IBM Watson Seminar

String Kernels and Cluster Kernels for Protein Classification

Christina Leslie
Department of Computer Science
Columbia University

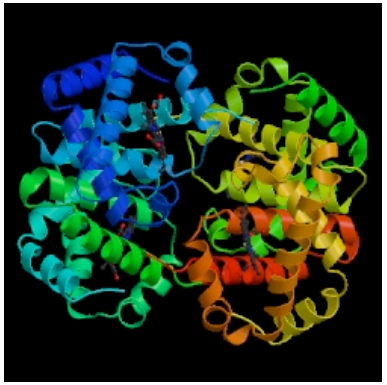
Protein Sequence Classification

- Protein represented by sequence of amino acids, encoded by a gene
- Easy to sequence proteins, difficult to obtain structure
- *Classification Problem:* Learn how to classify protein sequence data into families and superfamilies defined by structure/function relationships

Sequence

VLSPADKTNVKAAWGKVGAHAGEYGAEALER
MFLSFPTTKTYFPHFDLSHGSAQVKGHGKKV
ADALTNAVAHVDDMPNALSALSDLHAHKLRV
DPVNFKLLSHCLLVTLAAHLPAEFTPAVHAS
LDKFLASVSTVLTSKYR

3D Structure



Class

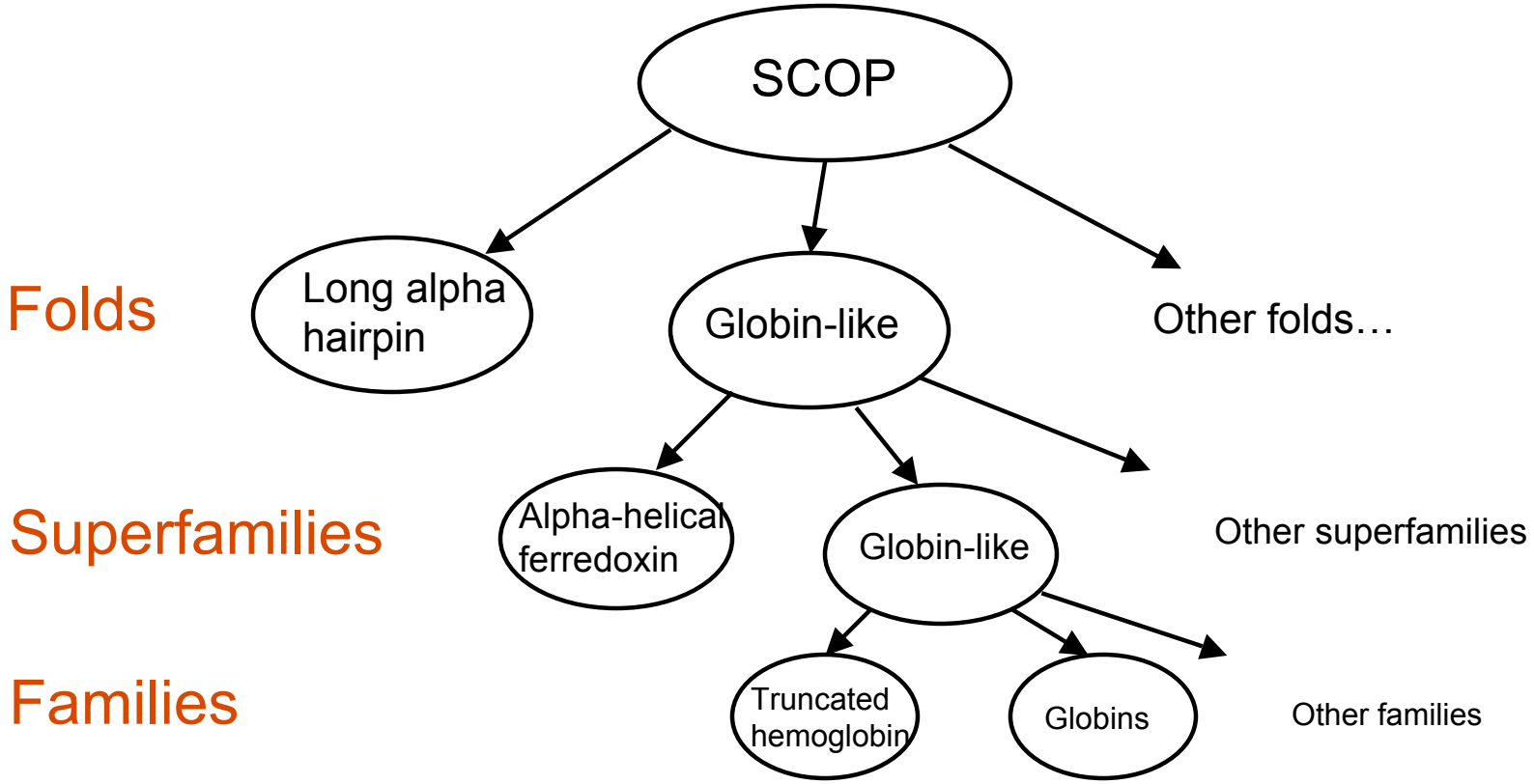
Globin family

Globin-like superfamily

Function

Oxygen transport

Structural Hierarchy of Proteins

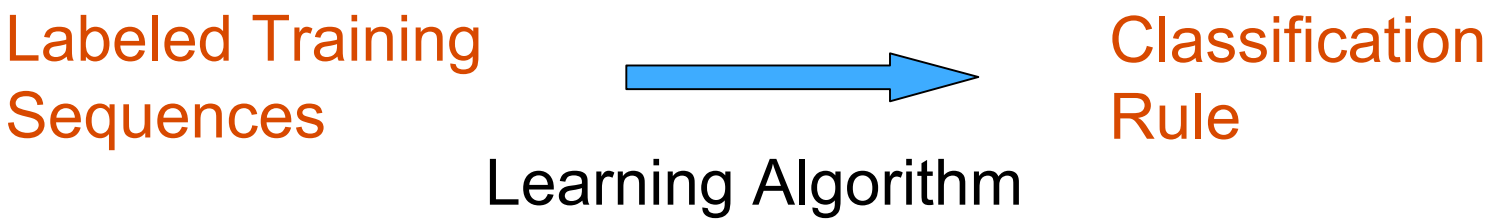


- *Remote homologs:* sequences that belong to the same superfamily but not the same family – remote evolutionary relationship
- Structure and function conserved, though sequence similarity can be low

Learning Problem

- Use *discriminative supervised learning* approach to classify protein sequences into structurally-related families, superfamilies
- Labeled training data: proteins with *known structure*, positive (+) if example belongs to a family or superfamily, negative (-) otherwise
- Focus on *remote homology detection*

Approach: Support vector machines (SVMs) with new *string kernels* based on inexact string matching



Beyond Classification: Protein Ranking

- *Ranking problem:* given query protein sequence, return ranked list of similar proteins from sequence database
- Limitations of classification framework
  - Small amount of labeled data (proteins with known structure), huge unlabeled databases
  - Missing classes: undiscovered structures
- Good local similarity scores, based on heuristic alignment: BLAST, PSI-BLAST

Approach: Use new *semi-supervised learning* methods – training on labeled and unlabeled data – to improve ranking performance

Outline

1. Protein classification: Mismatch kernel
   - SVMs and kernel methods
   - Inexact matching through mismatches
   - Efficient kernel computation, fast prediction
2. Experimental results on SCOP dataset
3. Other models for inexact matching
   - Kernels from gaps, substitutions, wildcards
4. Cluster kernels: Semi-supervised methods
   - Using unlabeled data to change the kernel

Support Vector Machine (SVM) Classifiers

- Training examples mapped to (usually high-dimensional) feature space by a *feature map*

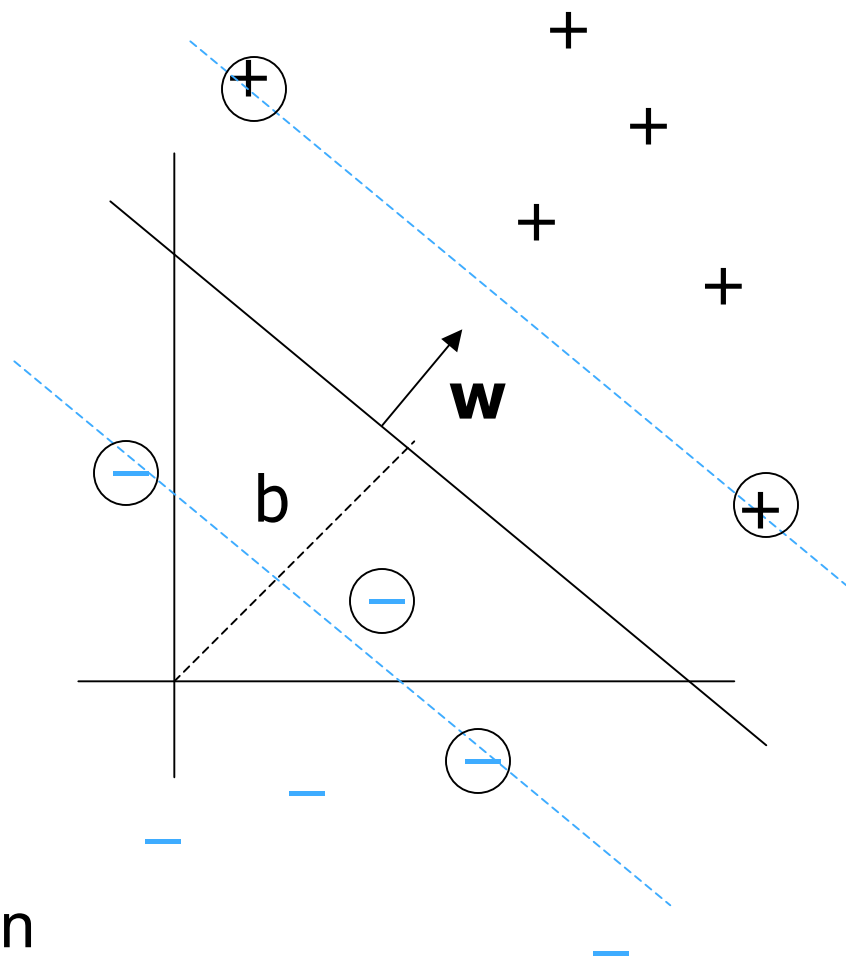
$$F(x) = (F_1(x), \ldots, F_N(x))$$

- Learn linear classifier in feature space

$$f(x) = \langle \mathbf{w}, F(x) \rangle + b$$

by solving optimization problem: trade-off between maximizing *geometric margin* and minimizing margin violations

- Large margin gives good generalization performance, even in high dimensions



Kernels for Discrete Objects

- Kernel trick: To train an SVM, can use *kernel* rather than explicit feature map
- Can define kernels for sequences, graphs, other *discrete* objects:

$$\{ \text{sequences} \} \xrightarrow{F} R^N$$

For sequences x, y, feature map F, kernel value is inner product in feature space

$$K(x, y) = \langle F(x), F(y) \rangle$$

- Original string kernels [Watkins, Haussler, later Lodhi *et al*.] require quadratic time in sequence length, $O(|x|\,|y|)$, to compute each kernel value $K(x, y)$

String Kernels for Biosequences

- We'll define new fast *string kernels* for biological sequence data
  - Biologically-inspired underlying feature map
  - Kernels scale linearly with sequence length, $O(c_K(|x| + |y|))$ to compute
  - Protein classification performance competitive with best available methods
  - Mismatches for *inexact sequence matching* (other models later)

Spectrum-based Feature Map

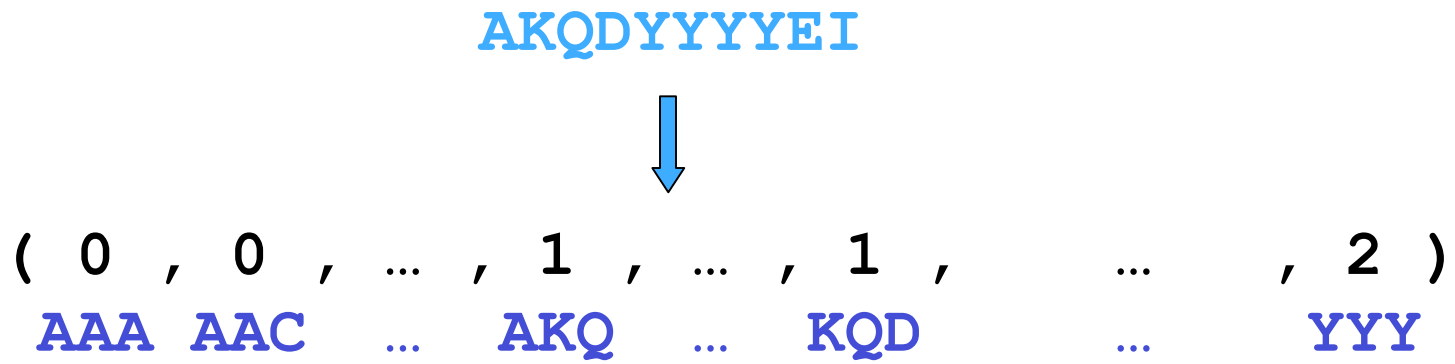
- Idea: feature map based on *spectrum* of a sequence
  - The k-spectrum of a sequence is the set of all k-length contiguous subsequences that it contains
  - Feature map is indexed by all possible k-length subsequences ("k-mers") from the alphabet of amino acids
  - Dimension of feature space = $|\Sigma|^k$ ($|\Sigma| = 20$ for amino acids)

k-Spectrum Feature Map

- Feature map for k-spectrum with no mismatches:

  For sequence x, $F_{(k)}(x) = (F_t(x))_{\{\text{k-mers } t\}}$,

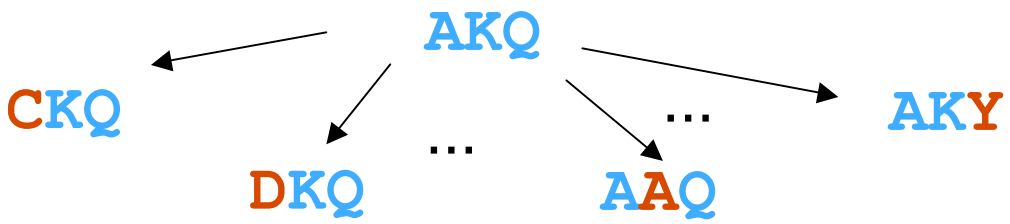
  where $F_t(x)$ = #occurrences of t in x



C. Leslie, E. Eskin, and W. Noble, *The Spectrum Kernel: A String Kernel for SVM Protein Classification.* Pacific Symposium on Biocomputing, 2002.

Inexact Matching through Mismatches

- For k-mer *s*, the *mismatch neighborhood* $N_{(k,m)}(s)$ is the set of all k-mers *t* within m mismatches from *s*
- Size of mismatch neighborhood is $O(|\Sigma|^m k^m)$



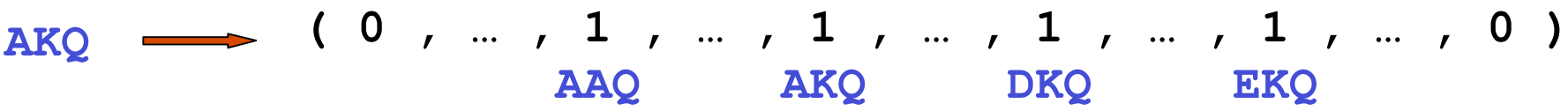
(k,m)-Mismatch Feature Map

- Feature map for k-spectrum, allowing m mismatches:

  For a k-mer s, $F_{(k,m)}(s) = (F_t(s))_{\{\text{k-mers } t\}}$,

  where $F_t(s) = 1$ if t is in neighborhood $N_{(k,m)}(s)$,

  $F_t(s) = 0$ otherwise

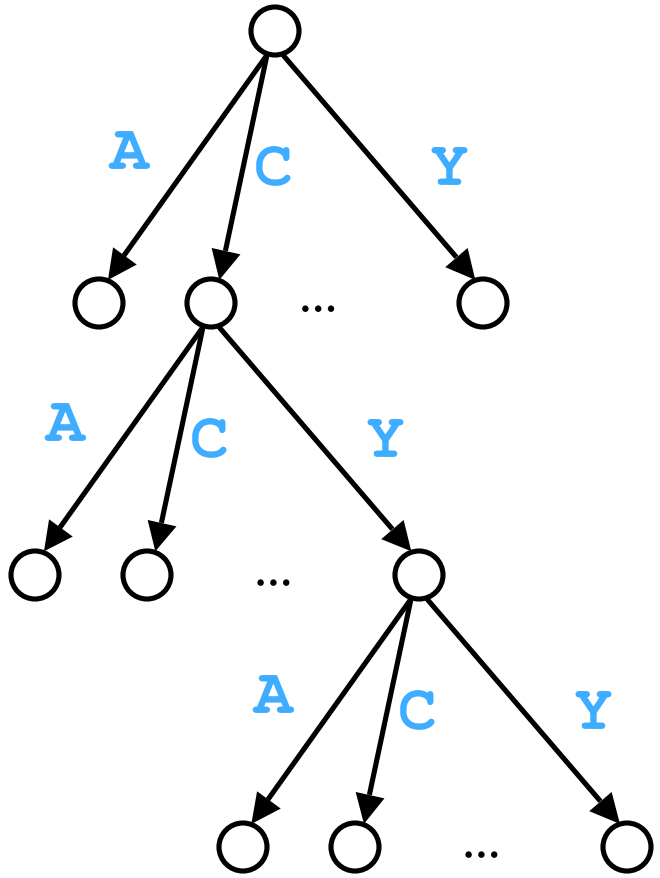


- Extend additively to longer sequences x by summing over all k-mers s in x

C. Leslie, E. Eskin, J. Weston and W. Noble, *Mismatch String Kernels for SVM Protein Classification.* Neural Information Processing Systems 2002.

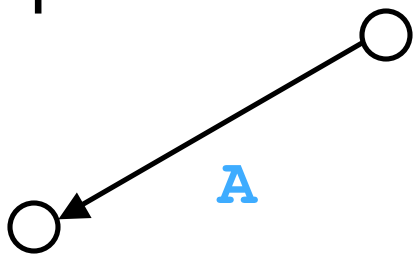
Computing the (k,m)-Mismatch Kernel

- Use *mismatch tree* to organize lexical traversal of all instances of k-mers (with mismatches) in the training data
  - Each path down to a leaf corresponds to a coordinate in feature map
  - Kernel values for all training sequences updated at each leaf node
  - Depth-first traversal can be accomplished with recursive function



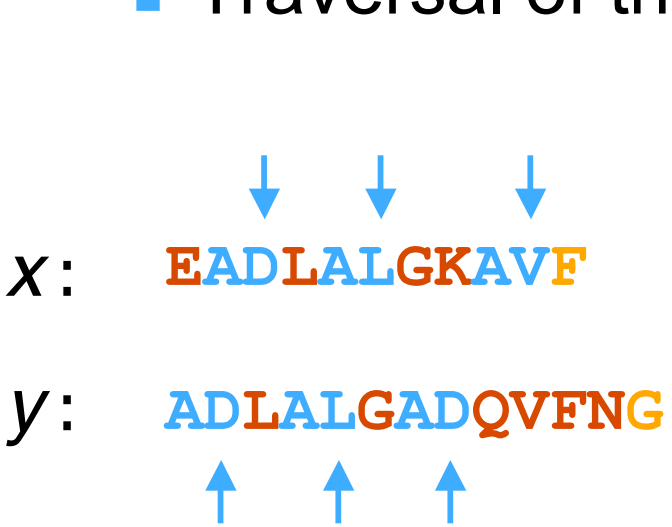
Computing the Kernel for Pair of Sequences

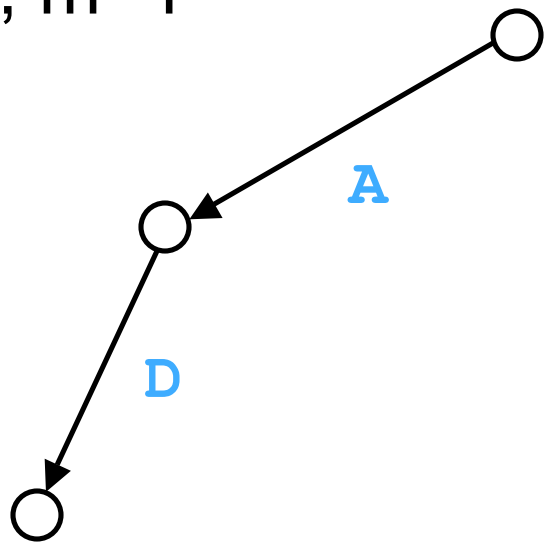
- Traversal of trie for k=3, m=1



Computing the Kernel for Pair of Sequences

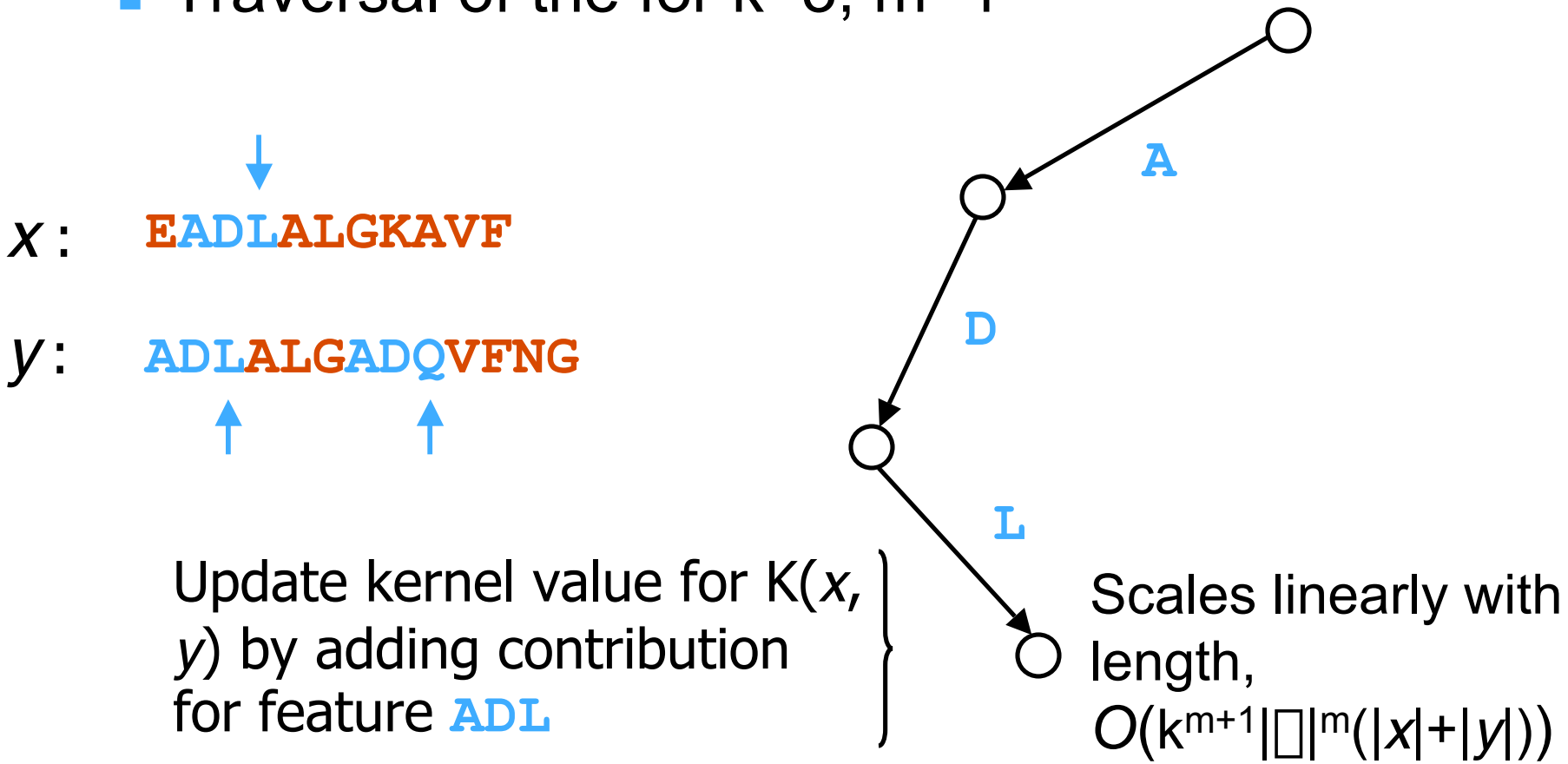
- Traversal of trie for k=3, m=1





Computing the Kernel for Pair of Sequences

- Traversal of trie for k=3, m=1



SVM Solution

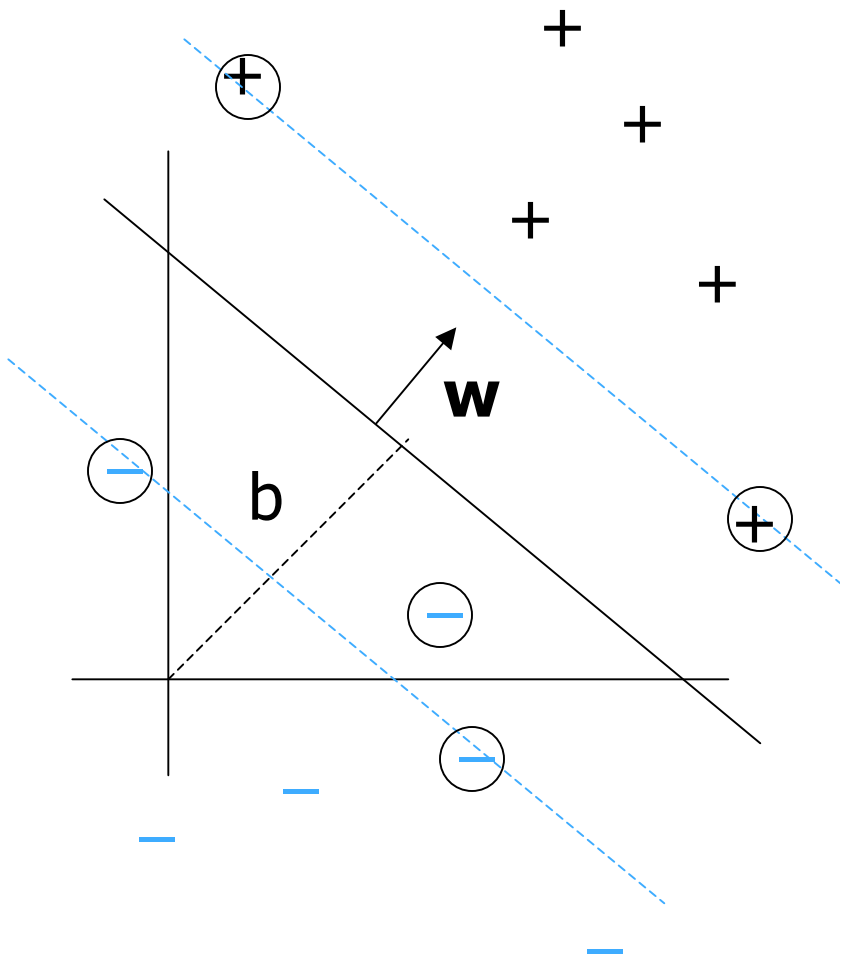
- Linear classifier defined in feature space by

  $$f(x) = \langle \mathbf{w}, F(x) \rangle + b$$

  where sign(f(*x*)) gives prediction
- SVM solution gives normal vector

  $$\mathbf{w} = \sum_i y_i \alpha_i F(x_i)$$

  as a linear combination of *support vectors*, involving weights α_i and labels y_i



Fast prediction

- SVM training determines subset of training sequences corresponding to *support vector sequences* and their weights: (x_i, α_i)
- Linear decision rule in feature space:

$$f(x) = \sum_i y_i \alpha_i \langle F(x_i), F(x) \rangle + b$$

- F(x) is sum of feature vectors F(s) for k-mers s in x

 Precompute per k-mer scores for classifier
 Test sequences can be classified in *linear time* via lookup of k-mers

Outline

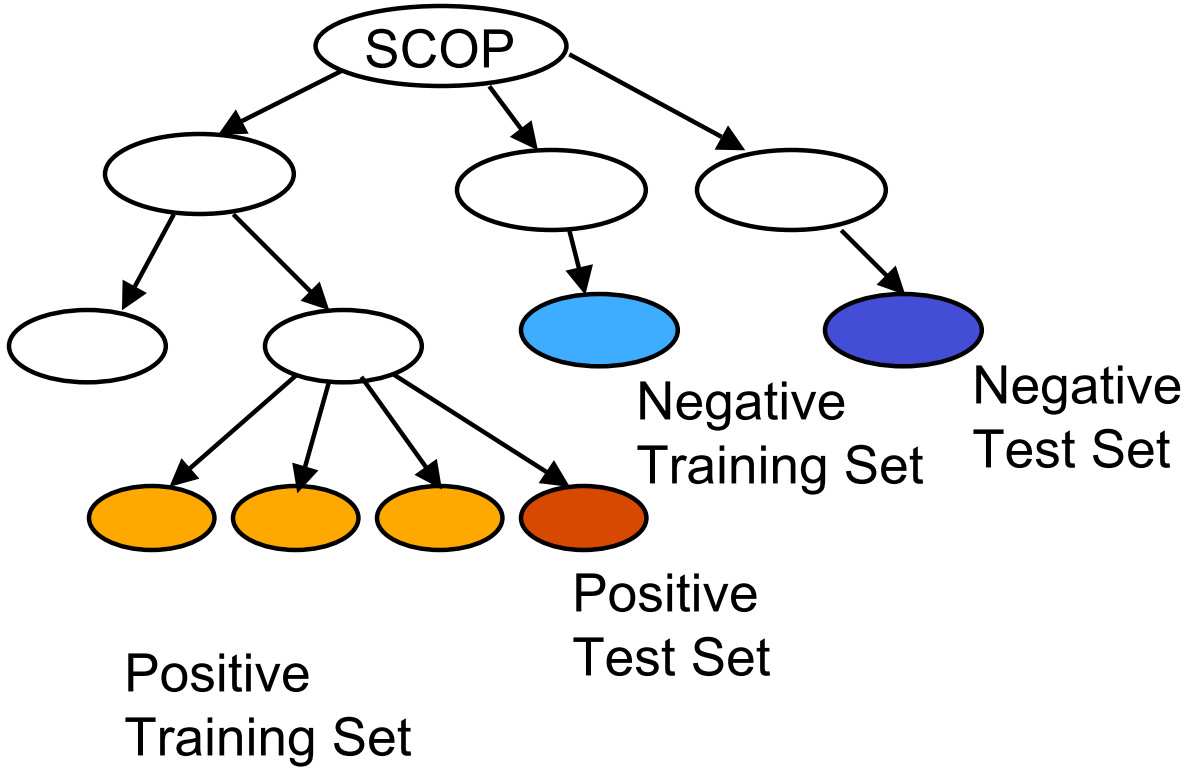
1. Protein classification: Mismatch kernel
   - SVMs and kernel methods
   - Inexact matching through mismatches
   - Efficient kernel computation, fast prediction
2. Experimental results on SCOP dataset
3. Other models for inexact matching
   - Kernels from gaps, substitutions, wildcards
4. Cluster kernels: Semi-supervised methods
   - Using unlabeled data to change the kernel

SCOP Experiments

Fold

Superfamily

Family

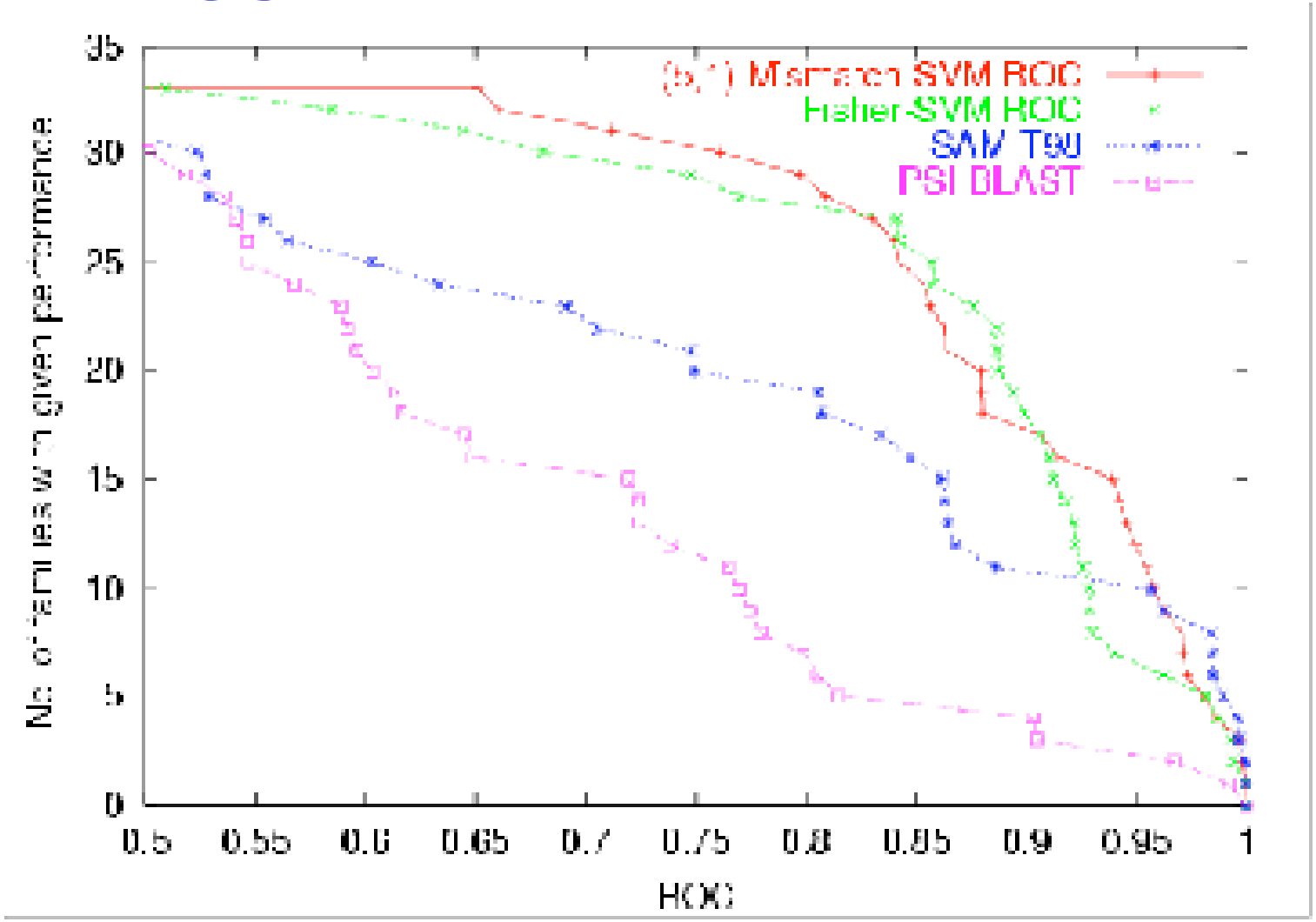


- Tested with experiments on SCOP dataset from Jaakkola *et al.*
- Experiments designed to ask: Could the method discover a new family of a known superfamily?

SCOP Experiments

- 160 experiments for 33 target families from 16 superfamilies
- Compared results against
  - SVM-Fisher (HMM-based kernel)
  - SAM-T98 (profile HMM)
  - PSI-BLAST (heuristic alignment-based method)
- *ROC scores*: area under the graph of true positives as a function of false positives, scaled so that both axes vary between 0 and 1

Results Across All Target Families



Background on Fisher-SVM

- Previous solution [Jaakkola, Diekhans, Haussler]:
  - Use positive examples to train profile HMM, (M_+ ,θ_0)
  - For each training example *x*, *Fisher score* is gradient of log-likelihood score for *x* given M_+ (evaluated at θ_0)

$$x \longrightarrow \nabla_{\theta} \log P(x \mid M_+, \theta)$$

- Method relies on generative model
  - Requires large amount of data or sophisticated priors to train M_+
  - Expensive: dynamic programming (quadratic in sequence length) – for each sequence *x*, forward-backward algorithm to compute features

Aside: Connection with Fisher Kernel

- Consider order k-1 Markov chain model for positive sequences, with parameters

  $\theta_{t|s_1..s_{k-1}} = P(x_j = t \mid x_{j-k+1}..x_{j-1} = s_1..s_{k-1})$

- Corresponding Fisher coordinate for *x* is

  (#occurrences of $s_1..\ s_{k-1}t$ in *x*)/$\theta_{t|s_1..s_{k-1}}$

  $\times$ (#occurrences of $s_1..s_{k-1}$ in *x*)

- Fisher kernel for Markov chain model similar to k-spectrum kernel

Interpretation of Mismatch-SVM Classifier

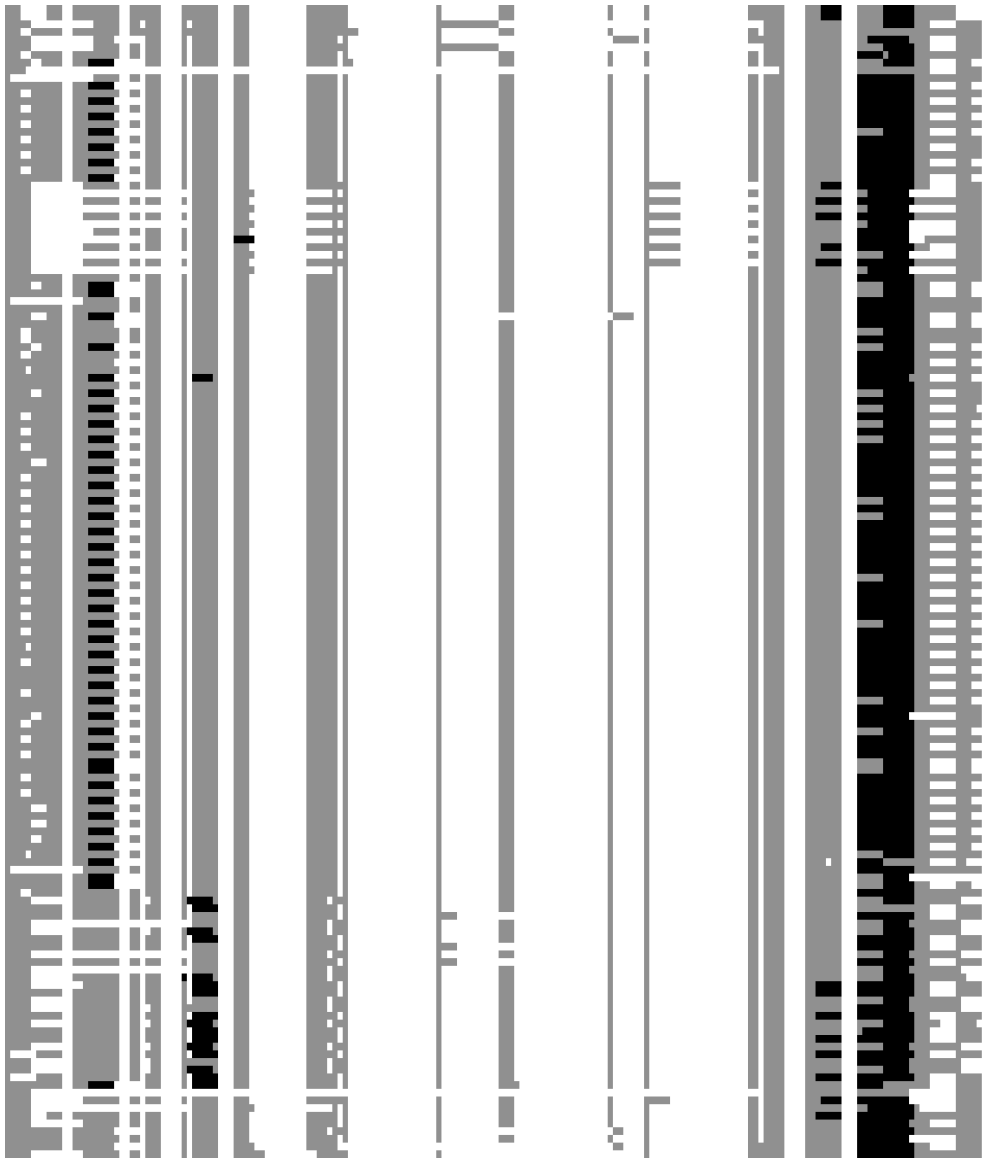
- Rank features by $|w_i|$, associate to +/- class by sign(w_i)
- Top positively-weighted k-mer features learned by SVM map to *conserved regions* in the *multiple alignment* of positive training sequences

```
>saq..---ksp..aelksifek..yaakeg....dpnqlsk...eel.....
..kqliqaef.................p..........
>snklh------..----fafrl..yd-ldk....d-ekisr...del.....
..lqvlrmmv.................gvnisdeqlgsi
>maa..pldqai..gllvatfhk..ysgkeg....dknslsk...gel.....
..keliqkelti...............g..........
>sfggf------..----kvfd-..---edg....d-gyisa...rel.....
..qmvlgk-l.................g..........
>snklh------..----fafrl..yd-ldk....d-dkisr...del.....
..lqvlrmmv.................gvnisdeqlgsi


                  *********
>sll.................k......g..................prtlddl
...fqeldkn...gdgevSFEEFQvlvkkisq......
>---.................-......-...................---adrt
...iqeadqd...gdsaiSFTEFVkvl-----ekvdv.
>pkl.................k......d..................ae-iagl
...medldrn...kdqeVNFQEyvtflgalamiynea.
>---.................-fsegs.e..................idrvekm
...ivsvdsn...rdGRVDFFEFkdmm-----rsvlv.
>---.................-......-...................---adrt
...iqeadqd...gdsaiSFTEFVkvl-----ekvdv.
```

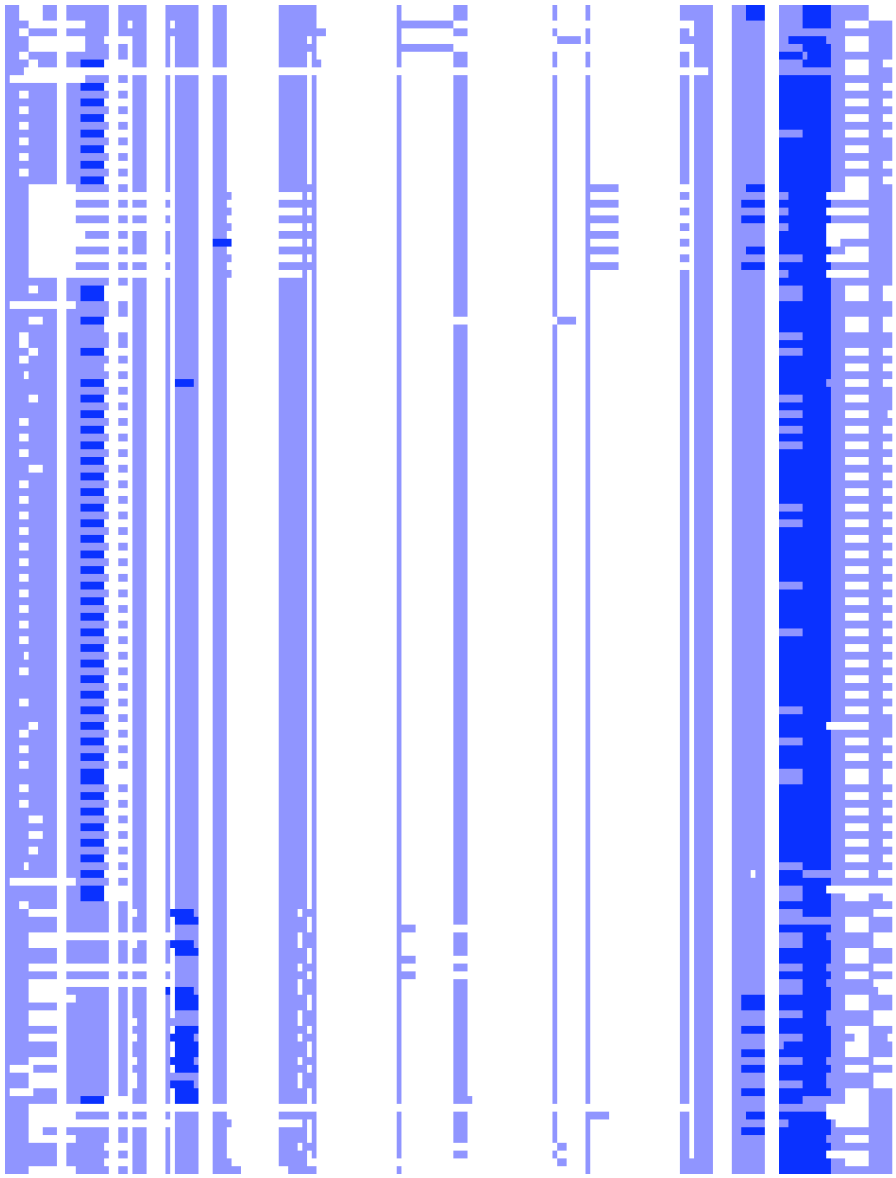
Interpretation of Mismatch-SVM Classifier

- Rank features by $|w_i|$, associate to +/- class by sign(w_i)
- Top positively-weighted k-mer features learned by SVM map to *conserved regions* in the *multiple alignment* of positive training sequences



Interpretation of Mismatch-SVM Classifier

- Rank features by $|w_i|$, associate to +/- class by sign
- Top positively-weighted k-mer features learned by SVM map to *conserved regions* in the *multiple alignment* of positive training sequences



Advantages of Mismatch-SVM

- Mismatch-SVM performs as well as SVM-Fisher but avoids computational expense, training difficulties of profile HMM
- Advantages of string kernel:
  - *Efficient computation*: scales linearly with sequence length
  - *Fast prediction*: classify test sequences in linear time
  - *Interpretation* of learned classifier
  - *General approach* for biosequence data, does not rely on alignment or generative model

Outline

1. Protein classification: Mismatch kernel
   - SVMs and kernel methods
   - Inexact matching through mismatches
   - Efficient kernel computation, fast prediction
2. Experimental results on SCOP dataset
3. Other models for inexact matching
   - Kernels from gaps, substitutions, wildcards
4. Cluster kernels: Semi-supervised methods
   - Using unlabeled data to change the kernel

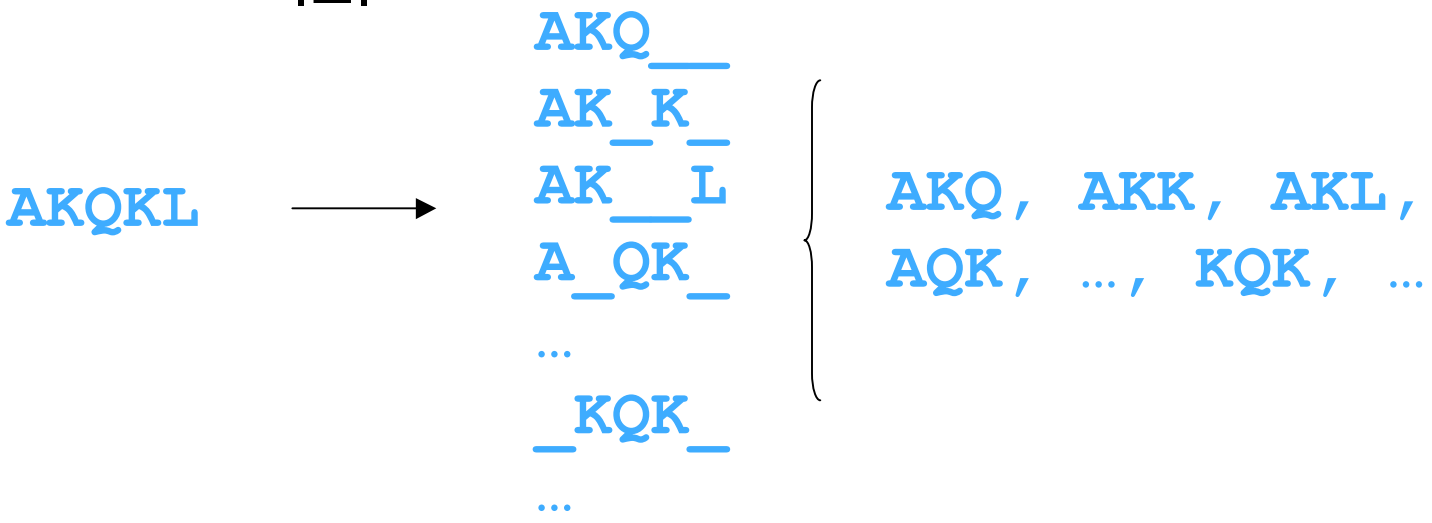
Other Fast(er) Kernels for Inexact Matching

- Mismatch kernel is linear in sequence length, but constant $c_K = k^{m+1}|\Sigma|^m$ depends on alphabet size
- Other models for inexact matching can achieve $O(c_K(|x| + |y|))$ with c_K independent of $|\Sigma|$
  - Restricted gaps
  - Probabilistic substitutions
  - Wildcards

C. Leslie and R. Kuang, *Fast String Kernels for Inexact String Matching.* To appear, COLT/KW 2003.

Inexact Matching through Gaps

- For g-mer *s*, the *gapped match set* $G_{(g,k)}(s)$ consists of all k-mers *t* that occur in *s* with (g - k) gaps
- Size of gapped match set is $O(g^{g-k})$, independent of $|\Sigma|$



(g,k)-Gappy Kernel

- Feature map:

  For a g-mer s, $F_{(g,k)}(s) = (F_t(s))_{\{\text{k-mers } t\}}$,
  where $F_t(s) = 1$ if t is in set $G_{(g,k)}(s)$,
  $F_t(s) = 0$ otherwise;
  extend additively by summing over g-mers s in x

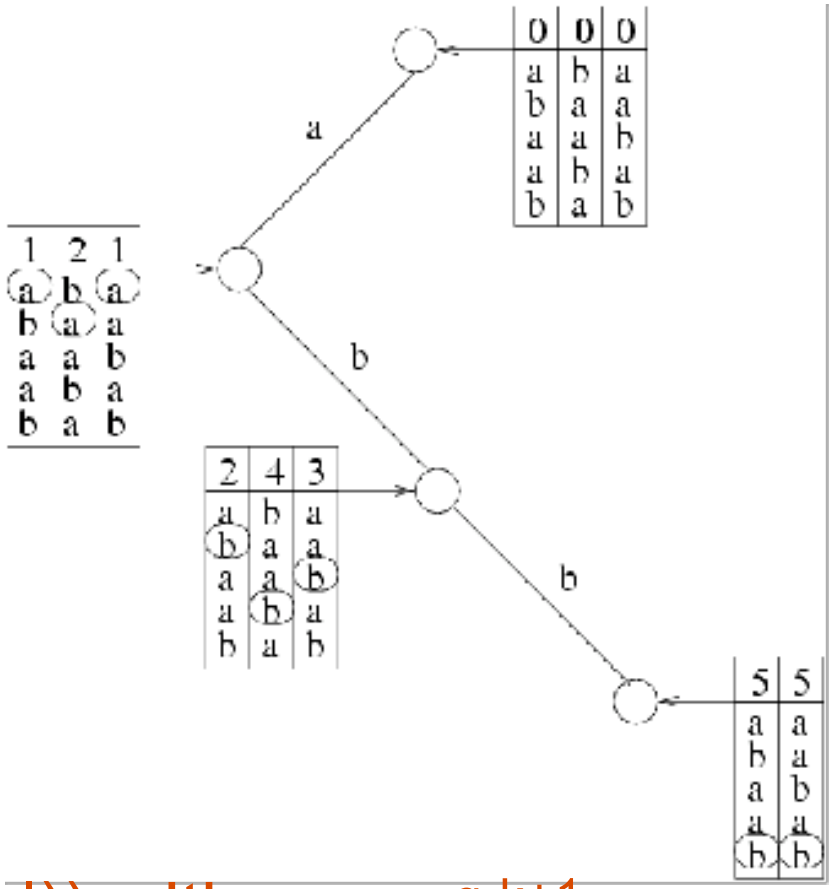
- Weighted version with gap penalty, $0 < \lambda \leq 1$:

  $F_t(s) = (1/\lambda^k) \sum_{\{\text{subseq}(s) = t\}} \lambda^{\text{length(subseq(s))}}$,
  can be computed by dynamic programming

Gives truncated version of Lodhi *et al.* string kernel

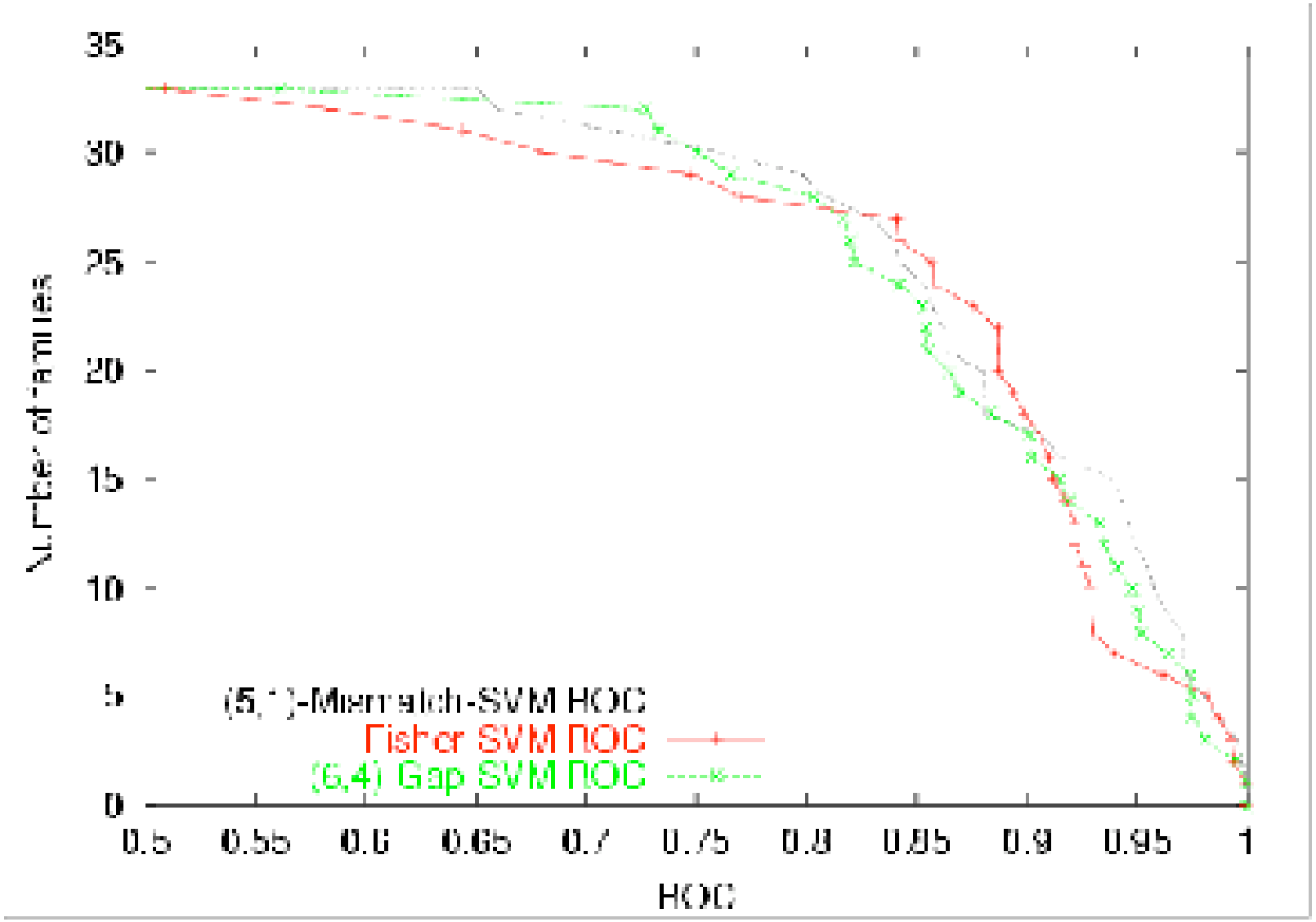
Gappy Kernel Computation

- Traverse instance g-mers in the data, greedily align to k-length paths (k-mer features)
- At leaf node, count instances for each input sequence (or perform restricted dynamic programming for weighted version)



$\Rightarrow$ $O(c_K(|x| + |y|))$ with $c_K = g^{g-k+1}$

Gappy Kernel SCOP Results



Inexact Matching through Probabilistic Substitutions

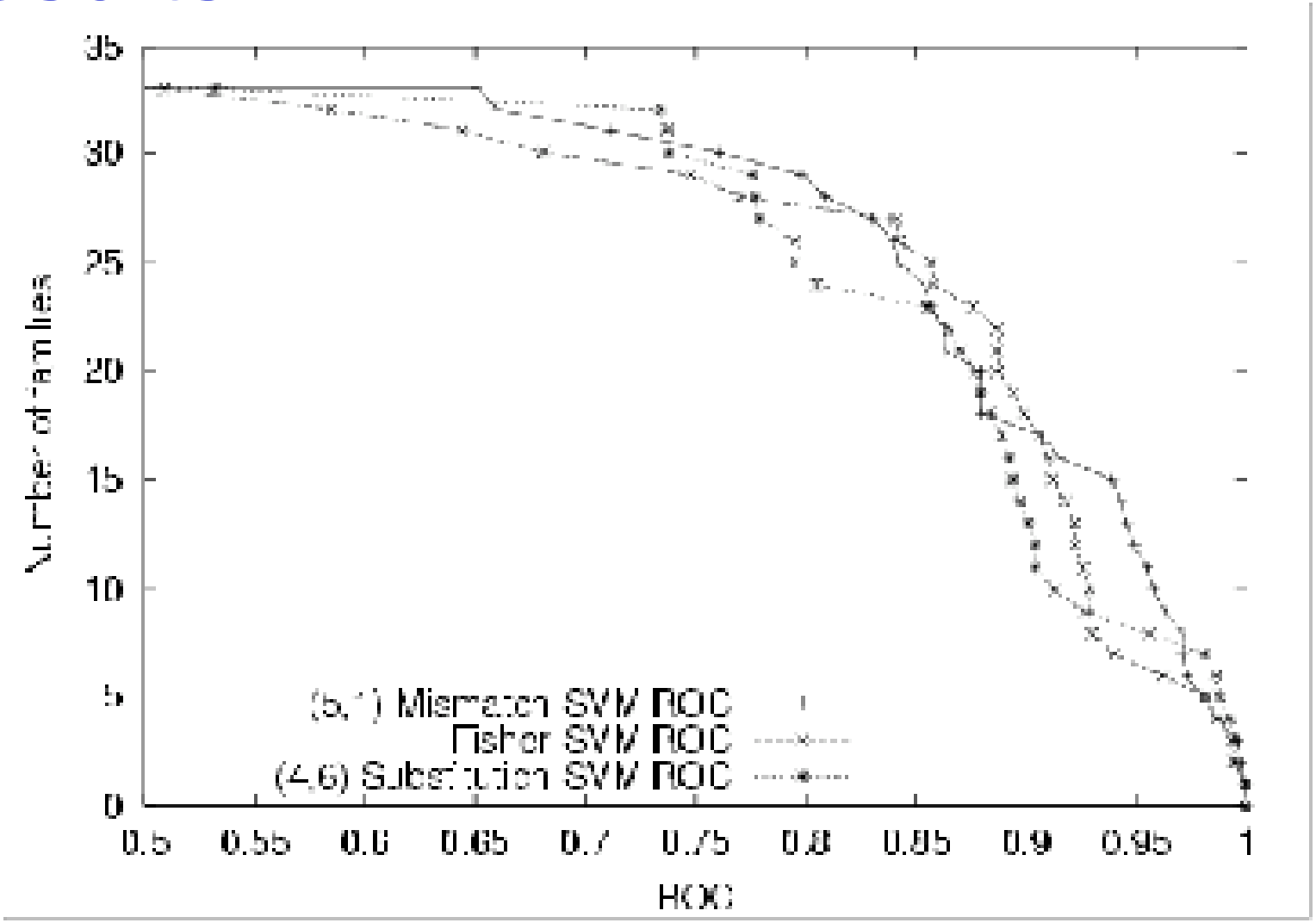
- Use *substitution matrices* to obtain P(a|b), substitution probabilities for residues a, b
- The *mutation neighborhood* $M_{(k,\square)}(s)$ is the set of all k-mers *t* such that

$$- \square_{i=1\ldots k} \log P(s_i|t_i) < \square$$

For a k-mer *s*, map $F_{(k,\square)}(s) = (F_t(s))_{\{\text{k-mers } t\}}$,
where $F_t(s) = 1$ if *t* is in neighborhood $M_{(k,\square)}(s)$,
$F_t(s) = 0$ otherwise;
extend additively

$\square$ $c_K = k\, N_\square$, where $N_\square$ is max size of mutation neighborhood

Substitution Kernel SCOP Results



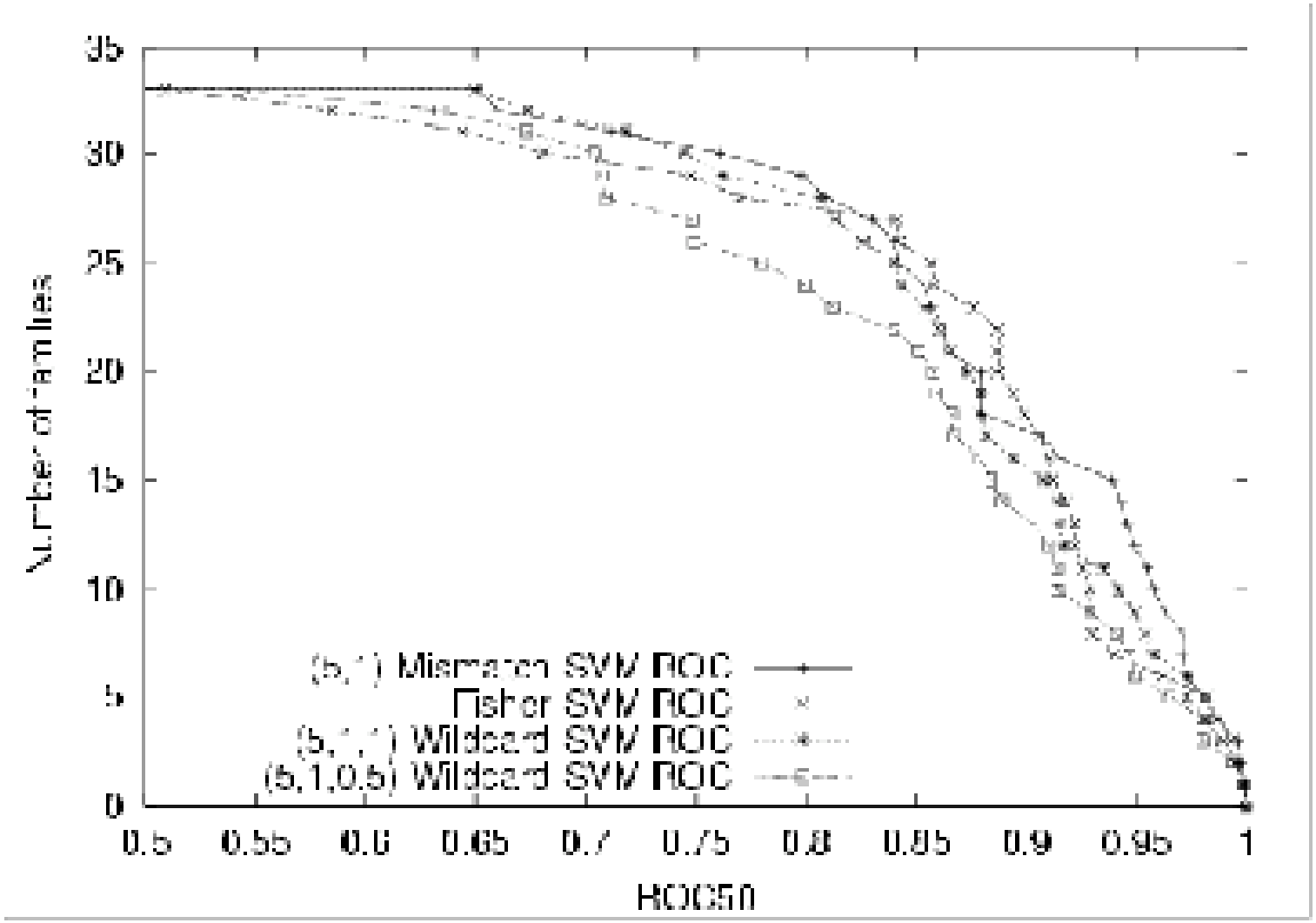
Inexact Matching through Wildcards

- Introduce wildcard character "□", define feature space indexed by k-mers from □□{□}, allowing up to m wildcards

  For a k-mer *s*, $F_{(k,m)}(s) = (F_t(s))_{\{\text{k-mers } t\}}$,
  where $F_t(s) = □^{\#\text{wildcards in } t}$, if *t* matches *s*,
  $F_t(s) = 0$ otherwise;
  extend additively

□ Using pruned depth k trie over □□{□},
$c_K = k^{m+1}$

Wildcard Kernel SCOP Results



Related Recent String Kernel Work

- For exact matching case, Vishwanathan and Smola compute *convex combinations* of kernels using suffix trees
- Ben-Hur *et al.* define a motif kernel: features are known motifs, stored using trie
- Li and Noble use feature vectors of pairwise alignment scores (Smith-Waterman, BLAST)
- Can describe all the kernels here using *transducer formalism* (finite state automata) of Cortes *et al.*

Outline

1. Protein classification: Mismatch kernel
   - SVMs and kernel methods
   - Inexact matching through mismatches
   - Efficient kernel computation, fast prediction
2. Experimental results on SCOP dataset
3. Other models for inexact matching
   - Kernels from gaps, substitutions, wildcards
4. Cluster Kernels: Semi-supervised methods
   - Using unlabeled data to change the kernel

Use of Unlabeled Data

- About 30,000 proteins with known structure (labeled proteins), but about 1 million sequenced proteins
- BLAST, PSI-BLAST: widely-used heuristic alignment-based sequence similarity scores
  - Good *local similarity score*, less useful for more remote homology
  - BLAST/PSI-BLAST E-values give good measure of distance between sequences
- Can we use unlabeled data, combined with good local distance measure, for *semi-supervised* approach to protein classification?

Cluster Kernels

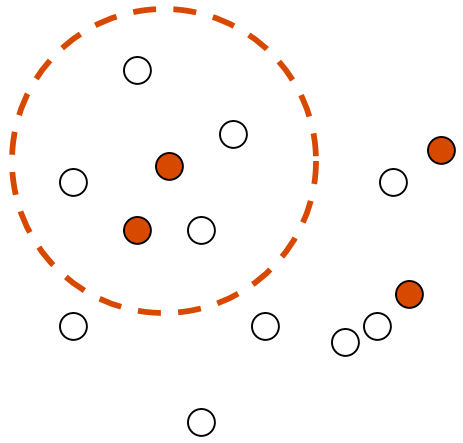
- Use unlabeled data to change the (string) kernel representation
- *Cluster assumption:* decision boundary should pass through low density region of input space; clusters in the data are likely to have consistent labels
  - Profile kernel
  - Bagged kernel

J. Weston, C. Leslie, D. Zhou, A. Elisseeff, and W. S. Noble, *Cluster Kernels for Semi-supervised Protein Classifiction.* Submitted.

Profile Kernel

- Represent sequence x by the average sequences in its neighborhood $N(x)$:

$$F^{Profile}(x) = (1/|N(x)|) \sum_{x' \text{ in } N(x)} F(x')$$

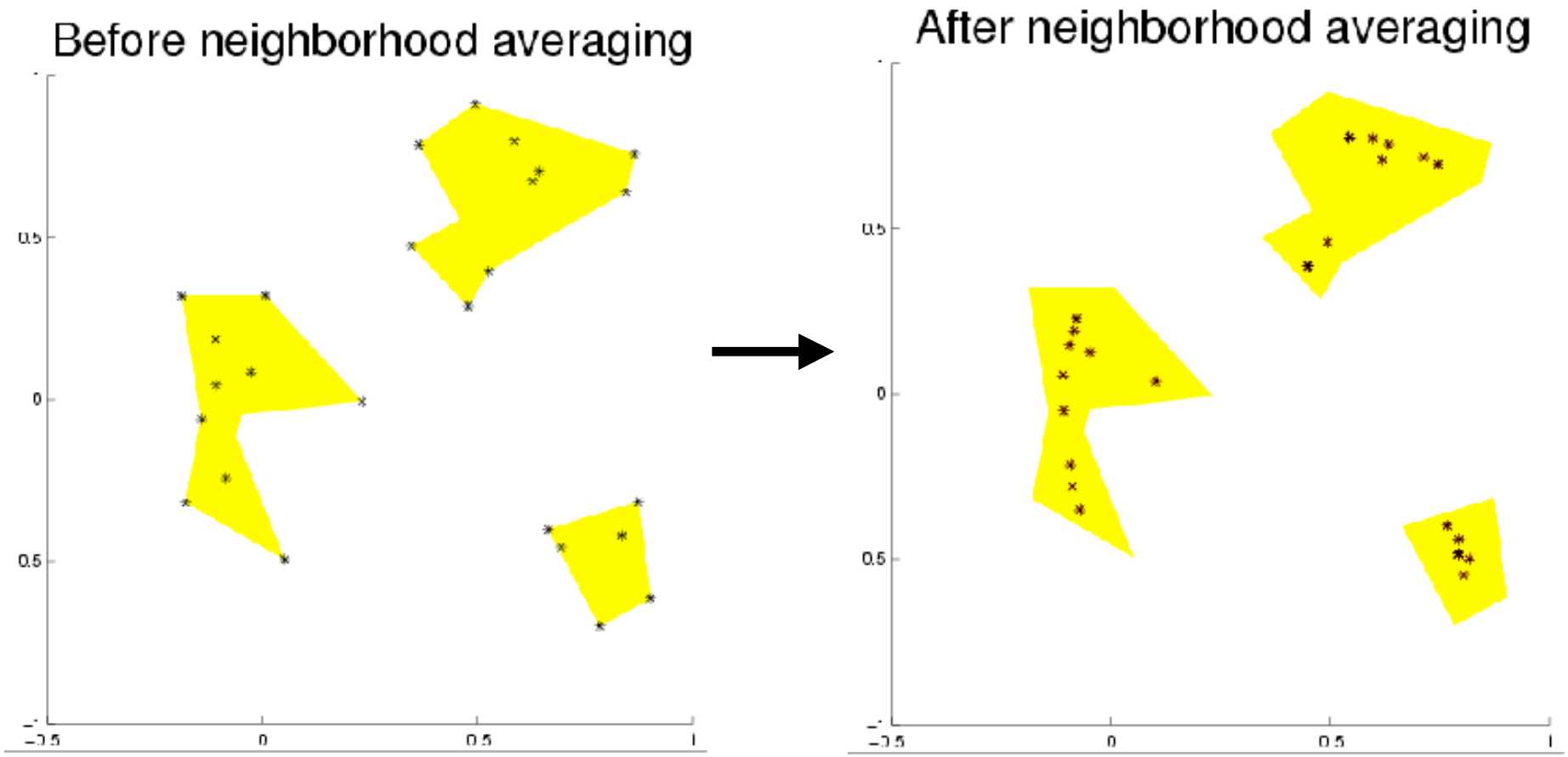


- Profile kernel:

$$K^{Profile}(x,y) = (1/|N(x)||N(y)|) \sum_{N(x)xN(y)} K(x',y')$$

- Use PSI-BLAST distance and mismatch kernel as base kernel

Profile kernel addresses cluster assumption



Bagged Kernel

- Use k-means clustering to cluster data (labeled + unlabeled), N bagged runs
- Using N runs, define

$$p(x,y) = (\#\text{ times } x,\ y \text{ are in same cluster})/N$$

- Bagged kernel:

$$K^{\text{Bagged}}(x,y) = p(x,y)\ K(x,y)$$

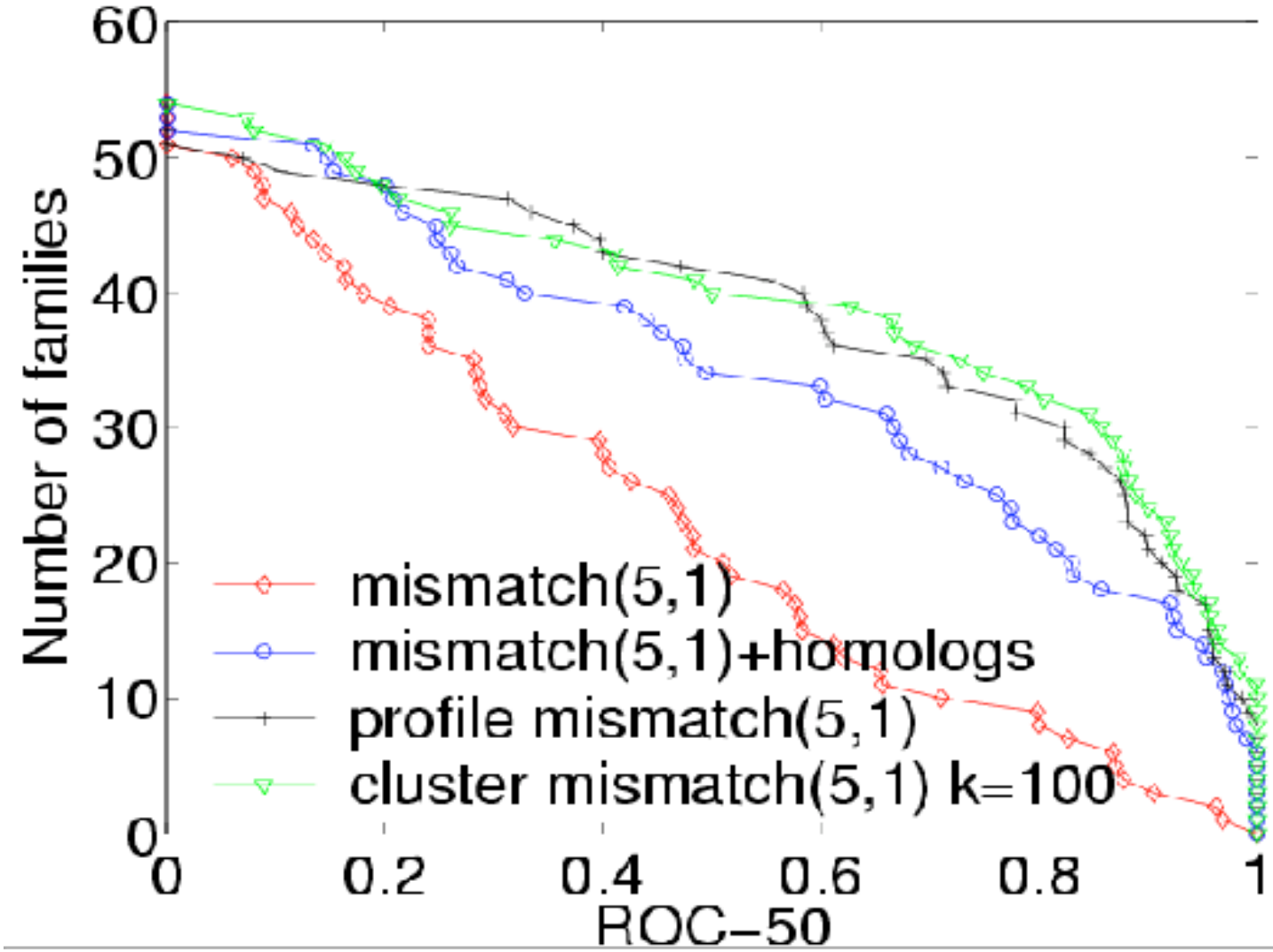
- Use PSI-BLAST for clustering, mismatch kernel for underlying kernel

,

Experimental Set-up

- *Full dataset:* 7329 SCOP protein sequences
- *Experiments:*
  - 54 target families (remote homology detection)
  - Test + training approximately for each experiment <4000 sequences, other data treated as unlabeled
- *Evaluation:* How well do cluster kernel approaches compare to the standard approach, adding positive homologs to dataset?

Results for Cluster Kernels



Conclusions

- SVMs with *string kernels* – like mismatch kernels – that incorporate *inexact matching* are competitive with best-known methods for protein classification
- Efficient kernel computation: $O(c_K(|x| + |y|))$, linear-time prediction, feature extraction
- Gaps, substitutions, and wildcards give kernel constant c_K that is independent of alphabet size
- Semi-supervised *cluster kernels* – using unlabeled data to modify kernel representation – improve on original string kernel

Future Work

- Full *multiclass* protein classification problem
  - 1000s of classes of different sizes, hierarchical labels
  - Use of unlabeled data for improving kernel representation
- *Domain segmentation* problem: predict and classify domains of multidomain proteins
- Develop and implement *semi-supervised ranking* approaches, make available on web server
- *Local structure* prediction: predict local conformation state (backbone angles) for short peptide segments, step towards structure prediction

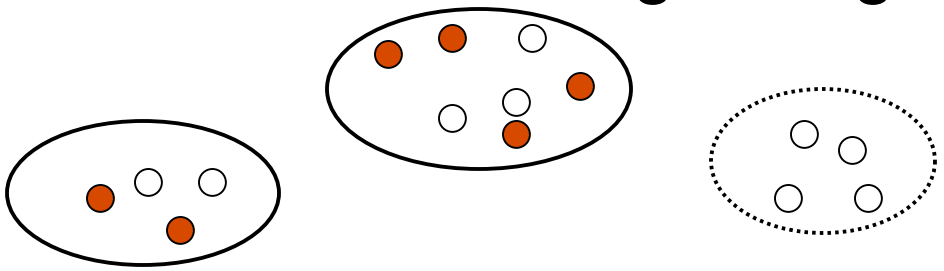
Protein Ranking

- *Pairwise sequence comparison:* most fundamental bioinformatics application
- BLAST, PSI-BLAST: widely-used heuristic alignment-based sequence similarity scores
  - Given query sequence, search unlabeled database and return ranked list of similar sequences
  - Query sequence does not have to belong to known family or superfamily
  - Good local similarity score, less useful for more remote homology
- Can we use *semi-supervised machine learning* to improve on PSI-BLAST?

Joint work with J. Weston, A. Elisseeff, and W. S. Noble

Ranking Induced by Clustering

- *Idea:* Map out protein sequence space by performing constrained clustering, using label constraints

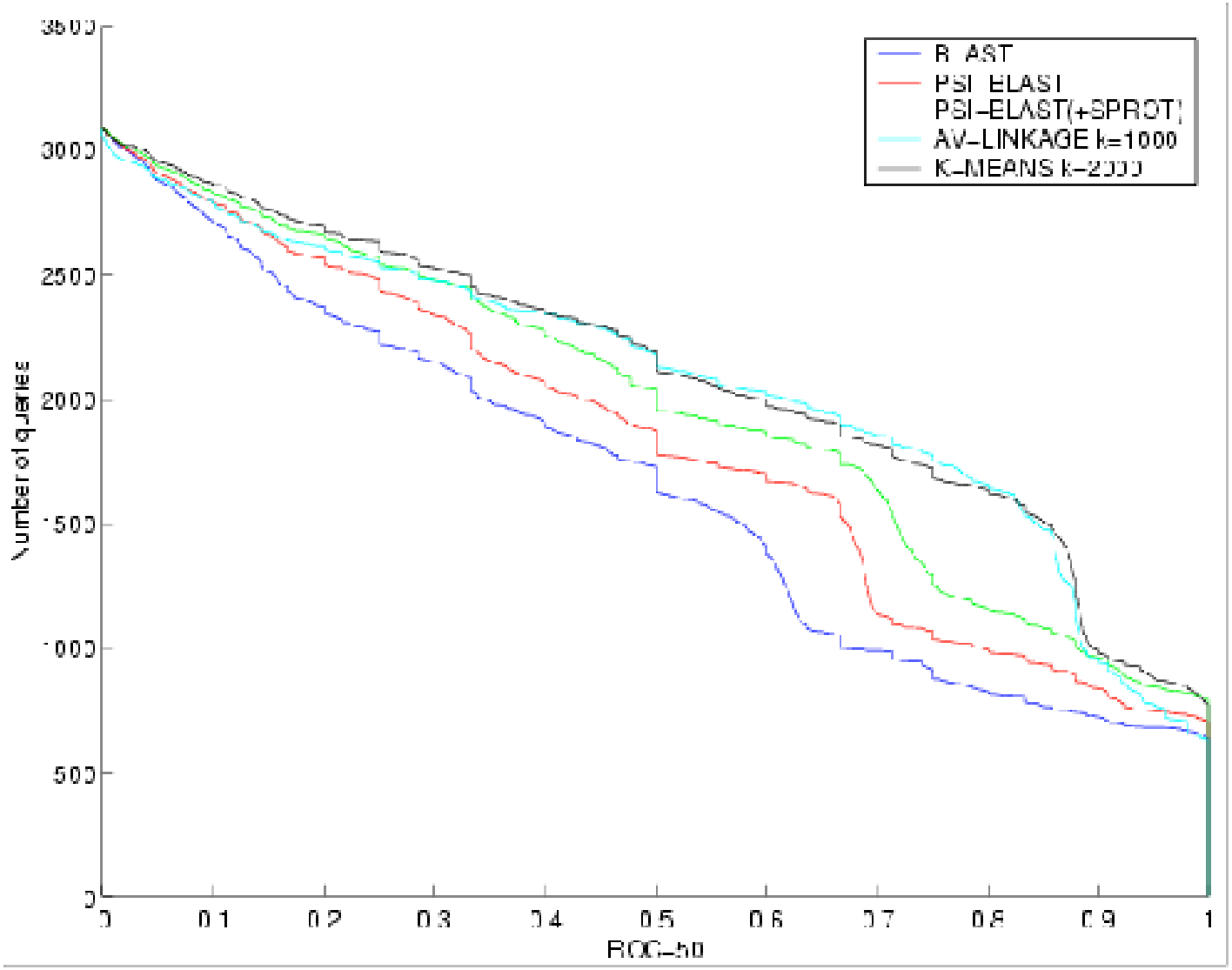


- Output ranking: rank by cluster, nearest cluster first
- Use dissimilarity measure derived from PSI-BLAST
- Best to use constrained clustering for model selection (number of clusters) based on labeled data, then use regular efficient clustering algorithms
  - Generalized (“kernel”) k-means
  - Hierarchical clustering (average linkage)

Experimental Set-up for Ranking

- *Training set:* 4246 SCOP protein sequences (from 554 superfamilies) – known classes
- *Test set:*
  - 3083 SCOP protein sequences (from 516 different superfamilies) – hidden classes
  - 101,403 unlabeled sequences from SWISSPROT
- *Task:* How well can we retrieve database sequences (from train + test sets) in same superfamily as query? Evaluate with ROC-50 scores
- For initial experiments, SWISSPROT sequences only used for PSI-BLAST scores, not for clustering

Ranking Results for Clustering



Label Propagation

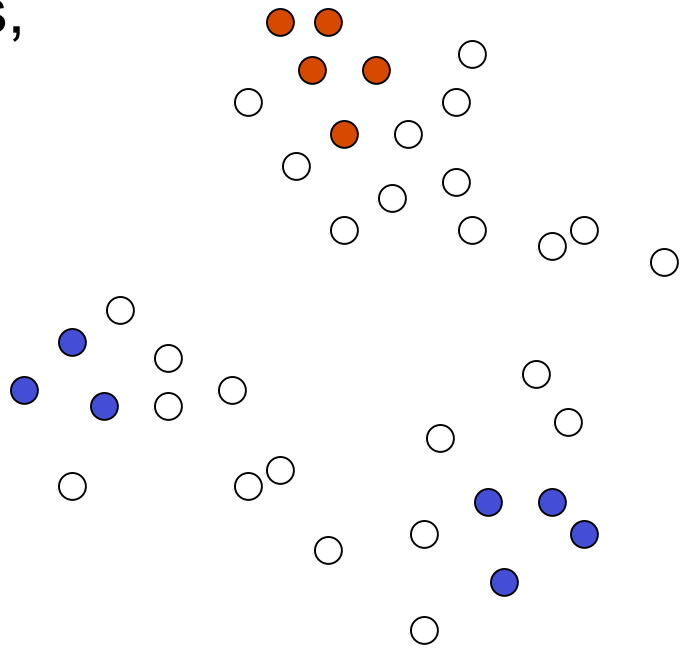
- Zhu and Ghahramani: Propagate labels through dense regions of example space

- Y is m x 2 matrix of label probabilities, where m is number of examples
- Clamp known labels:

  (1,0) for class 1, (0,1) for class 2
- K is matrix of transition probabilities (sparse, derived from PSI-BLAST)

Iterate until convergence to fixed point:

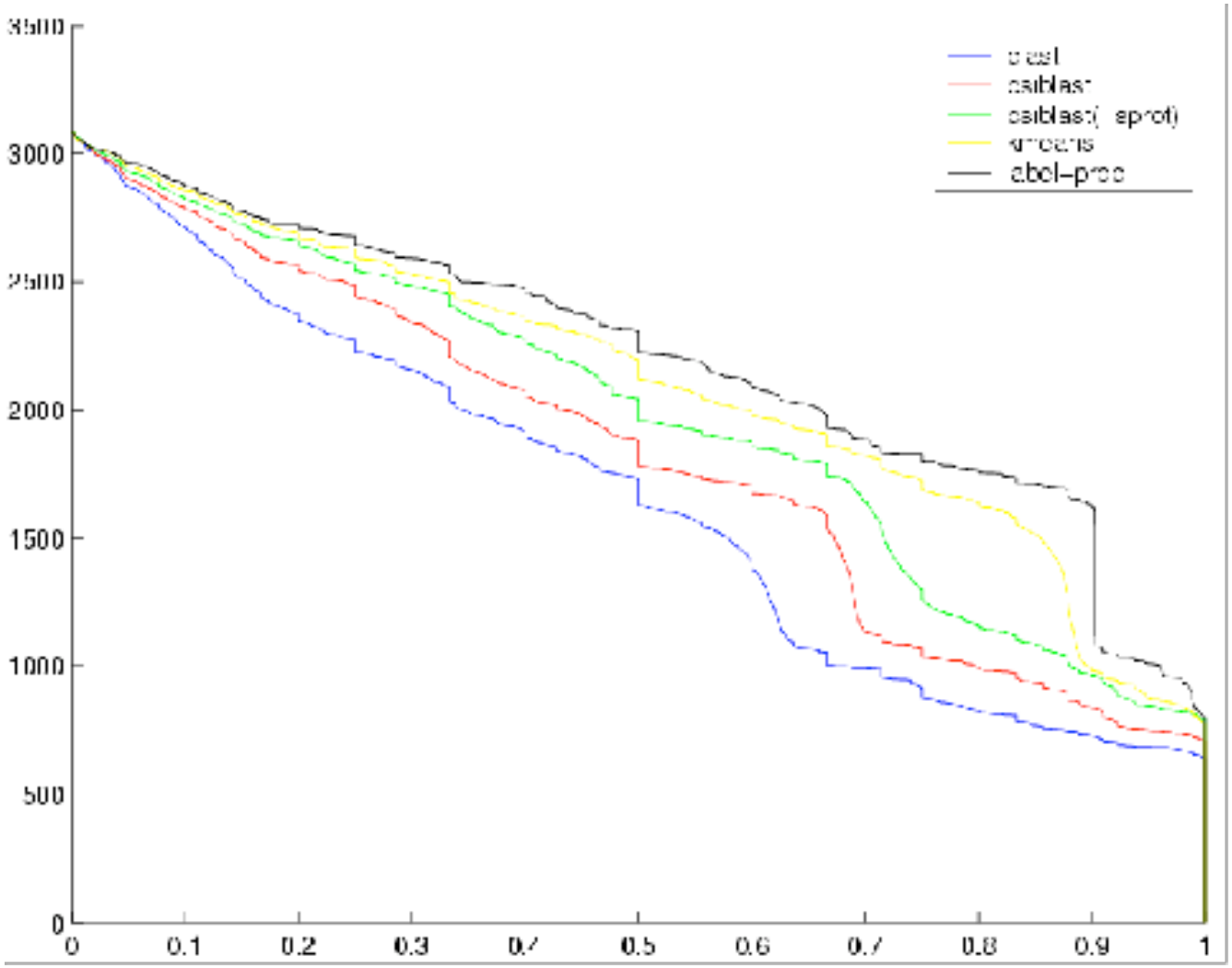
- Propagate: Y ← K Y
- Row normalize Y
- Clamp known labels



Online Semi-Supervised Approach

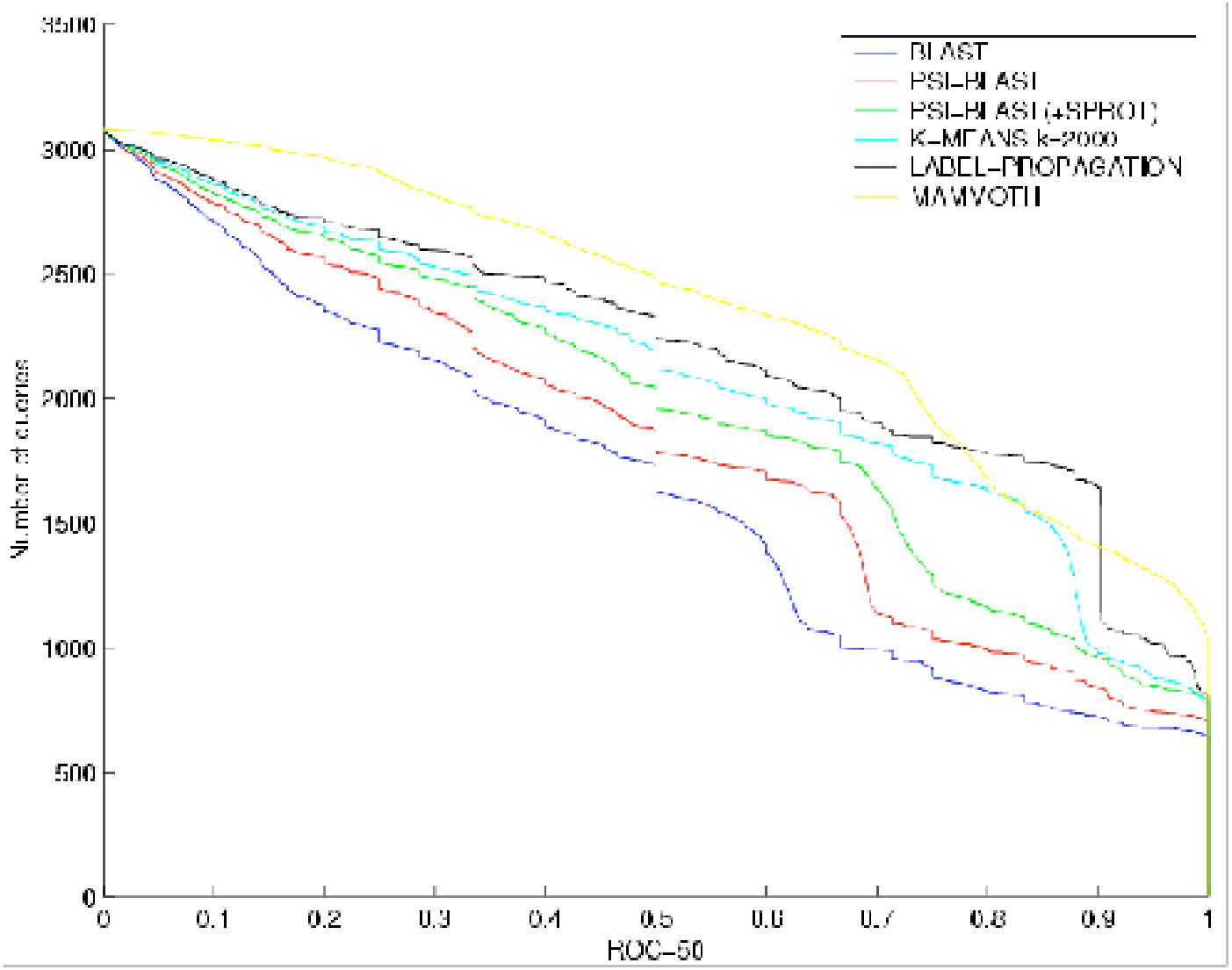
- *Idea:* PSI-BLAST returns good high-confidence prediction scores, can rule out extremely low-confidence scores
  - Given query, assign *positive pseudo-label* to sequences with good (small) E-values
  - Assign *negative pseudo-label* to sequences with poor (large) E-values
- Small number of (pseudo-)labeled examples, rest of database considered unlabeled: apply semi-supervised technique
- Known class information not used, but can use for model selection

Ranking Results for Label-Prop

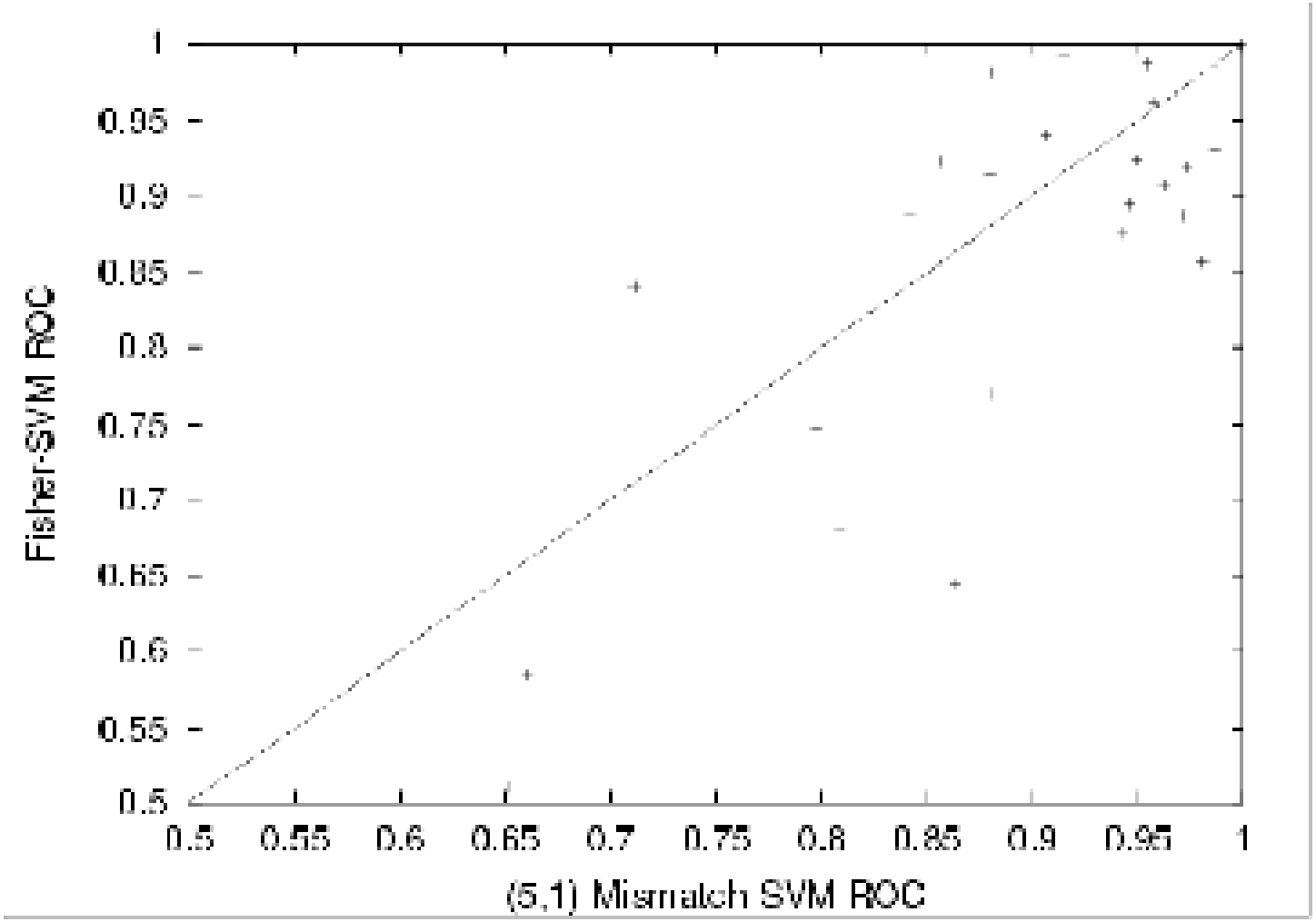


ROC-50

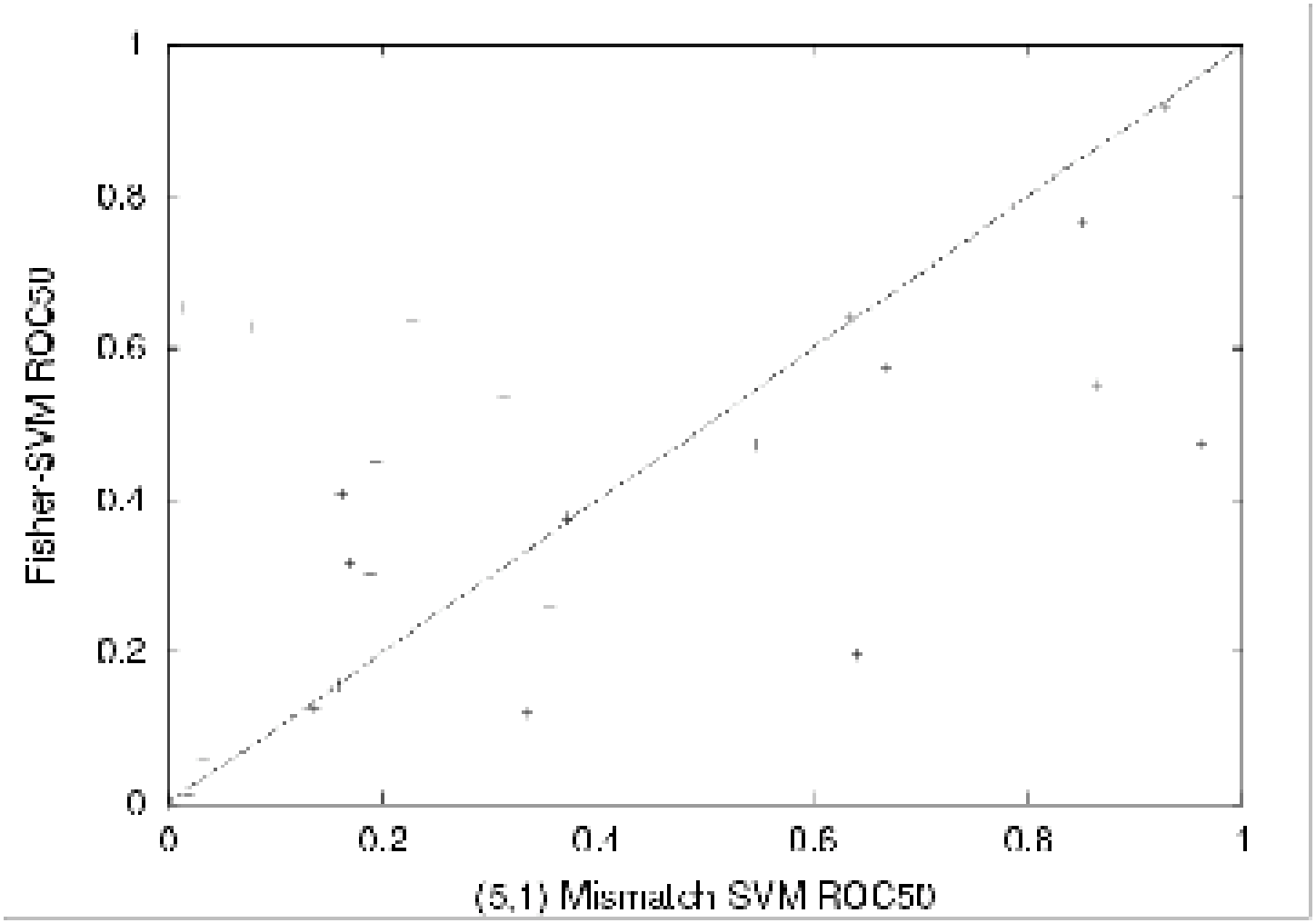
Ranking Results: Comparison with Structure-Structure Scores



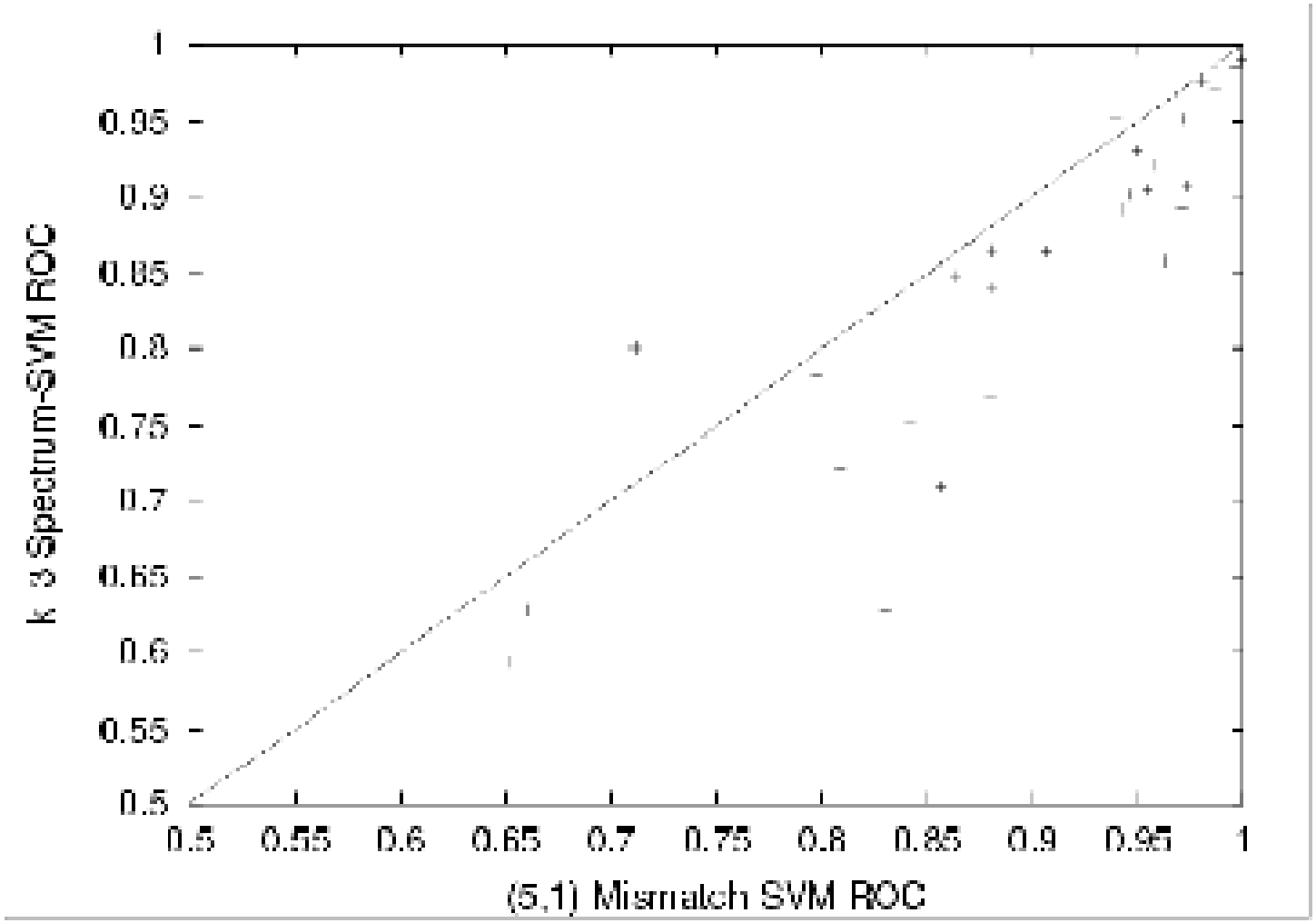
(5,1)-Mismatch vs Fisher Using ROC Scores



(5,1)-Mismatch vs Fisher Using ROC-50 Scores



(5,1)-Mismatch vs. 3-Spectrum Using ROC Scores



(5,1)-Mismatch vs. 3-Spectrum Using ROC-50 Scores