non-local information,
- involve secondary structure.

Feature functions

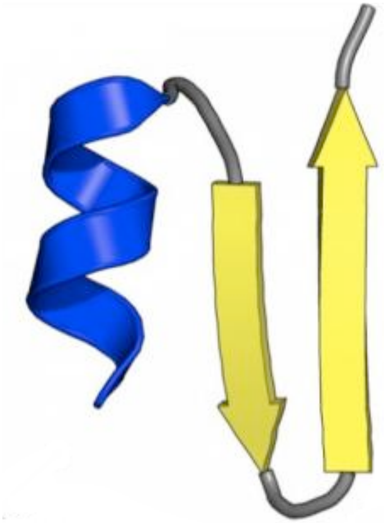
Features based only on the input alignment

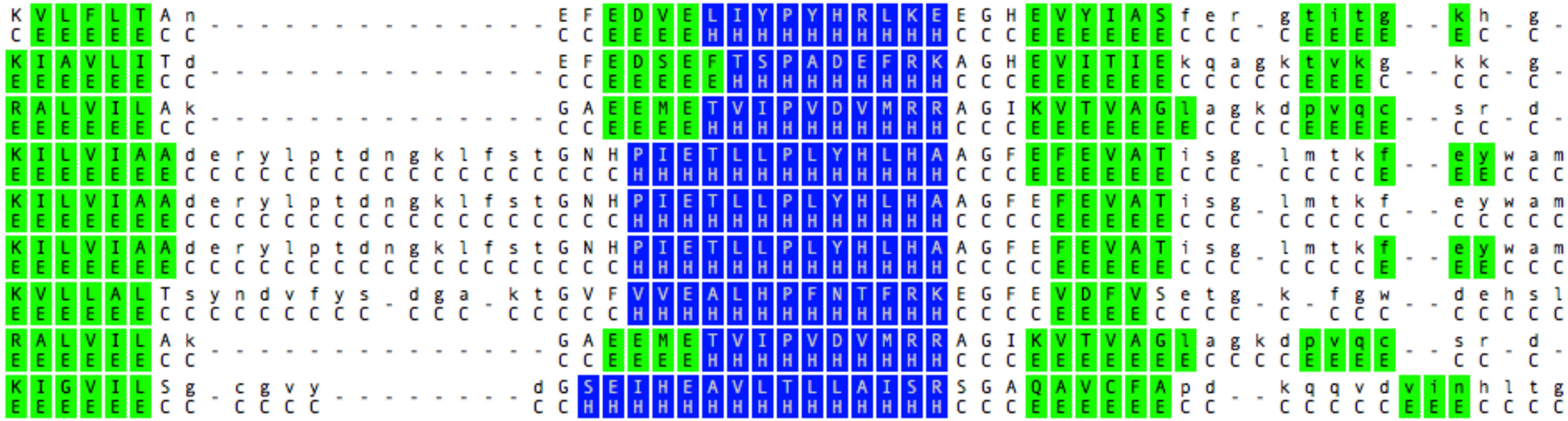
- Amino Acid Identity
- Average Substitution Score
- Information Content
- ...

Feature functions

There are three types of secondary structure

- α-helix,
- β-strand,
- coil.





http://www.ebi.ac.uk/training/online/

Feature functions

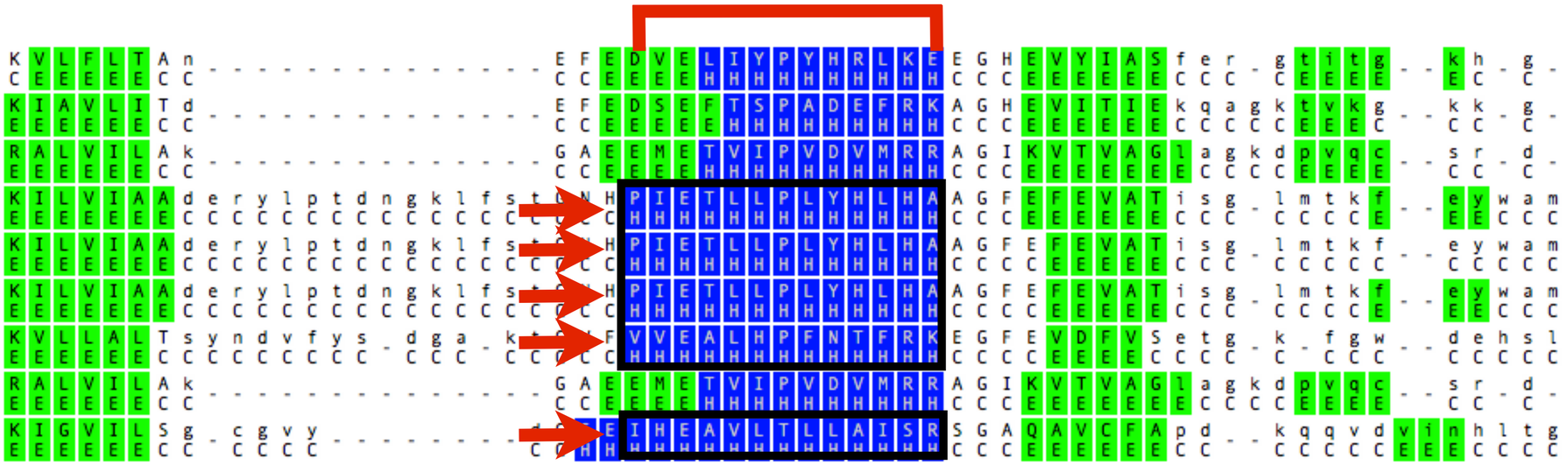
Features using predicted secondary structure

- Secondary Structure Percent Identity
- Secondary Structure Agreement
- Secondary Structure Blockiness
- ...

Secondary Structure Blockiness

A block B in alignment A is

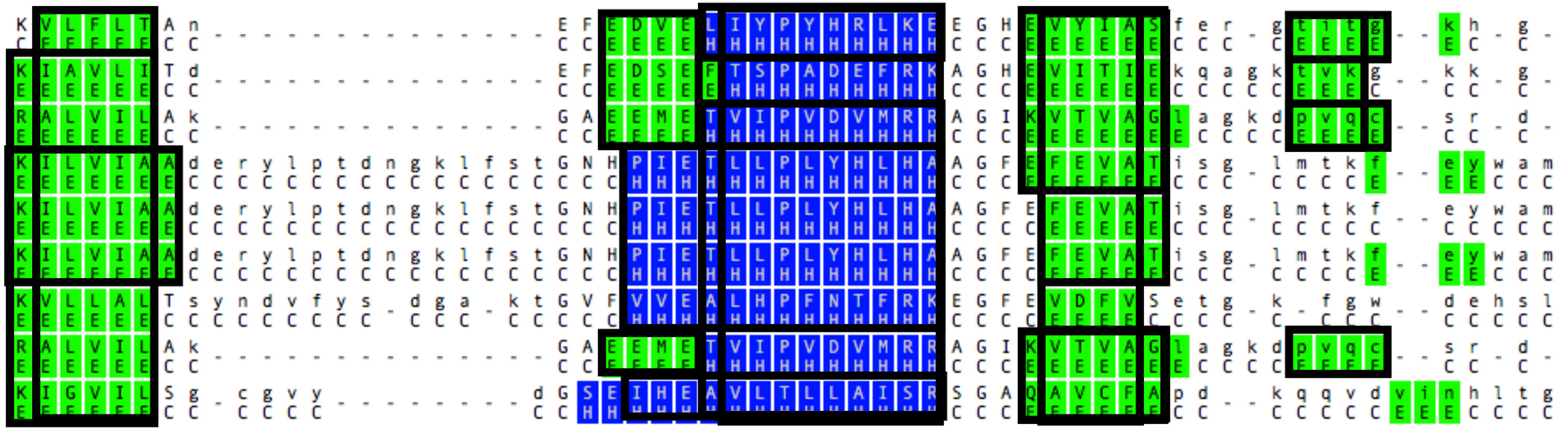
- an interval of at least l columns,
- a subset of at least k rows,
- with the same secondary structure for all residues in B.



Secondary Structure Blockiness

A block B in alignment A is

- an interval of at least l columns,
- a subset of at least k rows,
- with the same secondary structure for all residues in B.



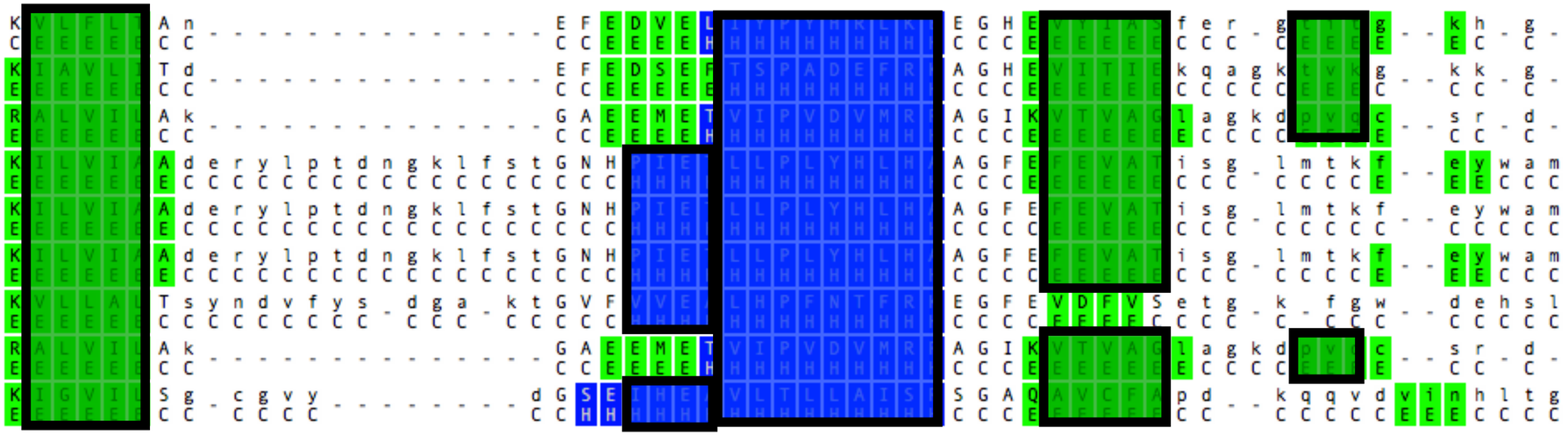
Secondary Structure Blockiness

A packing P for alignment A is

- a set of blocks from A,
- whose columns are disjoint.

The value of P is the number of substitutions it contains.

The Blockiness feature is the maximum value of any packing.



Secondary Structure Blockiness

A packing P for alignment A is

- a set of blocks from A,
- whose columns are disjoint.

The value of P is the number of substitutions it contains.

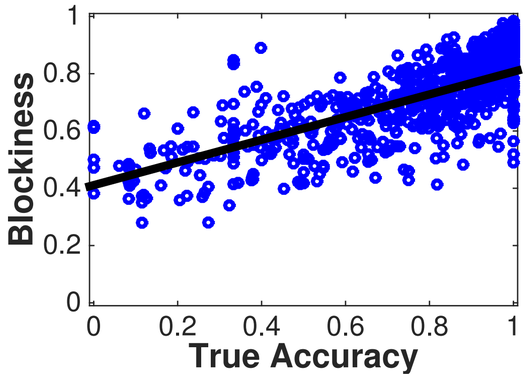
The Blockiness feature is the maximum value of any packing.

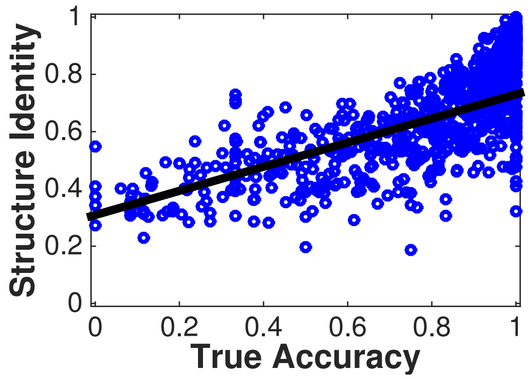
Theorem (Evaluating Blockiness)

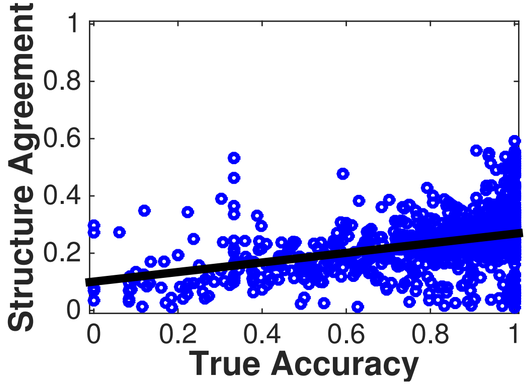
Blockiness can be computed in $O(mn)$ time,
for an alignment with m rows and n columns.

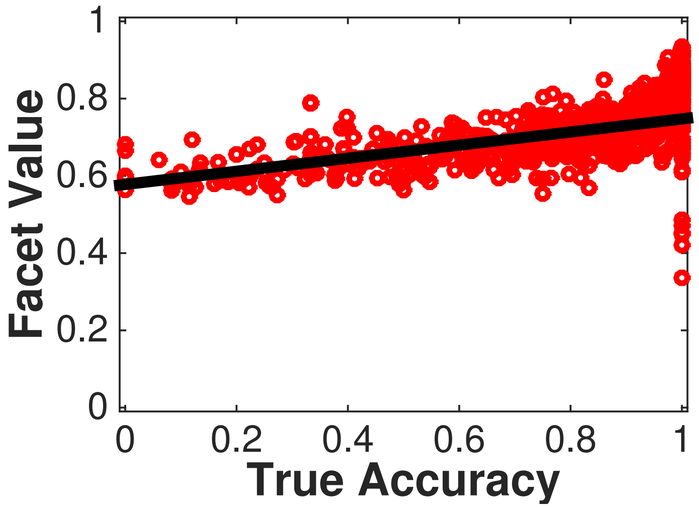
Accuracy estimation

Best features trend well with accuracy.









Facet estimator has less spread than its features.

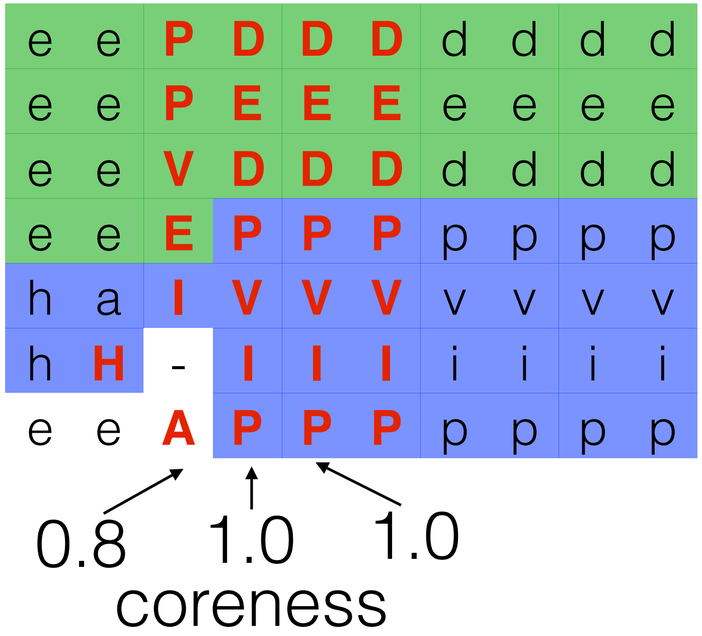
Predicting column coreness

Alignment accuracy is only measured on core columns

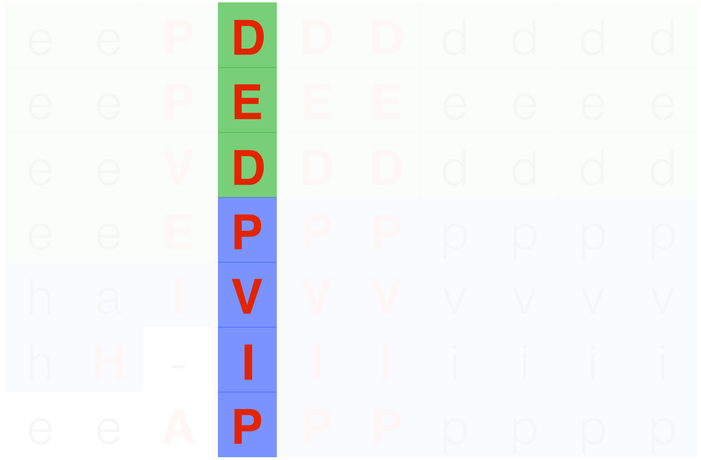
- Highly reliable columns in benchmark alignments
- Coreness is the fraction of a computed column that is core
- Weighting features by coreness improves `Facet`

```
rkeyagLYHEVAQAHGVDVSQVrqMKFGLFFLFDTLAVyqmsegrfafhkiindafttEAARAEARVyleefvresysnt
kkaqldLYNEVATEHGYDVTKId-MKFGNFLLFDTVWLlqmskgrfrfydlmkegfneNRAKDICRNflghwy-dsyvna
rielln HYQAAAAKFNVDIANVr----------------kWNYGVFFLYDVVA--FseTQAKAELSIyledyl--sytqa
rlklls FYNASASKYNKNIDLVr----------------kWNYGVFFVYDVIN--IddTVAKEELKLyienyv--actqp
```

Predicting column coreness

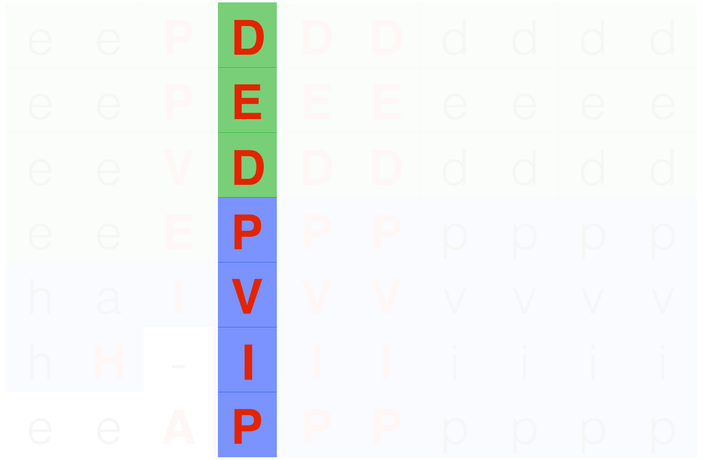


Predicting column coreness

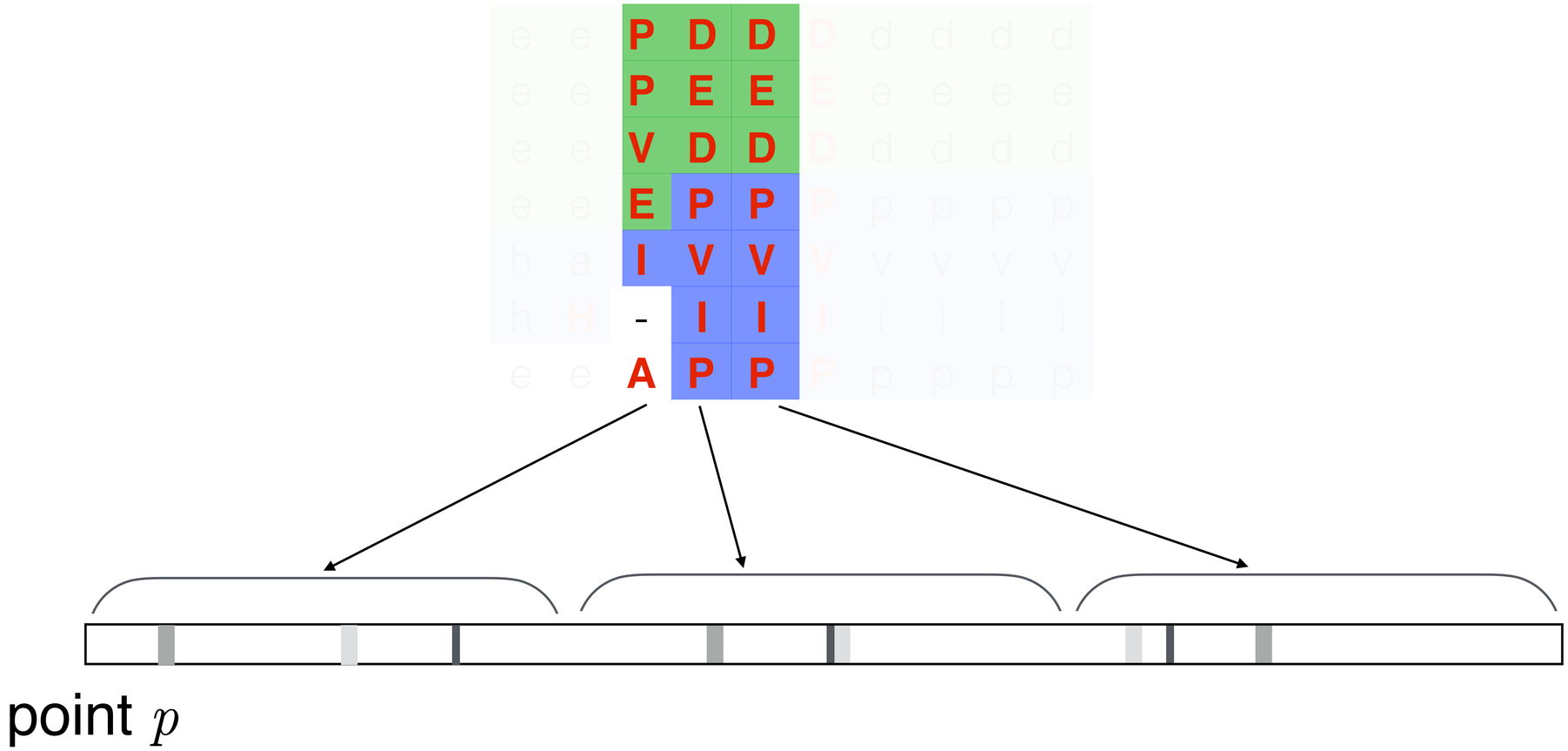


…	(D, α)	(E, α)	…	(P, β)	(V, β)	(I, β)	…
	0.22	0.22		0.33	0.11	0.11	

Predicting column coreness

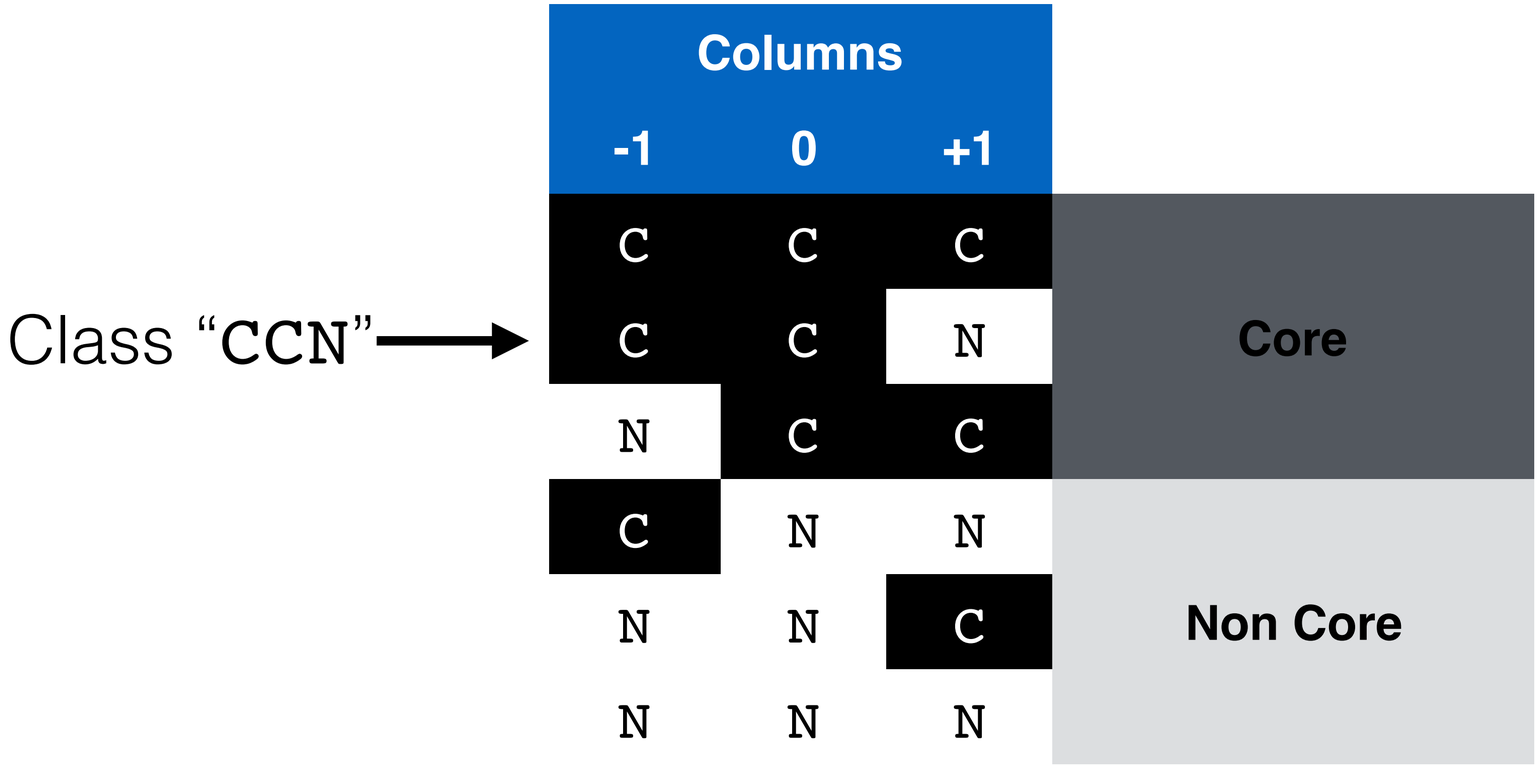


Predicting column coreness



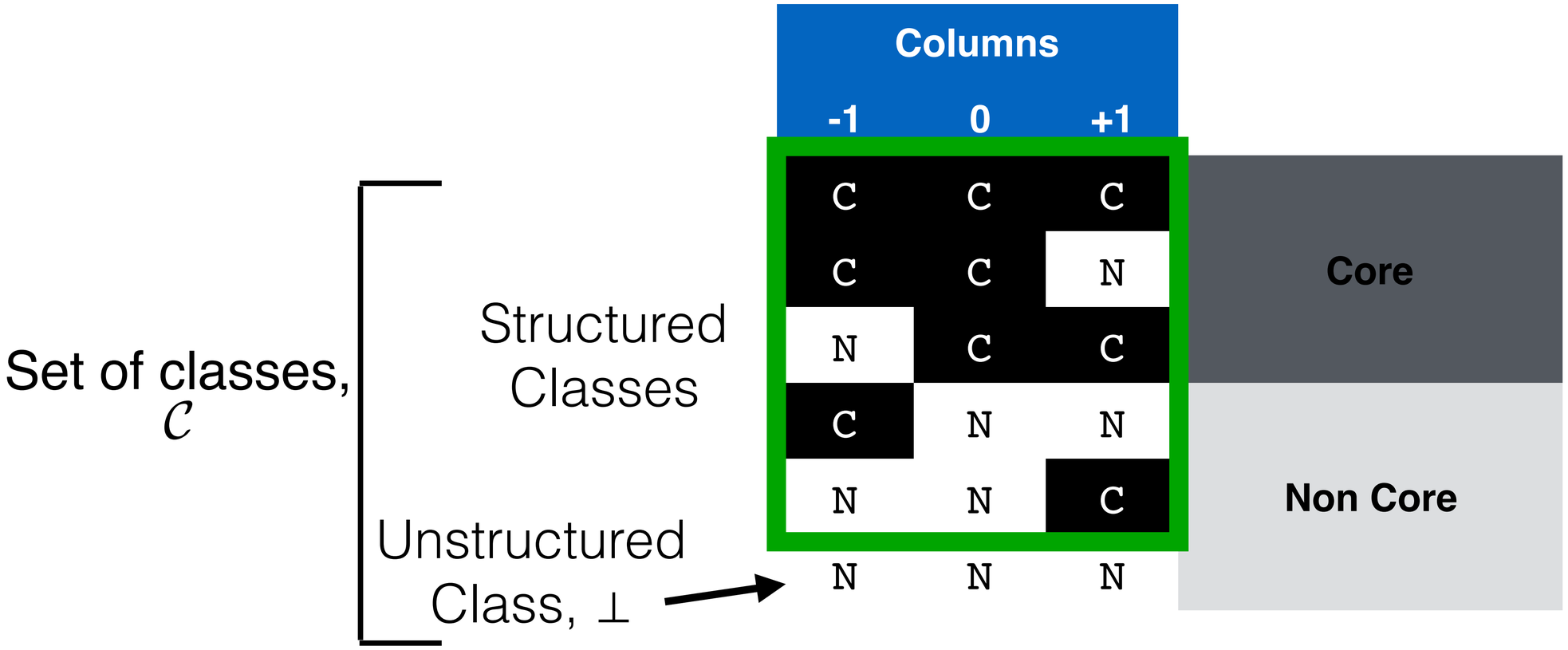
Predicting column coreness

Window classes are dependent on the coreness of individual columns



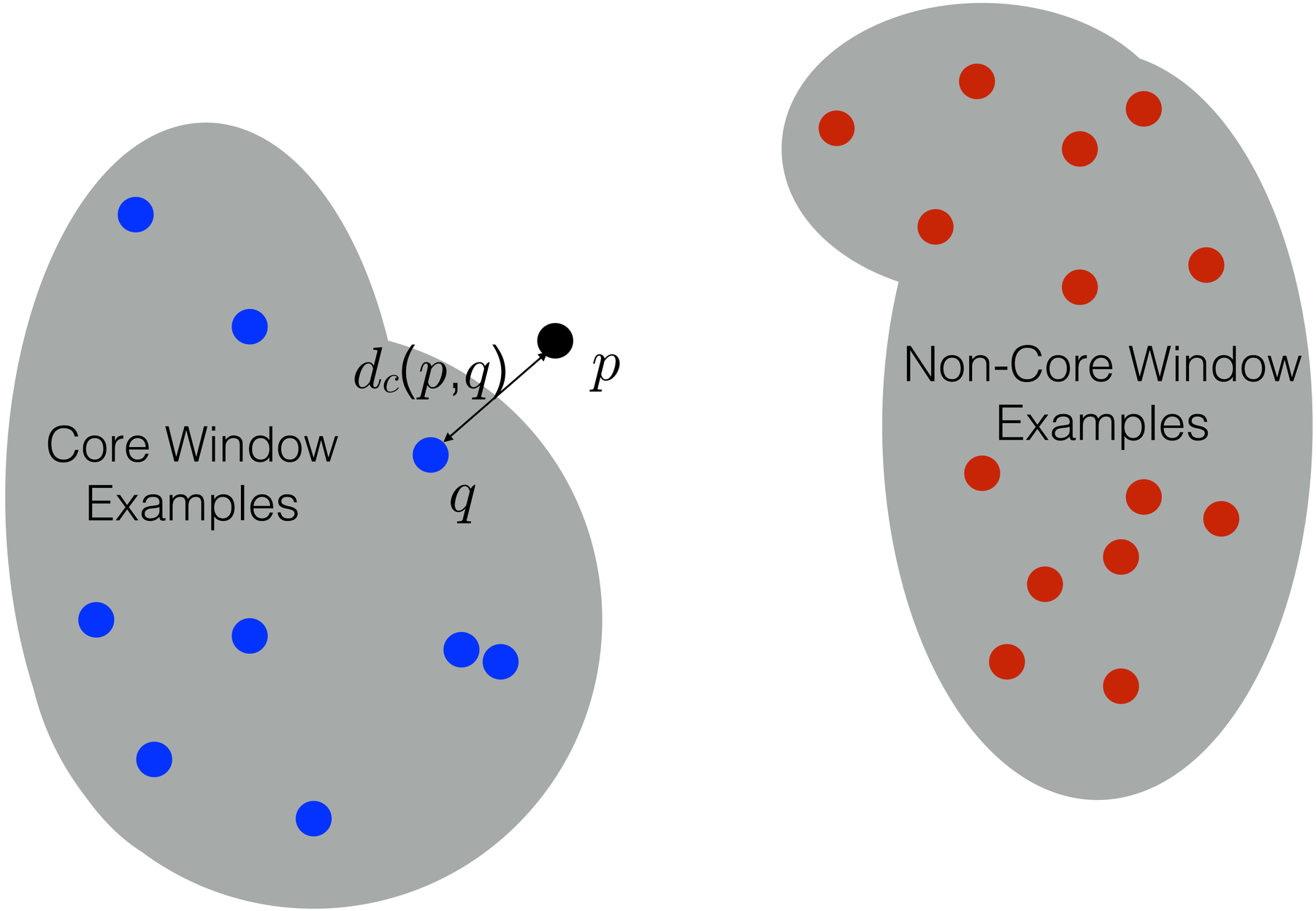
Predicting column coreness

Window classes are dependent on the coreness of individual columns



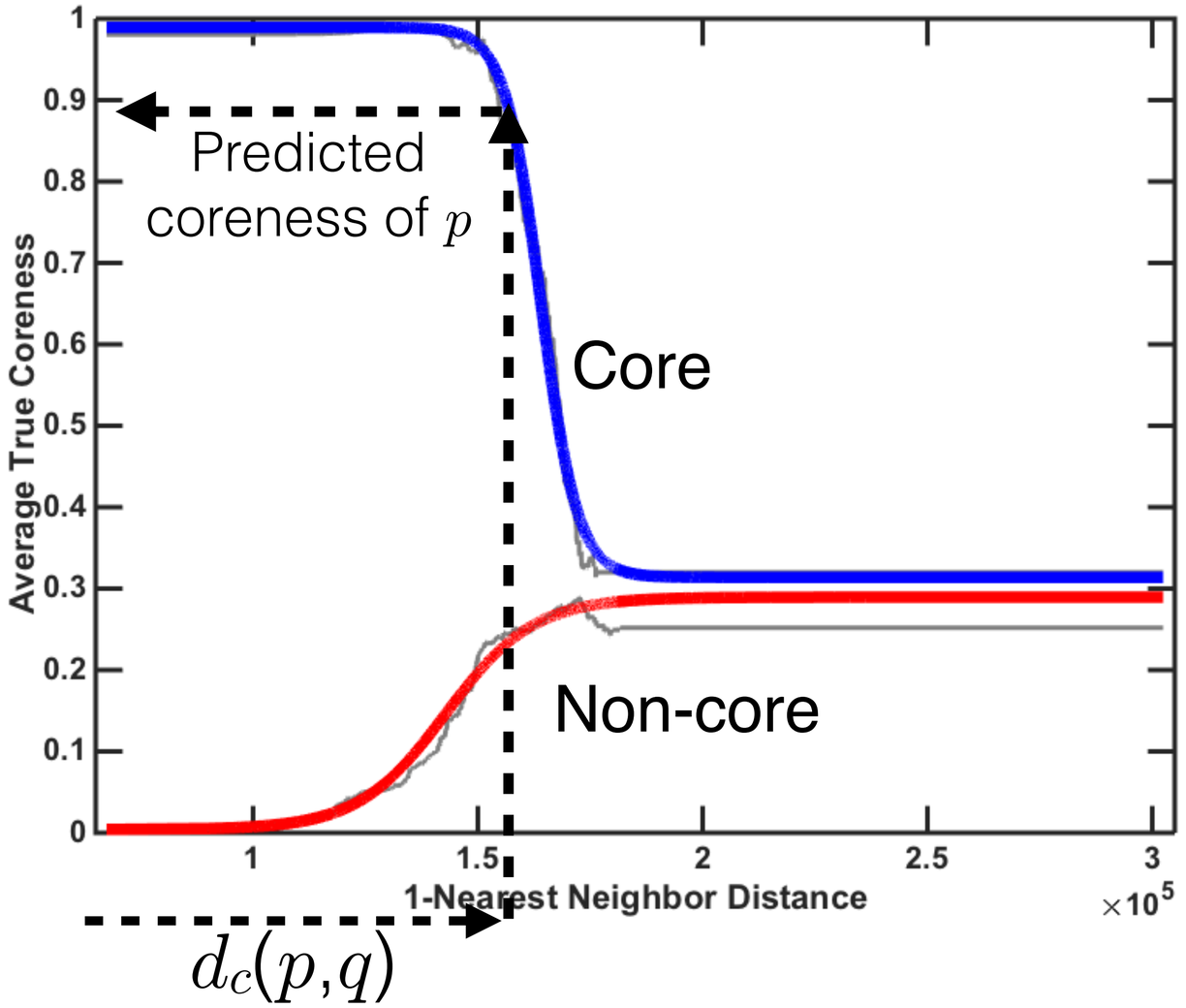
Classes “NCN” and “CNC” appeared very rarely in examples

Predicting column coreness



Predicting column coreness

Transform distance into a coreness based neighbor's class



The distance function

The distance between two points

- measures the difference in window composition
- as a weighted sum of state-pair frequencies
- on corresponding columns of the two windows
- is specific to each window class

$$d(V,W) \;=: \sum_{-w \le i \le +w} \; \sum_{p,q \in Q} V_i(p)\, W_i(q)\, \sigma_i(p,q)$$

The distance function

The distance between two points

- measures the difference in window composition
- as a weighted sum of state-pair frequencies
- on corresponding columns of the two windows
- is specific to each window class

$$d(V,W) \;=: \sum_{-w \le i \le +w} \; \sum_{p,q \in Q} V_i(p)\, W_i(q)\, \sigma_i(p,q)$$

each column in a window

The distance function

The distance between two points

- measures the difference in window composition
- as a weighted sum of state-pair frequencies
- on corresponding columns of the two windows
- is specific to each window class

$$d(V,W) \;=: \sum_{-w \le i \le +w} \; \sum_{p,q \in Q} V_i(p)\, W_i(q)\, \sigma_i(p,q)$$

each *pair* of states

The distance function

The distance between two points

- measures the difference in window composition
- as a weighted sum of state-pair frequencies
- on corresponding columns of the two windows
- is specific to each window class

$$d(V,W) \;=: \sum_{-w \le i \le +w} \; \sum_{p,q \in Q} V_i(p)\, W_i(q)\, \sigma_i(p,q)$$

state pair frequency

The distance function

The distance between two points

- measures the difference in window composition
- as a weighted sum of state-pair frequencies
- on corresponding columns of the two windows
- is specific to each window class

$$d(V,W) \;=: \sum_{-w \le i \le +w} \; \sum_{p,q \in Q} V_i(p)\, W_i(q)\, \sigma_i(p,q)$$

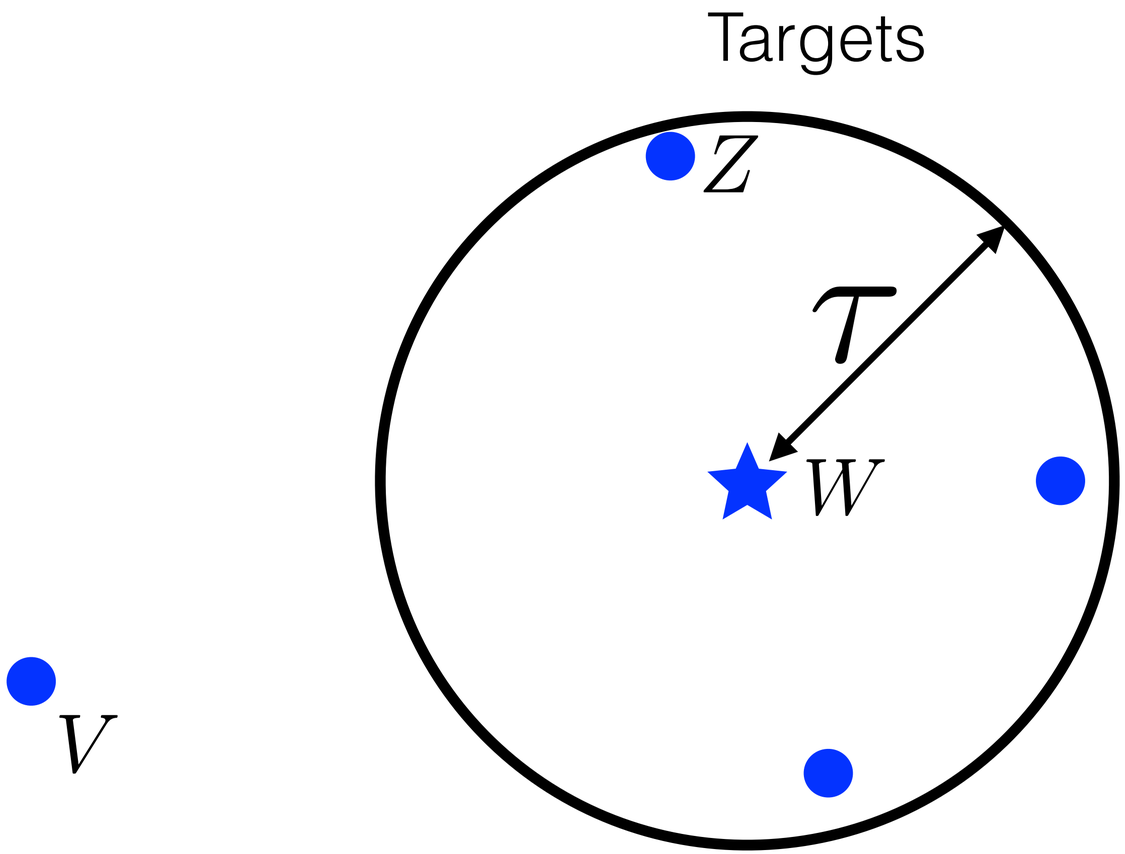
substitution of
the state pair

Learning the distance function

We learn the distance functions using

- a set of labeled training points representing all classes
- a touchstone of labeled examples from each structured class.

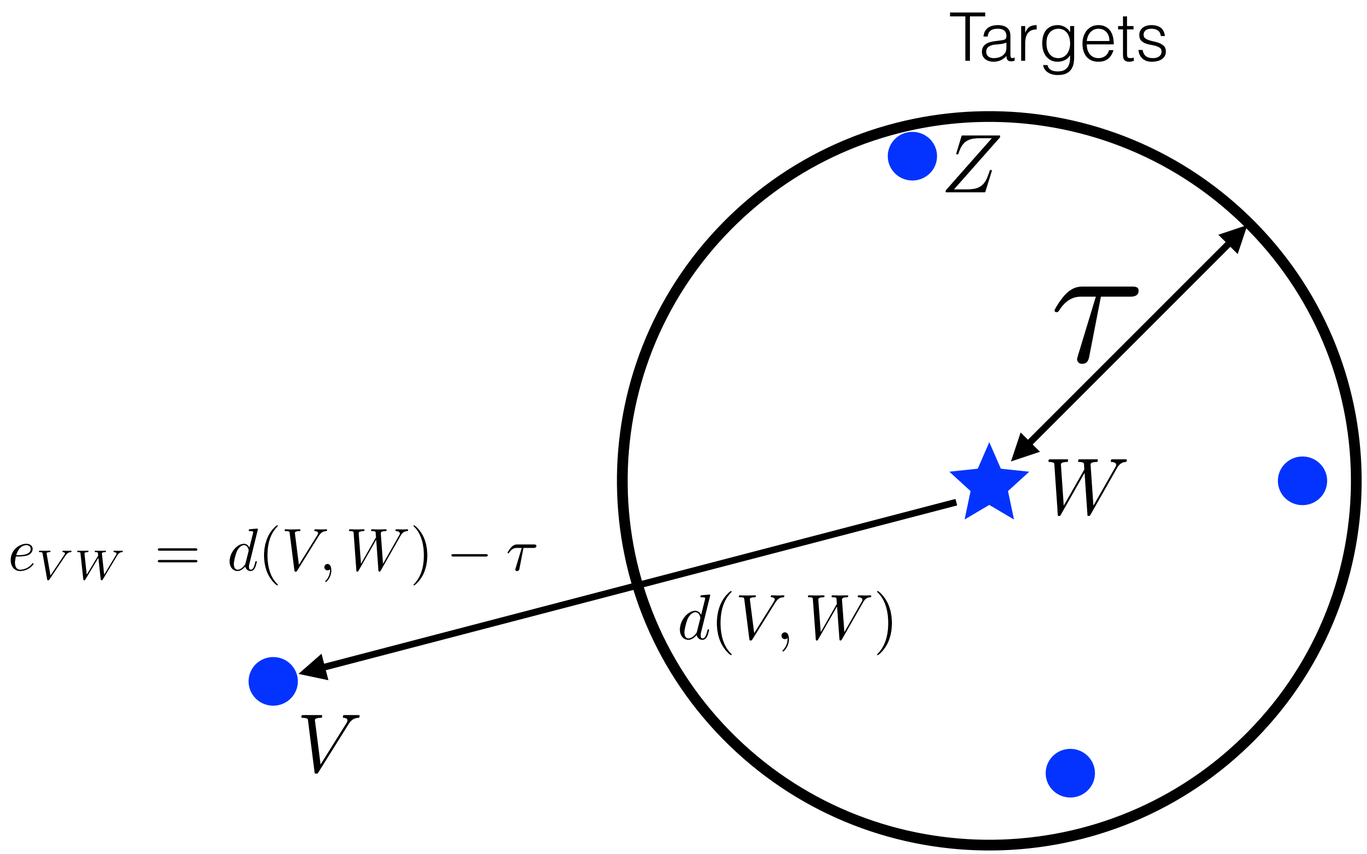
Learning the distance function



Targets are from the same window class

We want to pull targets close to the example

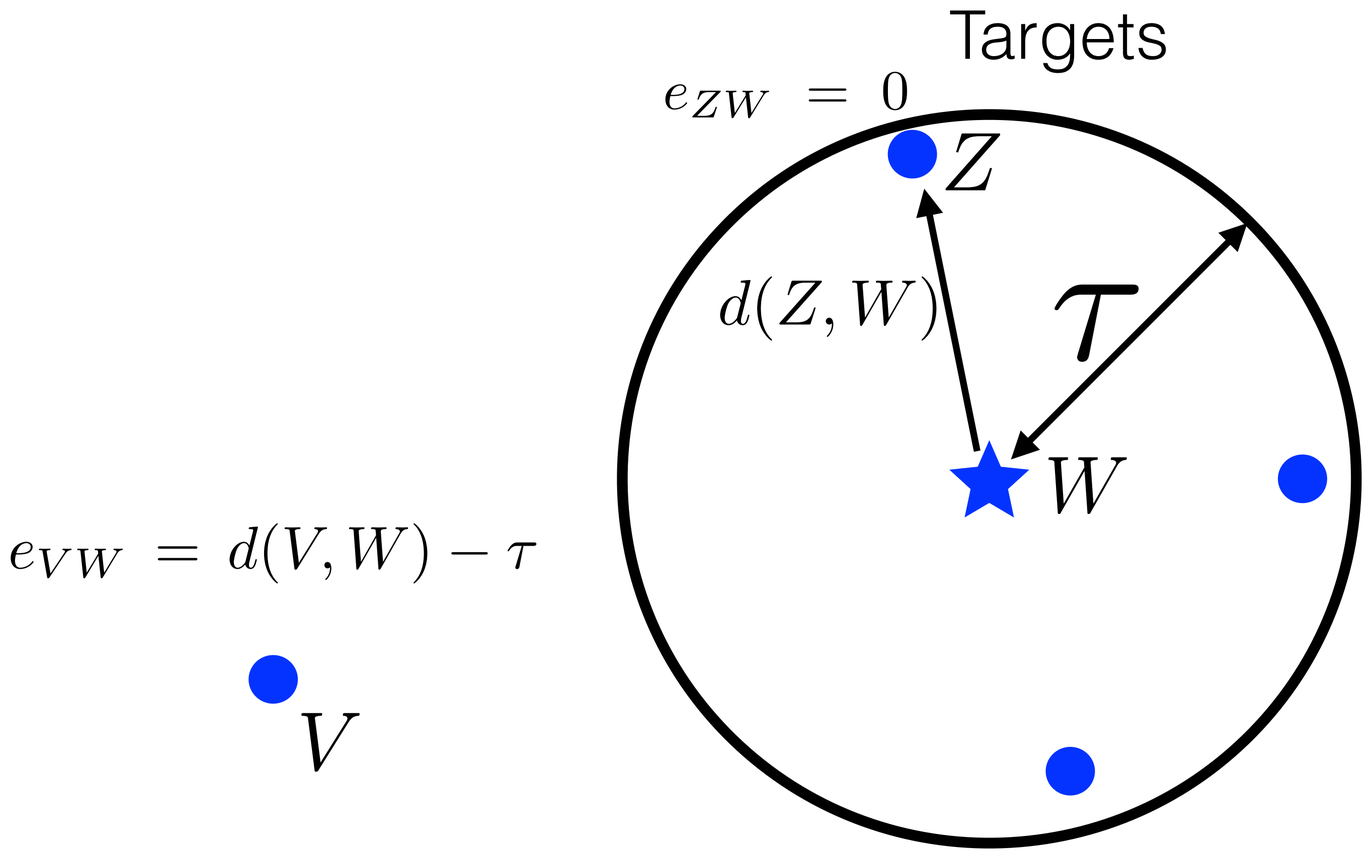
Learning the distance function



Targets are from the same window class

We want to pull targets close to the example

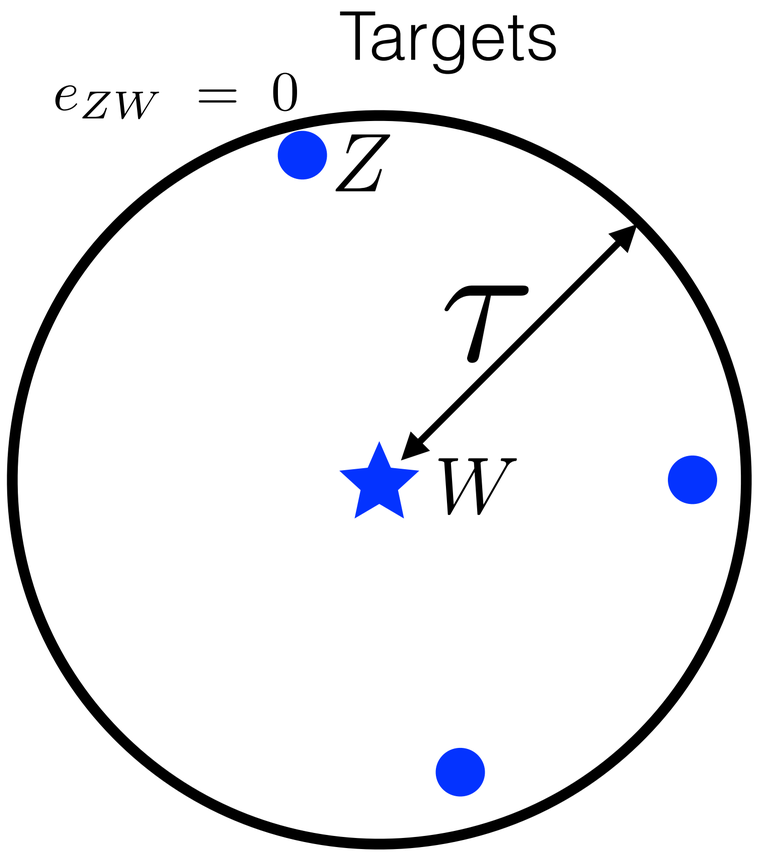
Learning the distance function



Targets are from the same window class

We want to pull targets close to the example

Learning the distance function

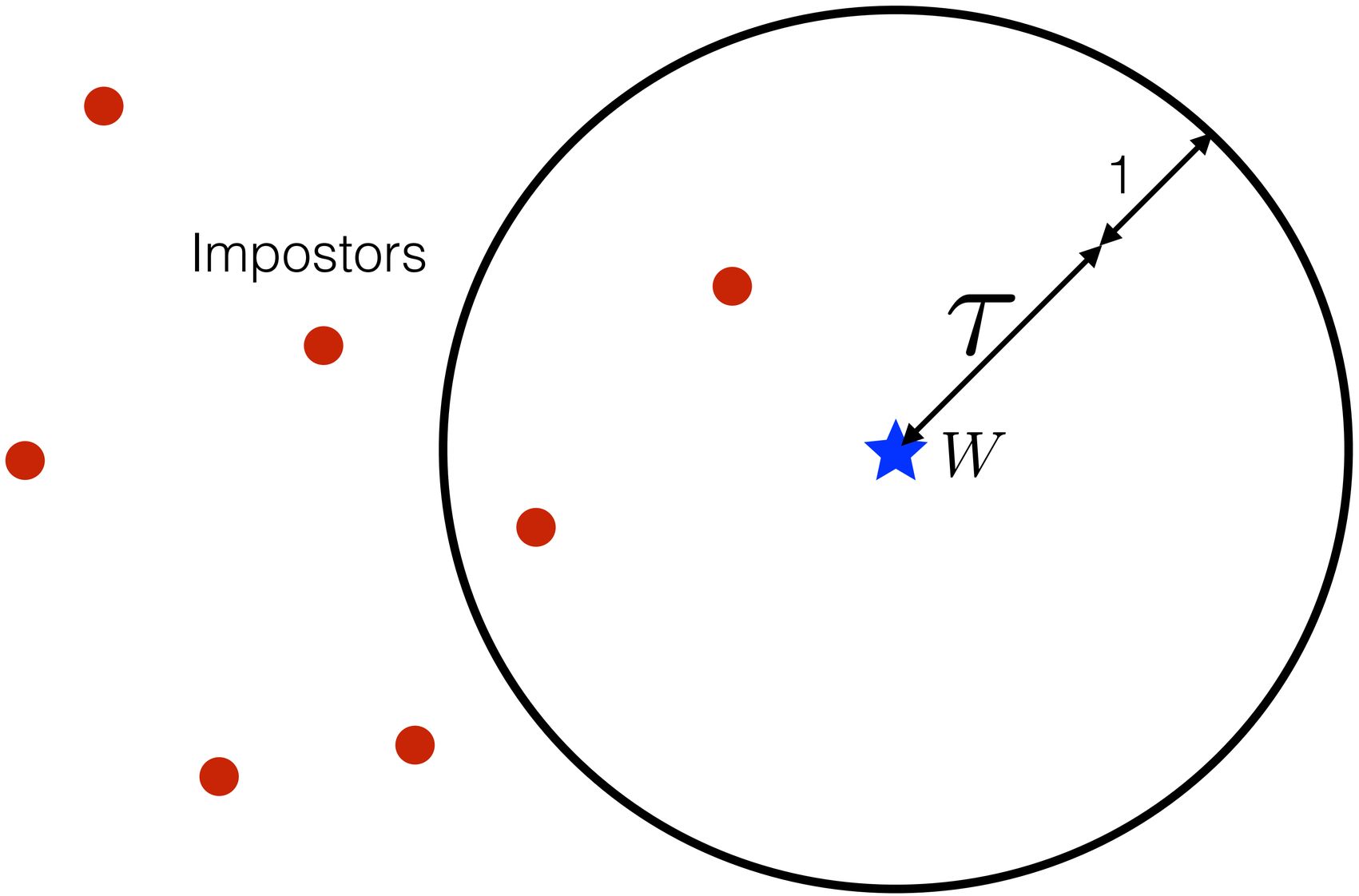


Targets are from the same window class

We want to pull targets close to the example

Target error for example W is $e_W = \dfrac{e_{VW} + e_{ZW} + \ldots}{4}$

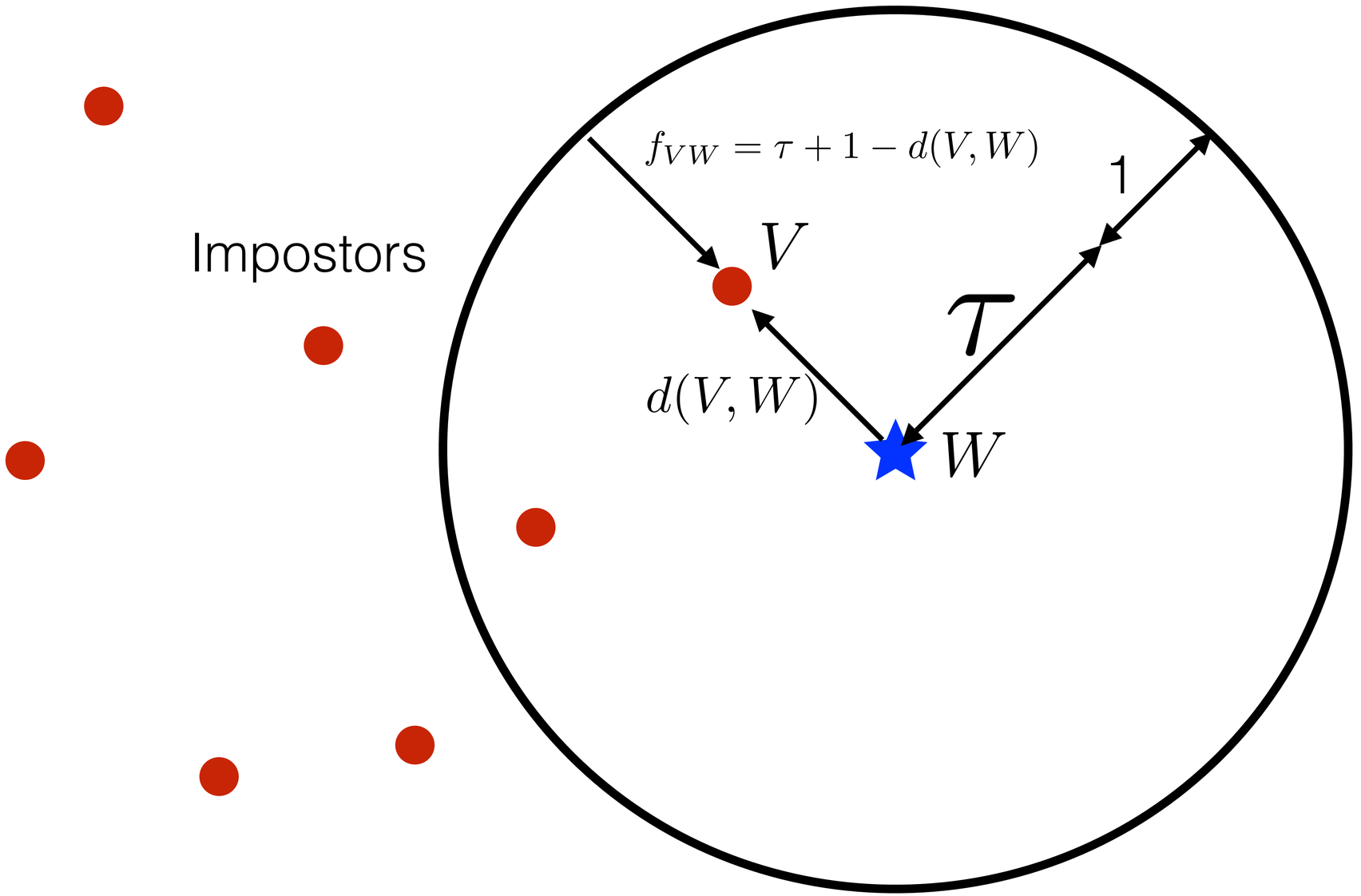
Learning the distance function



Impostors are from the different window classes than W

We want to push targets away from the example

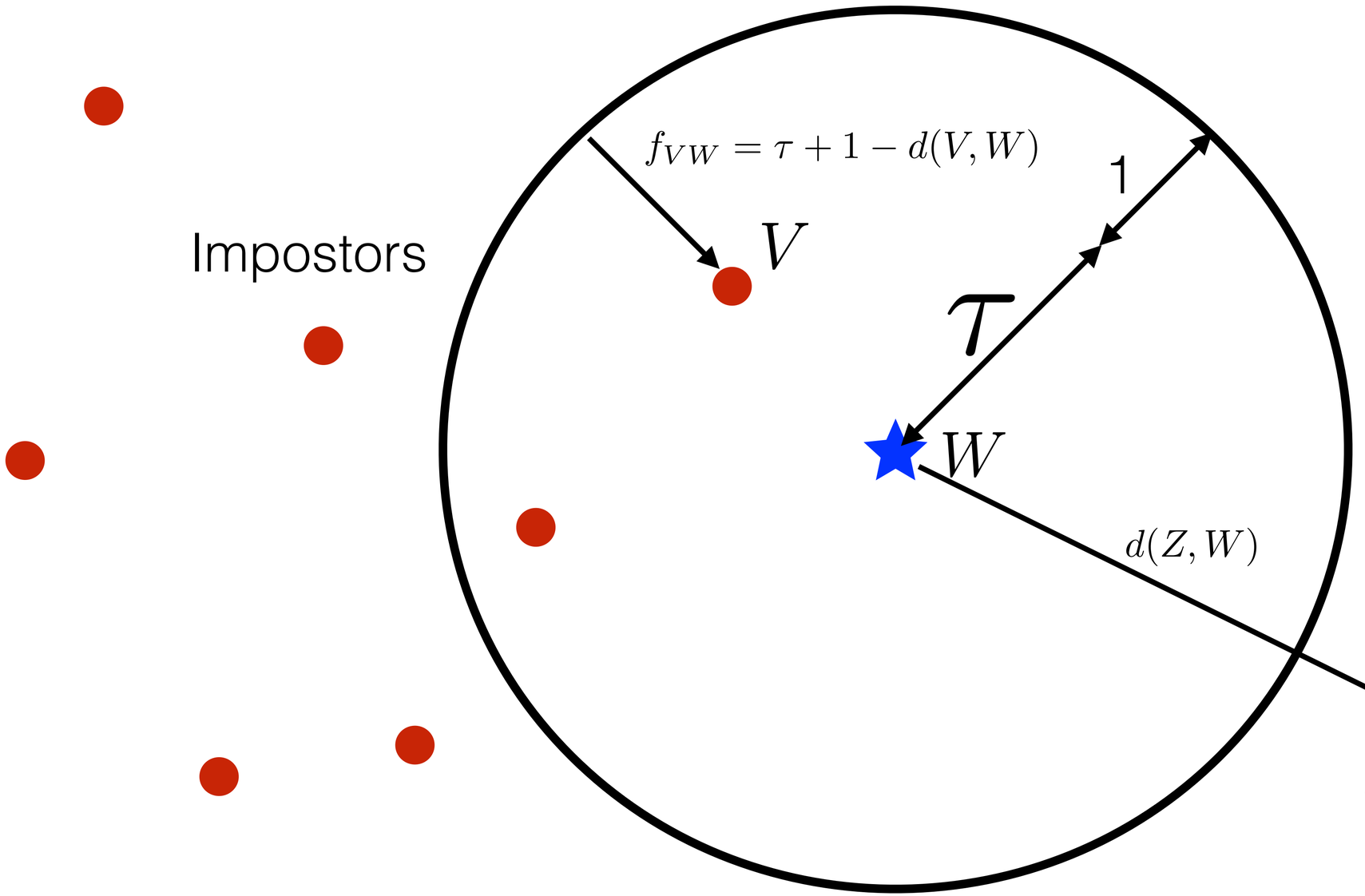
Learning the distance function



Impostors are from the different window classes than W

We want to push targets away from the example

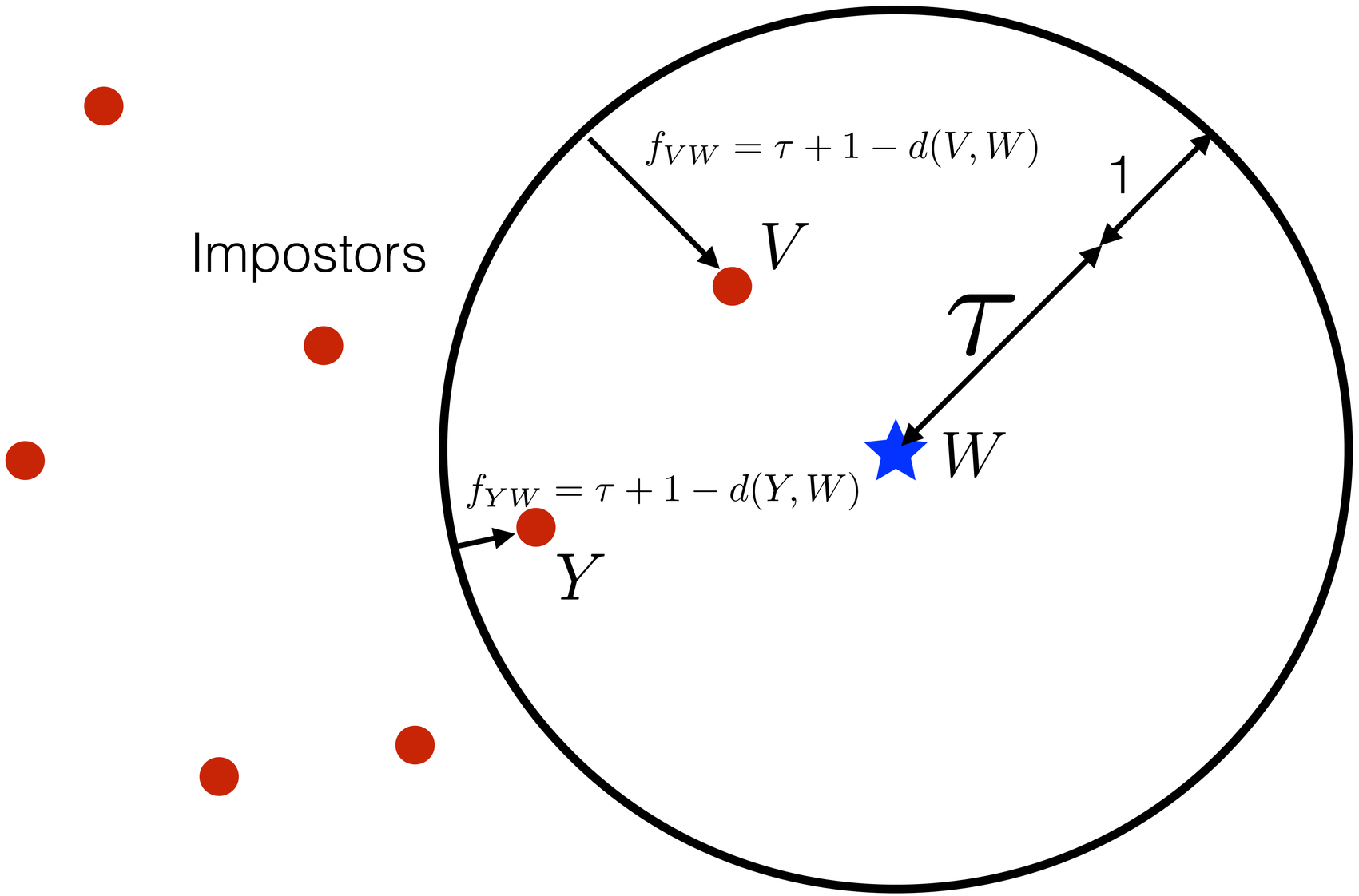
Learning the distance function



Impostors are from the different window classes than W

We want to push targets away from the example

Learning the distance function

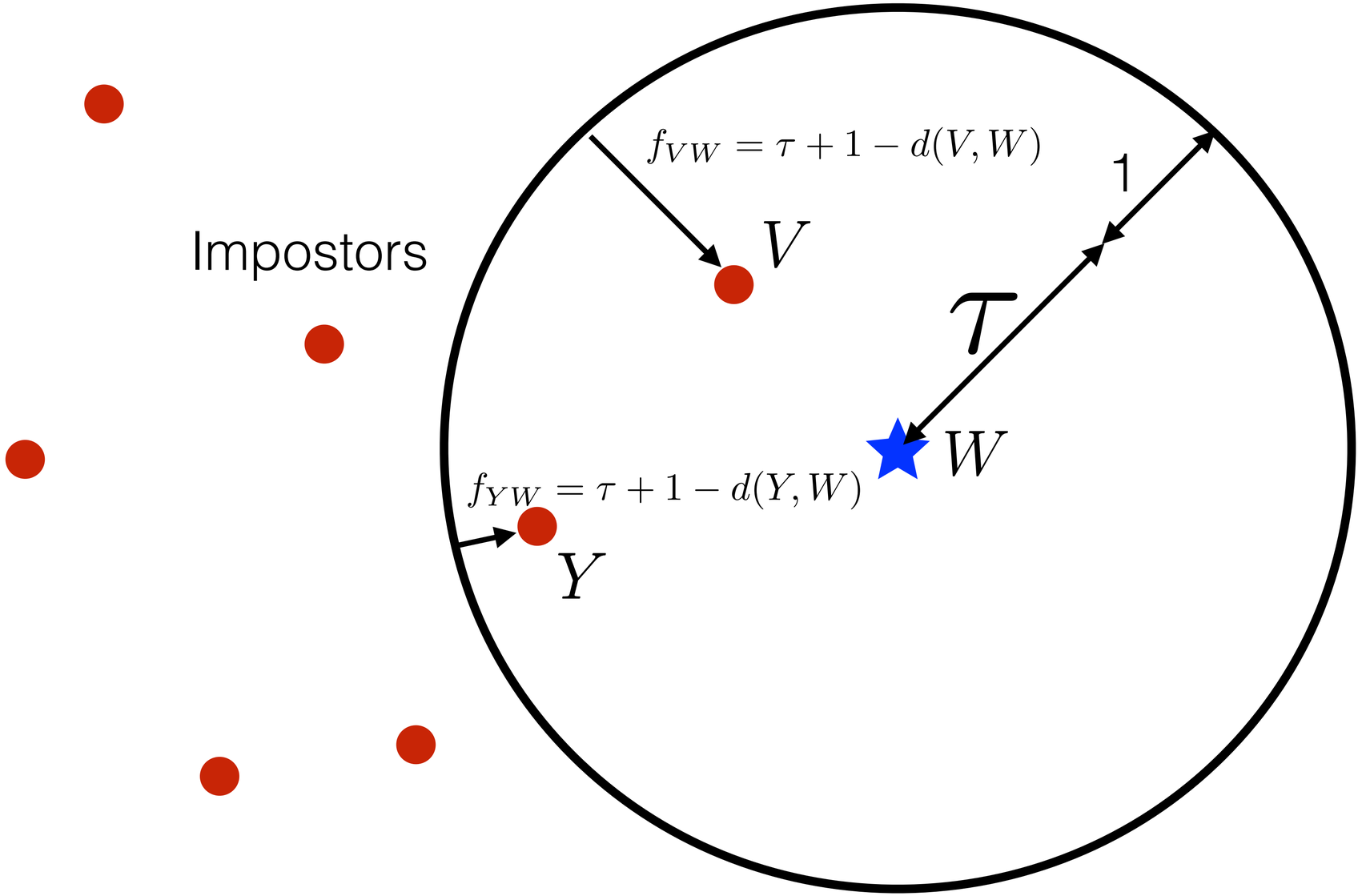


Impostors are from the different window classes than W

We want to push targets away from the example

Z $f_{ZW} = 0$

Learning the distance function



Impostors are from the different window classes than W

We want to push targets away from the example

Z $f_{ZW} = 0$

Impostor error for example W is $f_W = \max(f_{VW}, f_{ZW}, f_{YW}, ...)$

Learning the distance function

We find the distance function for all classes at once by solving a linear program which minimizes

$$\alpha \frac{1}{|\mathcal{C}|-1} \sum_{c \in \mathcal{C}-\{\perp\}} \frac{1}{|S_c|} \sum_{W \in S_c} e_W \quad + \quad (1-\alpha) \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|S_c|} \sum_{W \in S_c} f_W \, ,$$

Learning the distance function

We find the distance function for all classes at once by solving a linear program which minimizes

$$\alpha \frac{1}{|\mathcal{C}|-1} \sum_{c \in \mathcal{C}-\{\perp\}} \frac{1}{|S_c|} \sum_{W \in S_c} e_W + (1-\alpha) \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|S_c|} \sum_{W \in S_c} f_W \,,$$

minimize target error

minimize impostor error

controls the influence of target error vs impostor error

Learning the distance function

We find the distance function for all classes at once by solving a linear program which minimizes

$$\alpha \frac{1}{|\mathcal{C}|-1} \sum_{c \in \mathcal{C}-\{\perp\}} \frac{1}{|S_c|} \sum_{W \in S_c} e_W \quad + \quad (1-\alpha) \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|S_c|} \sum_{W \in S_c} f_W \,,$$

only structured classes have targets

Learning the distance function

We find the distance function for all classes at once by solving a linear program which minimizes

$$\alpha \frac{1}{|\mathcal{C}|-1} \sum_{c \in \mathcal{C}-\{\perp\}} \frac{1}{|S_c|} \sum_{W \in S_c} e_W \quad + \quad (1-\alpha) \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|S_c|} \sum_{W \in S_c} f_W \,,$$

for each example of the class c

Learning the distance function

We find the distance function for all classes at once by solving a linear program which minimizes

$$\alpha \frac{1}{|\mathcal{C}|-1} \sum_{c \in \mathcal{C}-\{\perp\}} \frac{1}{|S_c|} \sum_{W \in S_c} e_W + (1-\alpha) \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|S_c|} \sum_{W \in S_c} f_W \,,$$

target error is averaged

Learning the distance function

We find the distance function for all classes at once by solving a linear program which minimizes

$$\alpha \frac{1}{|\mathcal{C}|-1} \sum_{c \in \mathcal{C}-\{\perp\}} \frac{1}{|S_c|} \sum_{W \in S_c} e_W \quad + \quad (1-\alpha) \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|S_c|} \sum_{W \in S_c} f_W \,,$$

unstructured examples only have impostors

Learning the distance function

We find the distance function for all classes at once by solving a linear program which minimizes

$$\alpha \frac{1}{|\mathcal{C}|-1} \sum_{c \in \mathcal{C}-\{\perp\}} \frac{1}{|S_c|} \sum_{W \in S_c} e_W \quad + \quad (1-\alpha) \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|S_c|} \sum_{W \in S_c} f_W \,,$$

for all examples of that class

Learning the distance function

We find the distance function for all classes at once by solving a linear program which minimizes

$$\alpha \frac{1}{|\mathcal{C}|-1} \sum_{c \in \mathcal{C}-\{\perp\}} \frac{1}{|S_c|} \sum_{W \in S_c} e_W \quad + \quad (1-\alpha) \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|S_c|} \sum_{W \in S_c} f_W,$$

impostor error is a maximum

Learning the distance function

We find the distance function for all classes at once by solving a linear program which minimizes

$$\alpha \frac{1}{|\mathcal{C}|-1} \sum_{c \in \mathcal{C}-\{\perp\}} \frac{1}{|S_c|} \sum_{W \in S_c} e_W \quad + \quad (1-\alpha) \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|S_c|} \sum_{W \in S_c} f_W \,,$$

We include constraints to ensure that the distances are symmetric, that self distance is smaller than any other and satisfy the triangle inequality.

Augmenting existing features

We weight columns by predicted coreness when calculating

- Amino Acid Identity
- Average Replacement Score
- Secondary Structure Identity
- Secondary Structure Support
- Secondary Structure Blockiness

Predicted Alignment Coreness

We create a new feature function using predicted coreness

- Similar to the total column score (TC-Score)
- Threshold the coreness value to make a binary labeling
- Normalize by an estimate of the number of core columns

Predicted Alignment Coreness

The normalizer is a weighted sum of products of up to 3 of

- aggregate length of sequences in the input
- ratio of longest common subsequence and aggregate length
- ratio of maximum difference in length and aggregate length

We learn the normalizer coefficients using linear programming

Experimental results

We evaluate the accuracy of adaptive local realignment

- with the `Opal` aligner and `Facet` estimator,
- on over 800 benchmarks from `BENCH` and `PALI`,
- using 12-fold cross-validation.

Experimental results

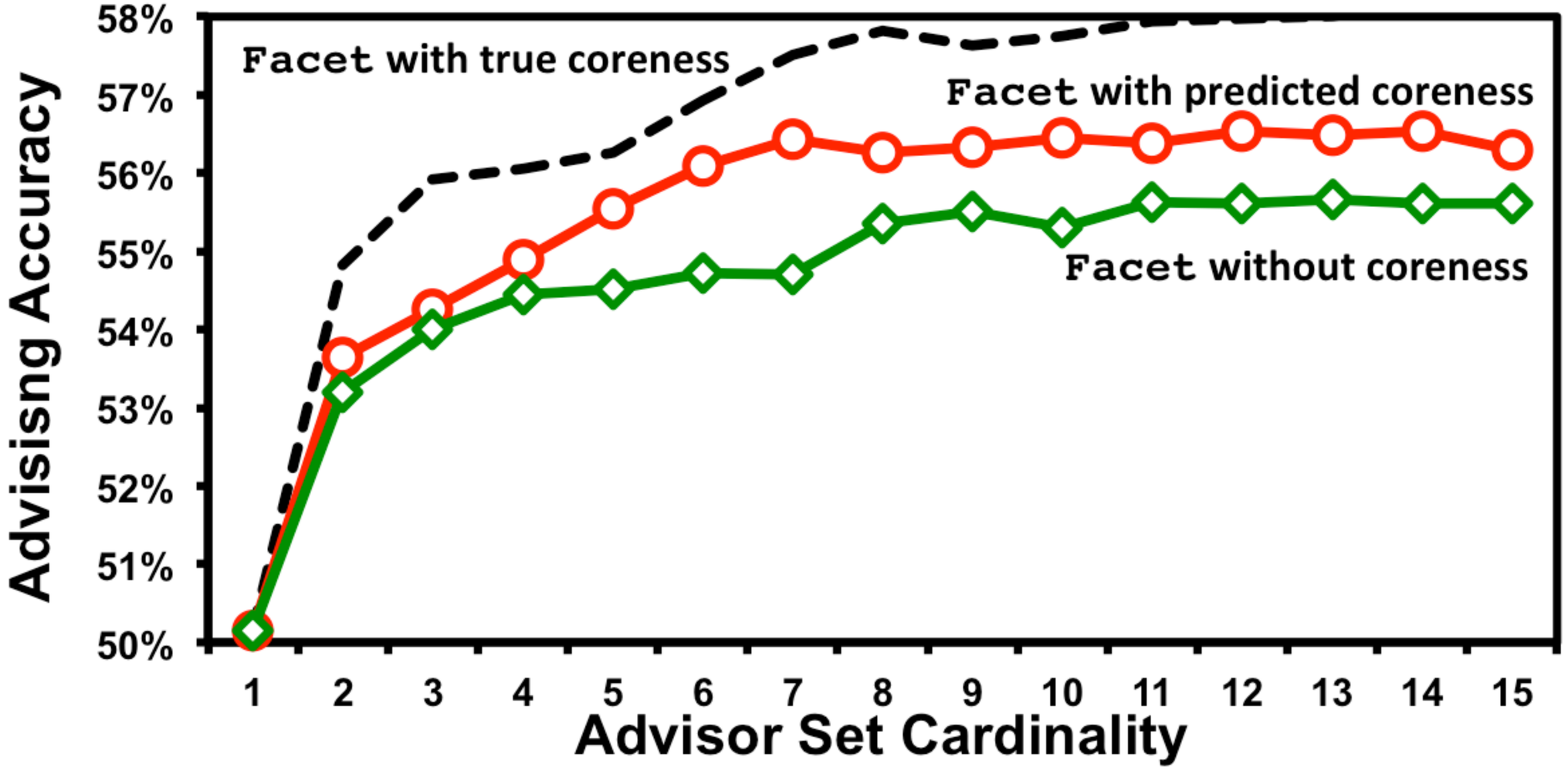
We correct for the bias in over-representation of easy-to-align benchmarks.

- The difficulty of a benchmark is its accuracy under the default parameter setting.
- Split the range of difficulties [0,1] into 10 bins.
- Report advisor accuracy uniformly averaged across bins.

The typical average accuracy is close to 50%.

Experimental results

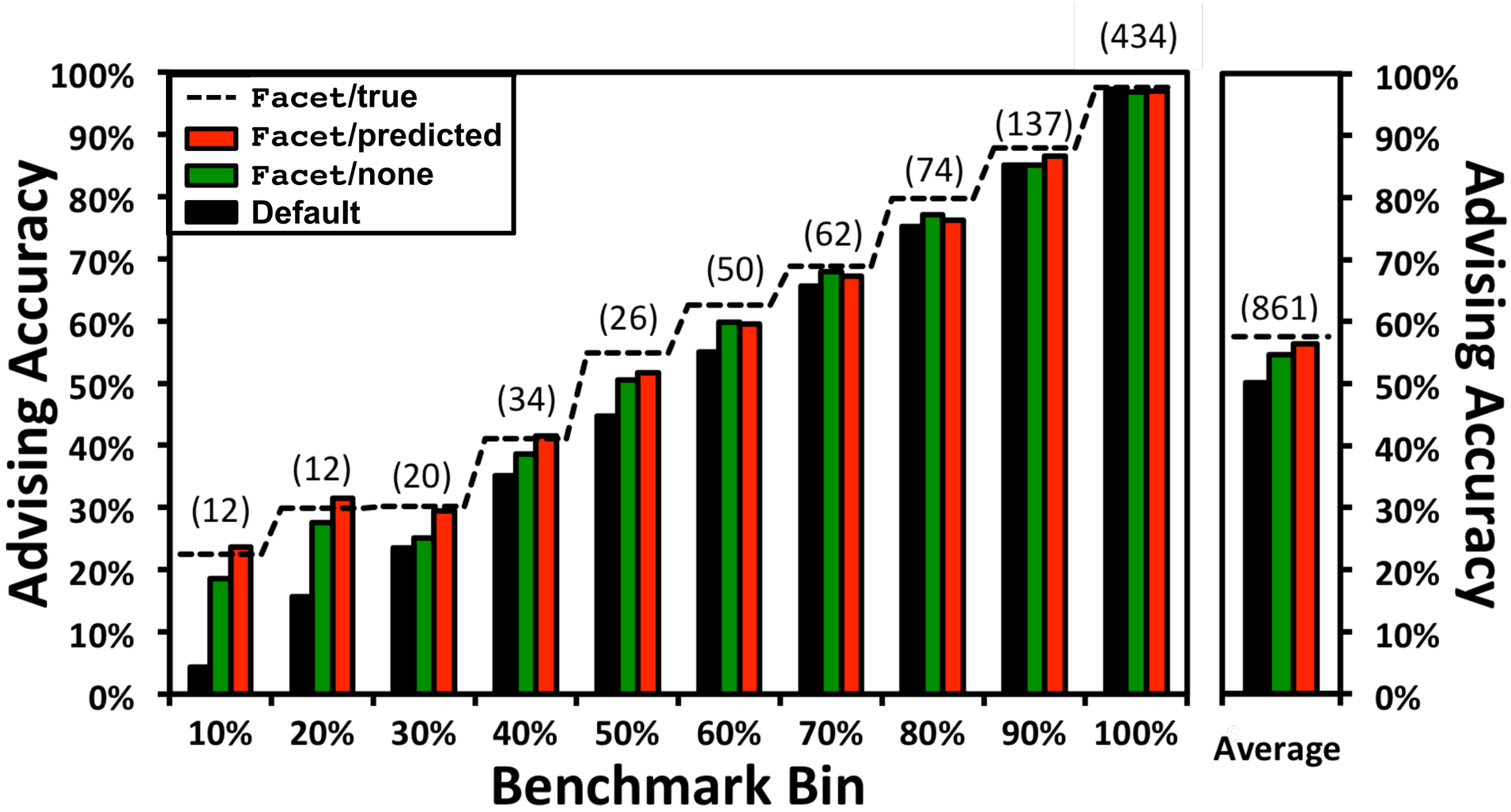
`Facet` modified with coreness versus set cardinality



Predicting coreness boosts accuracy for `Facet`

Experimental results

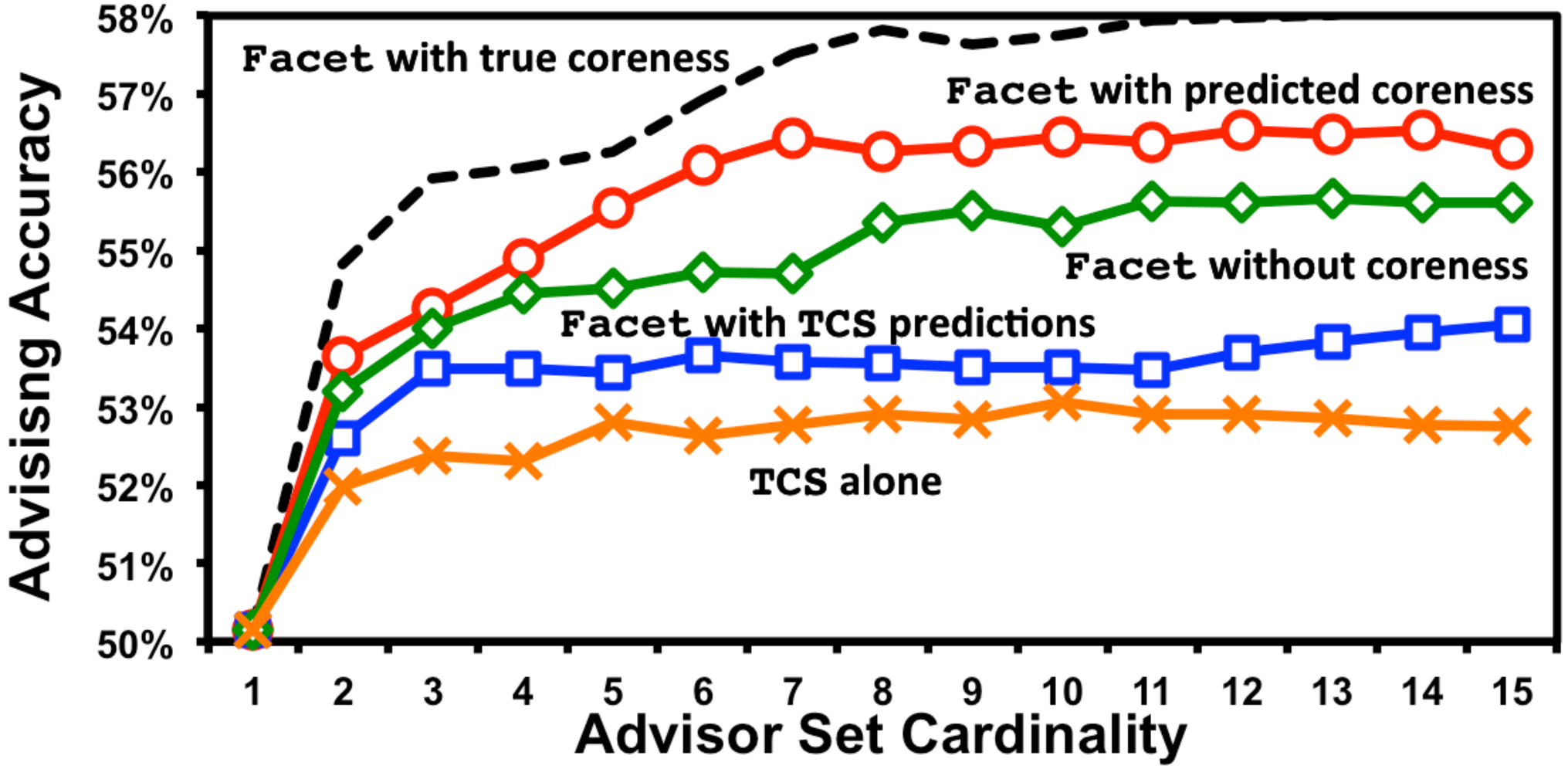
`Facet` modified with coreness within difficulty bins



Predicting coreness boosts accuracy for by more than 5%.

Experimental results

`Facet` modified with coreness versus greedy set cardinality



Predicting coreness boosts accuracy for `Facet`

Accuracy estimation software

Available for download:

- `Facet` accuracy estimator
- `Opal` aligner with parameter advising
- Parameter sets for advising

`facet.cs.arizona.edu`

Acknowledgments

People

William Pearson

Travis Wheeler

Funding

- University of Arizona
  NSF IGERT in Genomics
  Grant DGE-0654435

- NSF Grant IIS-1217886