Sequence Kernels for Predicting Protein Essentiality

Cyril Allauzen[2] Mehryar Mohri[1,2] Ameet Talwalkar[1]

[1]*Courant Institute (NYU)*
[2]*Google Research*

Essential Proteins

- Minimal set of proteins required to sustain life
  - understand how cells function
  - targets for therapeutic drugs
- Experimental techniques are expensive
  - gene knockout studies, RNA interference
- Protein sequences readily available
- Goal: Predict essentiality from sequence data
  - propose novel domain-based sequence kernels

Outline

- Initial Experiments
  - sequence kernels
- Domain-based Kernels
  - independent and joint
- Performance of Domain-based Kernels

Outline

- Initial Experiments
  - sequence kernels
- Domain-based Kernels
  - independent and joint
- Performance of Domain-based Kernels

Sequence Kernels

- n-gram kernel: *n*-grams
  - length *n* subsequences

- Motif kernel: sequence motifs [Ben-Hur & Brutlag '03]
  - short protein subsequences

- Pfam kernel: protein domains [Ben-Hur & Noble '05]
  - structural/functional protein regions
  - uses Pfam Library [Sonnhammer et al. '97]

Initial Experiments

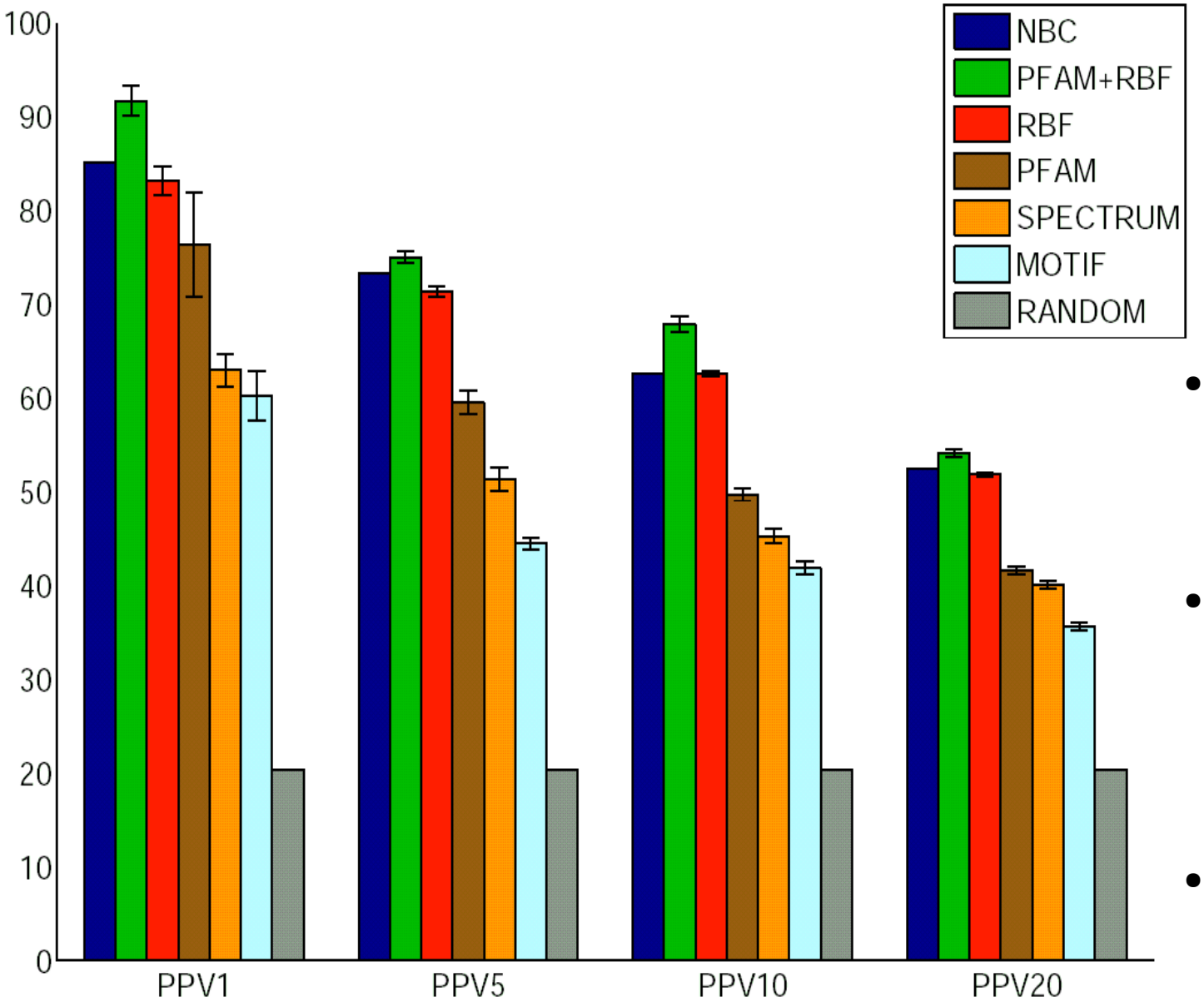
- Gustafson et al. '06
  - 4728 yeast proteins (20% essential)
  - 16 sequence features (size, evolutionary conservation)
  - Naïve Bayes classifier with 50% training
  - positive predictive value (PPV)
    - measured for top 1, 5, 10 and 20% of predictions
    - e.g. PPV5 is % of essential proteins in top 5% of predictions

Initial Experiments

- Gustafson et al. '06
  - 4728 yeast proteins (20% essential)
  - 16 sequence features (size, evolutionary conservation)
  - Naïve Bayes classifier with 50% training
  - positive predictive value (PPV)

- Our experiments
  - SVM [Cortes & Vapnik '95] with sequence kernels (linear), sequence features (RBF) and combinations
  - 8.3% training (otherwise identical to Gustafson)

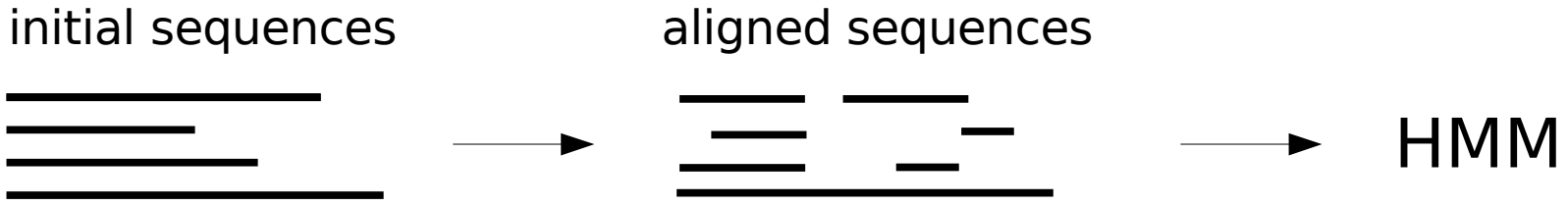
Initial Experiments



- Pfam is best sequence kernel
- Pfam/RBF combination gives best results
- state-of-the-art results with less training data

Pfam Library [Sonnhammer et al. '97]

- Represents common protein domains
- For each domain:
  - literature curated data (10 to 1000+ sequences)
  - multiple sequence alignment ⇨ HMM

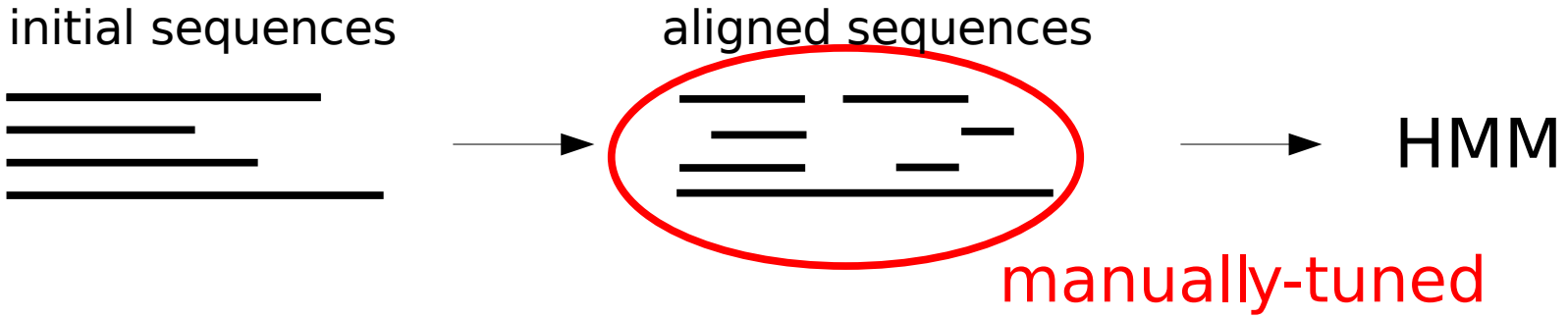


  - HMM generates domain prediction score (feature in Pfam Kernel)

Pfam Library [Sonnhammer et al. '97]

- Represents common protein domains

- For each domain:
  - literature curated data (10 to 1000+ sequences)
  - multiple sequence alignment ⇨ HMM



  - HMM generates domain prediction score (feature in Pfam Kernel)

Domain-Based Kernels

- Pfam kernel performs well
  - manually tuned sequence alignments

- Can we find a more general solution to extract similarity from domains?

Domain-Based Kernels

- Pfam kernel performs well
  - manually tuned sequence alignments
- Can we find a more general solution to extract similarity from domains?

- Use automata-based n-gram kernels
  - intuitive
  - efficiently computable

Outline

- Initial Experiments
  - sequence kernels
- Domain-based Kernels
  - independent and joint
- Performance of Domain-based Kernels

n-gram Kernel

- Measure similarity by all *n*-gram counts

$$K_n(x, y) = \sum_{|z|=n} c_x(z)\, c_y(z)$$

where $c_x(z)$ is number of occurrences of z in x

n-gram Kernel

- Measure similarity by all *n*-gram counts

$$K_n(x, y) = \sum_{|z|=n} c_x(z)\, c_y(z)$$

where $c_x(z)$ is number of occurrences of z in x

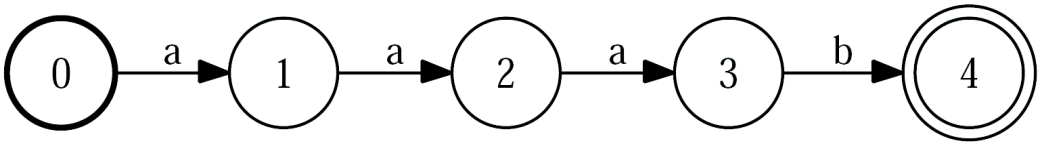
- Example: $x = aaab$ and $y = bbbbab$
  - bigrams:

$$\begin{bmatrix} aa \\ ab \\ ba \\ bb \end{bmatrix} \Longrightarrow \vec{x} = \begin{bmatrix} 2 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \vec{y} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 3 \end{bmatrix} \Longrightarrow K_2(x, y) = \vec{x} \cdot \vec{y} = 1$$

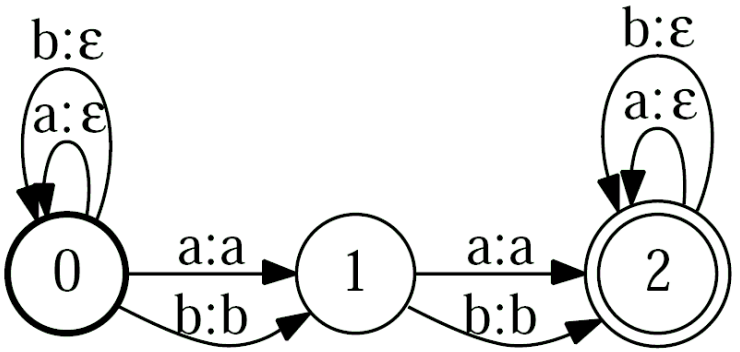
n-gram Kernel with Transducers

- Input Automata:

$x = aaab$



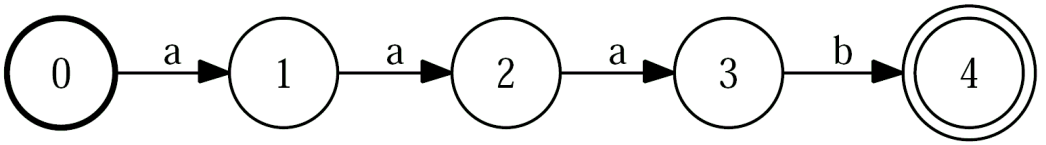
- Counting Transducer (T_2)



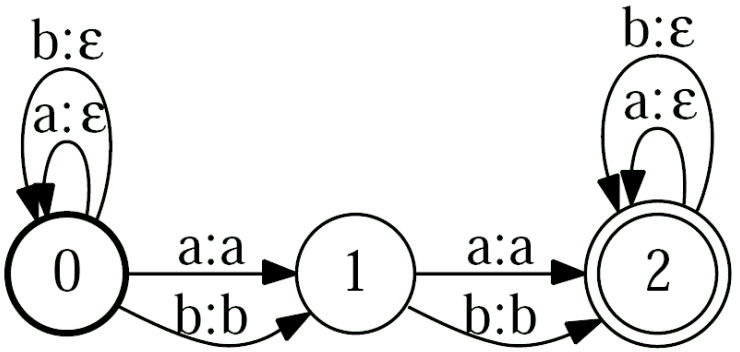
n-gram Kernel with Transducers

- Input Automata:

$x = aaab$



- Counting Transducer (T_2)



- Transducer representation
  - $K_n(x, y) = (T_n \circ T_n^{-1})(x, y)$
  - guaranteed to be PDS [Cortes et al. '04]

Independent Domain Kernel

- Represent sequences in feature space of P domains
  - each feature is n-gram similarity with domain

$$K_I(x, y) = \begin{bmatrix} K_n(x, D_1) & \dots & K_n(x, D_P) \end{bmatrix} \begin{bmatrix} K_n(y, D_1) \\ \vdots \\ K_n(y, D_P) \end{bmatrix}$$

Independent Domain Kernel

- Represent sequences in feature space of P domains
  - each feature is n-gram similarity with domain

$$K_I(x,y)=\begin{bmatrix}K_n(x,D_1) & \dots & K_n(x,D_P)\end{bmatrix}\begin{bmatrix}K_n(y,D_1)\\ \vdots \\ K_n(y,D_P)\end{bmatrix}$$

- Example: $x=aaab$ and $y=bbbbab$
  - $D_i=\{aaa,bbb,aabbb\}$
    $D_j=\{abab,babab\}$

bigrams

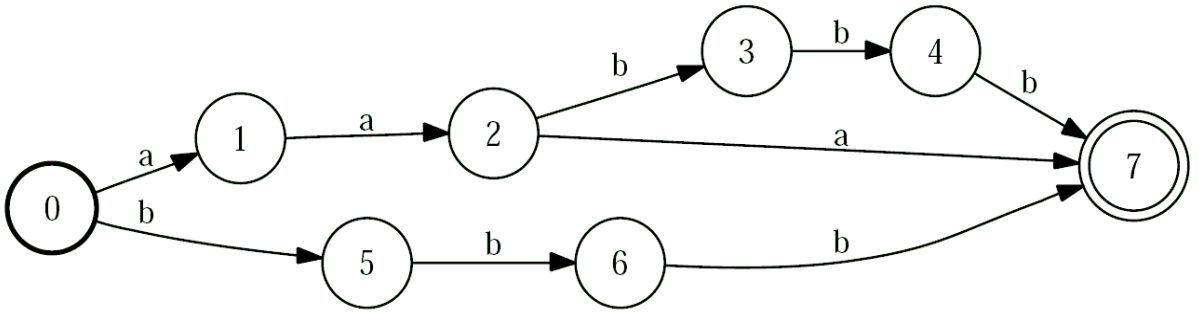
$$\vec{D}_i=\begin{bmatrix}3\\1\\0\\4\end{bmatrix},\quad \vec{D}_j=\begin{bmatrix}0\\4\\3\\0\end{bmatrix}$$

  - IDK: $K_I(x,y)=\begin{bmatrix}\vec{x}\cdot\vec{D}_i & \vec{x}\cdot\vec{D}_j\end{bmatrix}\begin{bmatrix}\vec{y}\cdot\vec{D}_i\\ \vec{y}\cdot\vec{D}_j\end{bmatrix}=119$

IDK with Transducers

- Inputs and counting transducer as in *n*-gram kernel

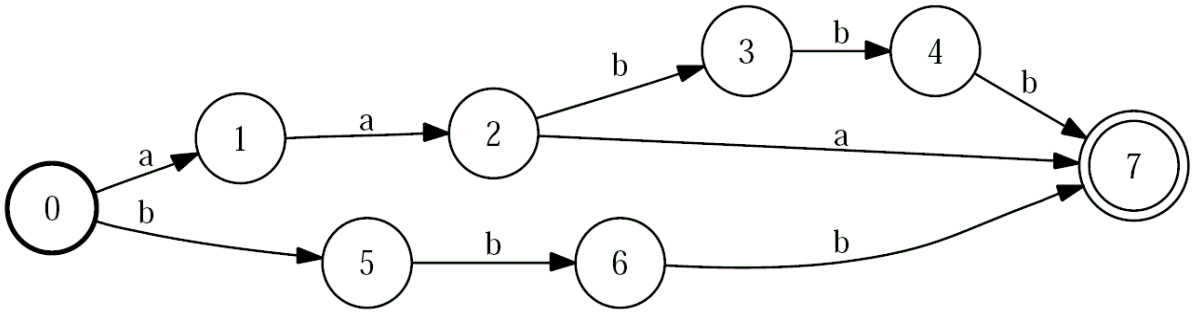
- Domain Automata: $D_i = \{aaa, bbb, aabbb\}$



IDK with Transducers

- Inputs and counting transducer as in *n*-gram kernel

- Domain Automata: $D_i = \{aaa, bbb, aabbb\}$



- Calculate $K_n(x, D_i)$
  - compute $T_n \circ x$ and $T_n \circ D_i$, project onto outputs, optimize
  - intersect projections and compute sum of all paths

- Requires $m \times P$ kernel computations (K_n)
  - $K_I(x, y)$ grows in $O(|x|+|y|)$

Does IDK capture domain similarity?

- $D_i = \{ aaa, bbb, aabbb \}$ characterized by “aa” and “bb”
- $x = aaab$ contains “aa” but no “bb”
- $y = bbbbab$ contains “bb” but no “aa”

Does IDK capture domain similarity?

- $D_i = \{ aaa, bbb, aabbb \}$ characterized by “aa” and “bb”
- $x = aaab$ contains “aa” but no “bb”
- $y = bbbbab$ contains “bb” but no “aa”
- x and y are similar to D_i for different reasons
  - but IDK reports large similarity
  - similarity to domains measured independently

Does IDK capture domain similarity?

- $D_i = \{ aaa, bbb, aabbb \}$ characterized by “aa” and “bb”
- $x = aaab$ contains “aa” but no “bb”
- $y = bbbbab$ contains “bb” but no “aa”
- x and y are similar to D_i for different reasons
  - but IDK reports large similarity
  - similarity to domains measured independently
- New idea: calculate similarity based on n-grams common to all three together (x, y, D_i)

Joint Domain Kernel

- Measure 3-way similarity for each domain
  - *n*-gram kernel with weights on each *n*-gram

$$K_{D_i}(x, y) = \sum_{|z|=n} c_x(z)\, c_{D_i}^2(z)\, c_y(z)$$

- JDK: $K_J(x, y) = \sum_{i=1}^{P} K_{D_i}(x, y)$

Joint Domain Kernel

- Measure 3-way similarity for each domain
  - *n*-gram kernel with weights on each *n*-gram

$$K_{D_i}(x, y) = \sum_{|z|=n} c_x(z)\, c_{D_i}^2(z)\, c_y(z)$$

- JDK: $K_J(x, y) = \sum_{i=1}^{P} K_{D_i}(x, y)$

- Example:
  - $K_{D_i}(x, y) = 1$
  - $K_{D_j}(x, y)\, 1 \cdot 4^2 \cdot 1 = 16$

bigrams:

$$\vec{x} = \begin{bmatrix} 2 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \vec{y} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 3 \end{bmatrix} \qquad \vec{D}_i = \begin{bmatrix} 3 \\ 1 \\ 0 \\ 4 \end{bmatrix}, \quad \vec{D}_j = \begin{bmatrix} 0 \\ 4 \\ 3 \\ 0 \end{bmatrix}$$

JDK with Transducers

- All automata and transducers identical to IDK
- Calculate $K_{D_i}(x, y)$
  - compute $T_n \circ D_i$, project onto outputs, optimize (A_i)
  - compute $T_n \circ x$ and $T_n \circ y$, project, optimize (Y_x, Y_y)
  - intersect $A_i \circ Y_x$ and $A_i \circ Y_y$ and find sum of all paths
- $K_{D_i}(x, y) = (T_n \circ A_i \circ A_i \circ (T_n)^{-1})(x, y)$ (PDS)
- Requires $m^2 \times P$ kernel computations (K_{D_i})
  - $K_J(x, y)$ grows in $O(|x||y|)$

Treatment of Gaps

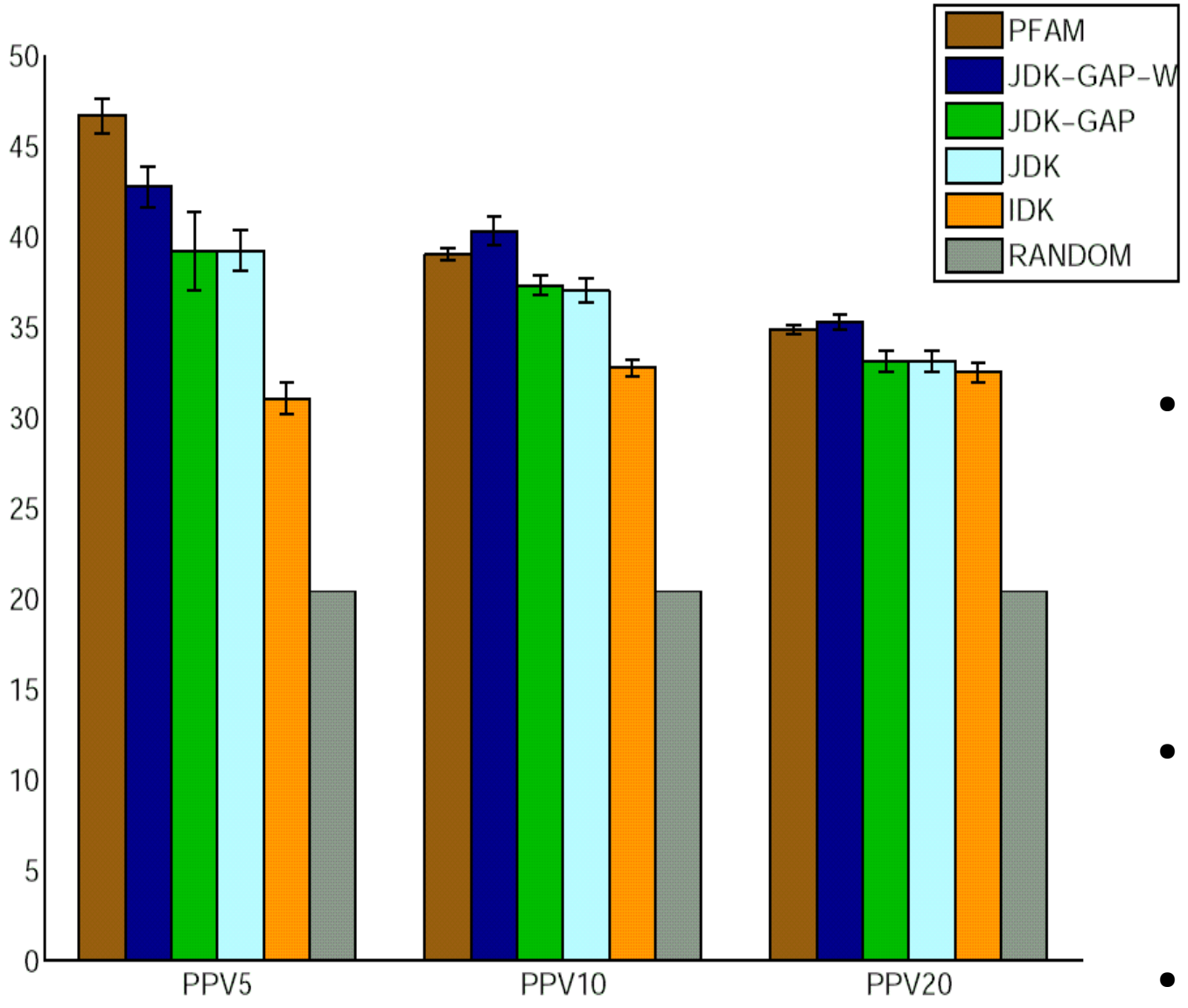
- Can we do better if we use Pfam alignment data?
  - gap symbol used to pad alignments

- Two approaches:
  - ignore them
  - treat as wildcards that match all other symbols

Outline

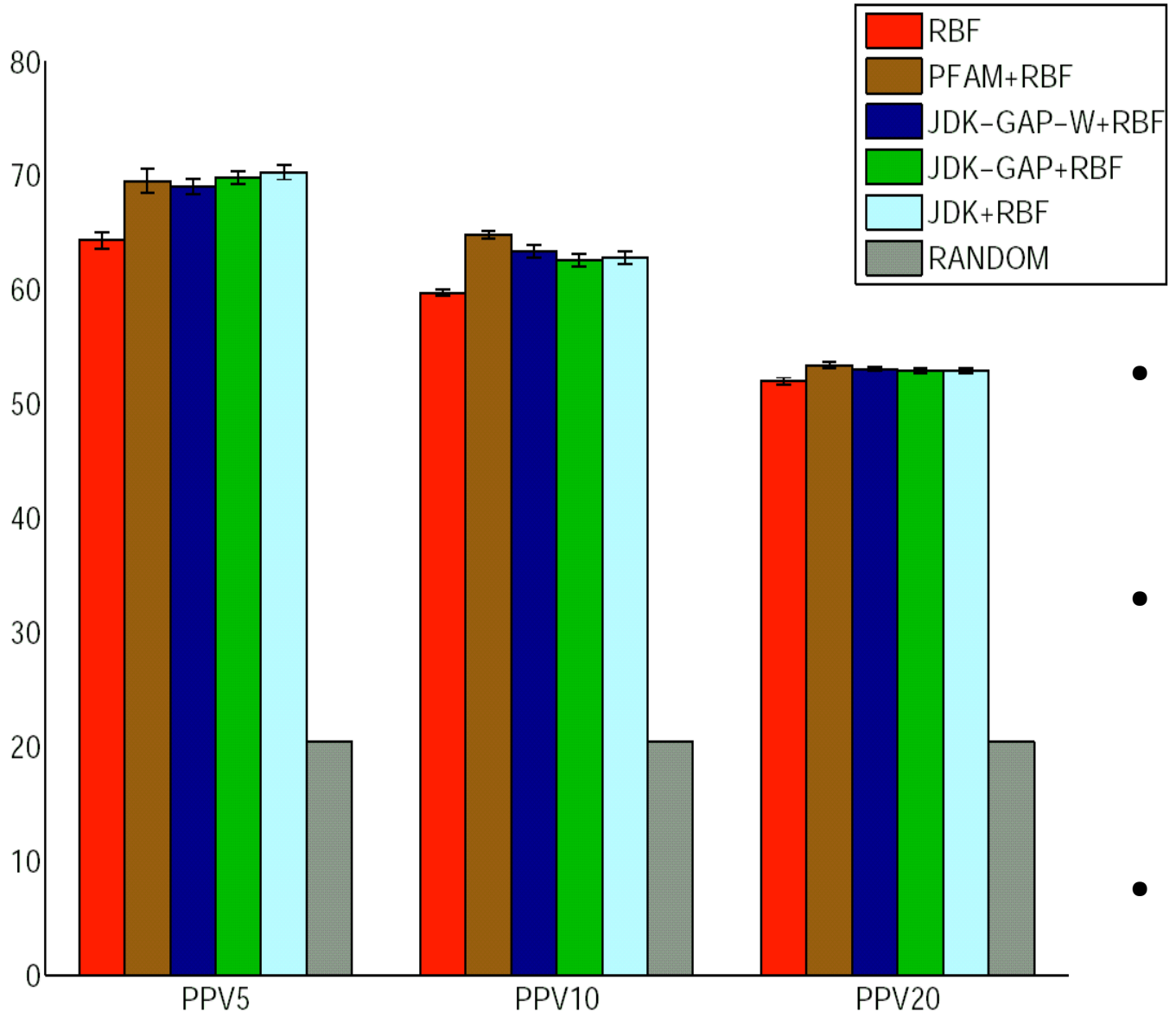
- Initial Experiments
  - sequence kernels
- Domain-based Kernels
  - independent and joint
- Performance of Domain-based Kernels

Sequence Kernels Alone



- Used 500 points
  - IDK: 30 min
  - JDK: 1 – 2.5 hrs
- Gap-expanded JDK comparable to Pfam
- JDK outperfoms IDK

Sequence Kernels Combined



- Combined kernels perform better
- Treatment of gaps by JDK does not affect performance
- JDK without gaps comparable to Pfam
  - no manual tuning!

Conclusion

- General domain-based kernels
  - useful whenever sequence similarity based on proximity to large sequence domains
- State-of-the-art results for protein essentiality using domain-based kernels
- Future Work: Generalize domain-based kernels using moments of counts