Technical University of Denmark DTU

Comparing methods for MHC peptide binding using PSSM, SMM, and ANN

Kamilla Kjærgaard Jensen, s112819

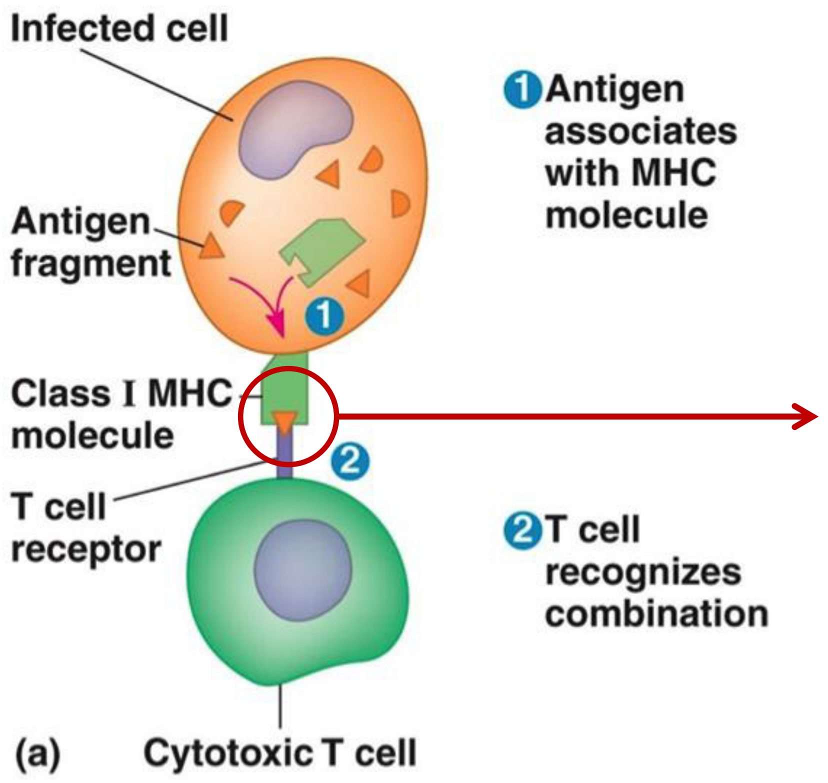
Marie Louise Jespersen, s152153

Patrick Zecchin, s150701

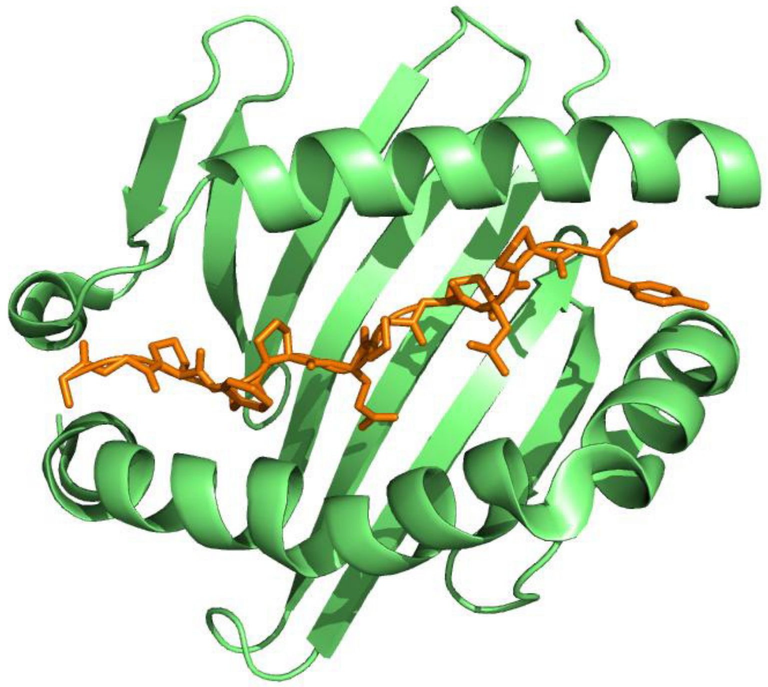
DTU Systems Biology
Department of Systems Biology

Introduction

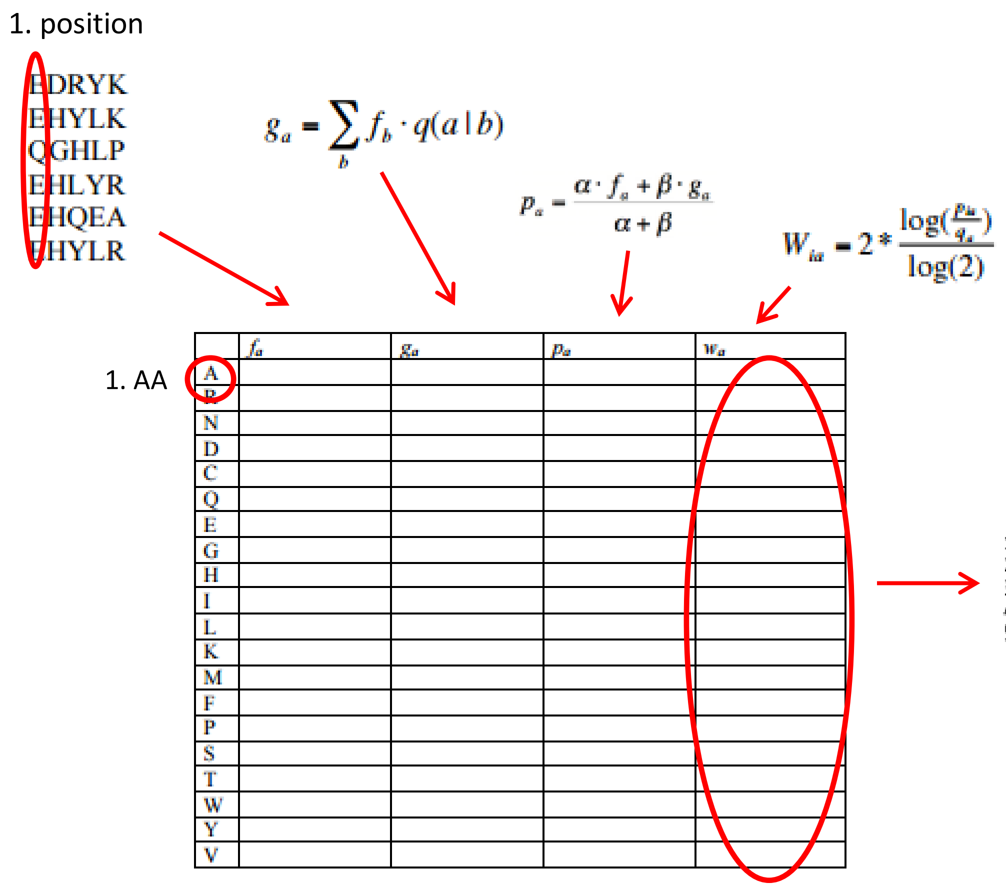
Overview



MHC peptide complex



Position Specific Scoring Matrix (PSSM)



EDRYK
EHYLK
QGHLP
EHLYR
EHQEA
EHYLR

$$g_a = \sum_b f_b \cdot q(a|b)$$

$$p_a = \frac{\alpha \cdot f_a + \beta \cdot g_a}{\alpha + \beta}$$

$$W_{ia} = 2 * \frac{\log(\frac{p_{ia}}{q_a})}{\log(2)}$$

	f_a	g_a	p_a	w_a
A				
R				
N				
D				
C				
Q				
E				
G				
H				
I				
L				
K				
M				
F				
P				
S				
T				
W				
Y				
V				

Optimisation of β:

- Small β for large datasets
- Large β for small datasets

```
       A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V
 1 Y  -2 -2 -2 -3 -2 -1 -2 -3  2 -1 -1 -2 -1  3 -3 -2 -2  2  7 -1
 2 L  -1 -2 -3 -4 -1 -2 -3 -4 -3  2  4 -2  2  0 -3 -2 -1 -2 -1  1
 3 P  -1 -2 -2 -2 -3 -2 -1 -2 -2 -3 -3 -1 -3 -4  8 -1 -1 -4 -3 -3
 4 S   1 -1  0 -1 -1  0  0 -1 -1 -3 -3  0 -2 -3 -1  5  1 -3 -2 -2
 5 C  -1 -4 -3 -4  9 -3 -4 -3 -3 -2 -2 -3 -2 -3 -3 -1 -1 -3 -3 -1
```

Calculate scores using the PSSM for peptides in the test set

Stabilization Matrix Method (SMM)

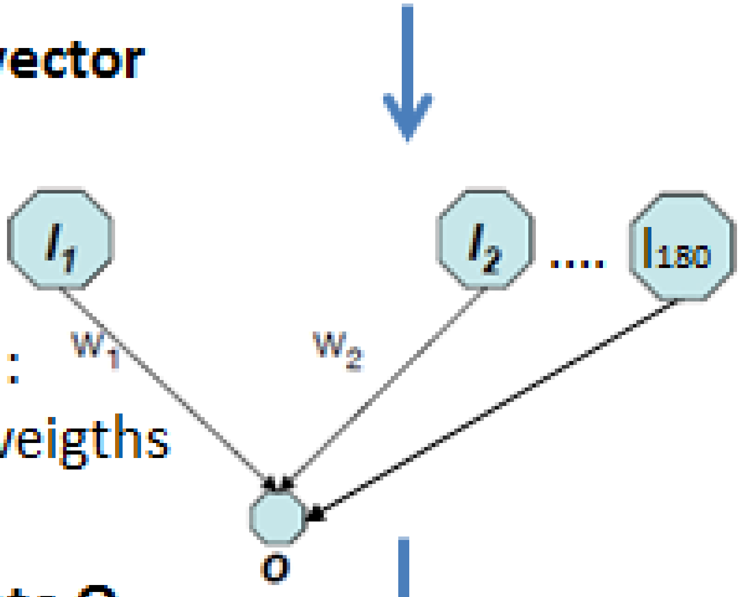
Input vector: Sparse encoding

Length = peptide length * 20

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Step 1: Input vector

1. round:
random weigths



Step 2: Calculate O

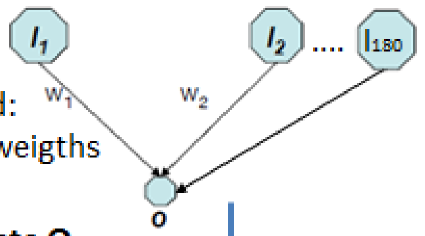
$$O = \sum_i I_i \cdot w_i$$

SMM – gradient descent

Input vector: Sparse encoding
Length = peptide length * 20

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Step 1: Input vector



1. round:
random weigths

Step 2: Calculate O

$$O = \sum_i I_i \cdot w_i$$

Step 3: Calculate E

Gradient descent

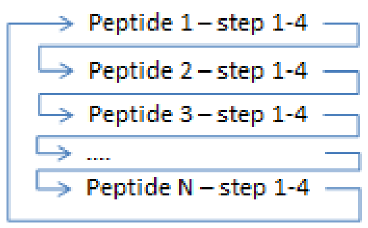
$$E_{\text{per target}} = \tfrac{1}{2} \cdot (O - t)^2 + \frac{\lambda}{N} \sum_i w_i^2$$

Step 4: Update weights

$$\frac{\partial E}{\partial w_i} = (O - t) \cdot I_i + \frac{2 \cdot \lambda}{N} \cdot w_i$$

$$\Delta w_i = -\varepsilon \cdot \frac{\partial E}{\partial w_i}$$

Gradient descent



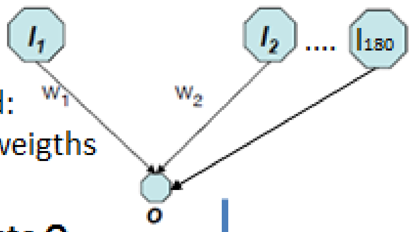
SMM – Monte Carlo

Input vector: Sparse encoding
Length = peptide length * 20

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Step 1: Input vector

1. round: random weigths



Step 2: Calculate O

$$O = \sum_i I_i \cdot w_i$$

Step 3: Calculate E

Monte Carlo

$$E = \frac{1}{2} \cdot \sum_i (O_i - t_i)^2 + \lambda \cdot \sum_i w_i^2$$

Step 4: Update weights

Update weights if the move is accepted

Monte Carlo

Calculates global error

Peptide 1 ... N - Step 1-4

$$P(accept) = \min\left(1, \left(e^{\frac{-\Delta E}{T}}\right)\right)$$

If $\Delta E \leq 0$
- Accept move

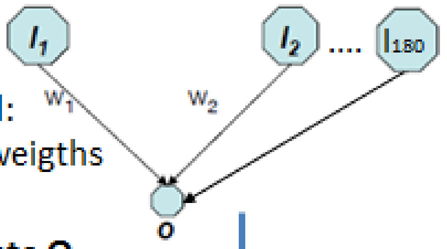
If $\Delta E > 0$
- Pick a random number i $(0 \leq i \leq 1)$
- Accept if $\Delta E \leq i$

SMM - output

Input vector: Sparse encoding
Length = peptide length * 20

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Step 1: Input vector



1. round: random weigths

Step 2: Calculate O

$$O = \sum_i I_i \cdot w_i$$

Step 3: Calculate E

Gradient descent

$$E_{\text{per target}} = \tfrac{1}{2} \cdot (O - t)^2 + \frac{\lambda}{N} \sum_i w_i^2$$

Monte Carlo

$$E = \tfrac{1}{2} \cdot \sum_i (O_i - t_i)^2 + \lambda \cdot \sum_i w_i^2$$

Step 4: Update weights

$$\frac{\partial E}{\partial w_i} = (O - t) \cdot I_i + \frac{2 \cdot \lambda}{N} \cdot w_i$$

$$\Delta w_i = -\varepsilon \cdot \frac{\partial E}{\partial w_i}$$

Update weights if mc move is accepted

w 1	w 2	w 3	w 4	w 5	w 6	w 7	w 8	w 9	w1 0	w 11	w 12	w 13	w 14	w 15	w 16	w 17	w 18	w 19	w 20

Vector of final weights
Length = input vector

Output = weight matrix

		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
1	Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
2	L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
3	P	-1	-2	-2	-2	-3	-2	-1	-2	-2	-3	-3	-1	-3	-4	8	-1	-1	-4	-3	-3
4	S	1	-1	0	-1	-1	0	0	-1	-1	-3	-3	0	-2	-3	-1	5	1	-3	-2	-2
5	C	-1	-4	-3	-4	9	-3	-4	-3	-3	-2	-2	-3	-2	-3	-3	-1	-1	-3	-3	-1

Calculate affinities using the SMM for peptides in the test set

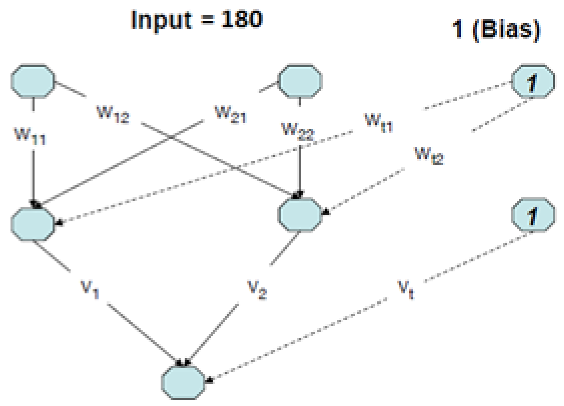
Artificial Neural Network (ANN)

Input vector: Sparse or BLOSUM encoding
Length = peptide length * 20

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Step 1: Input vector

1. round:
random weigths



Step 2: Calculate output of next neuron (H or O)

$$o = \sum x_i \cdot w_i \quad O = g(o)$$

Step 3: Calculate E

$$E = \tfrac{1}{2} \cdot (O - t)^2$$

Step 4: Update weights

$$\Delta w_j = -\varepsilon \cdot \frac{\partial E}{\partial w_j}; \Delta v_{jk} = -\varepsilon \cdot \frac{\partial E}{\partial v_{jk}}$$

$$\frac{\partial E}{\partial w_j} = (O - t) \cdot g'(o) \cdot H_j$$

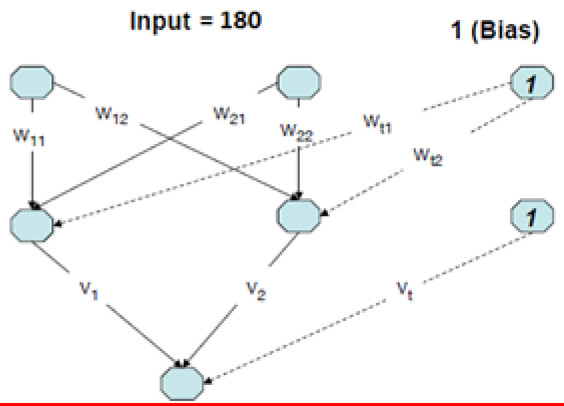
$$\frac{\partial E}{\partial v_{jk}} = g'(h_j) \cdot I_k \cdot (O - t) \cdot g'(o) \cdot w_j$$

ANN – forward and back

Input vector: Sparse or BLOSUM encoding
Length = peptide length * 20

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Step 1: Input vector



1. round:
random weigths

Forward

Step 2: Calculate output of next neuron (H or O)

$$o = \sum x_i \cdot w_i \quad O = g(o)$$

Back propagation

Step 3: Calculate E

$$E = \tfrac{1}{2} \cdot (O - t)^2$$

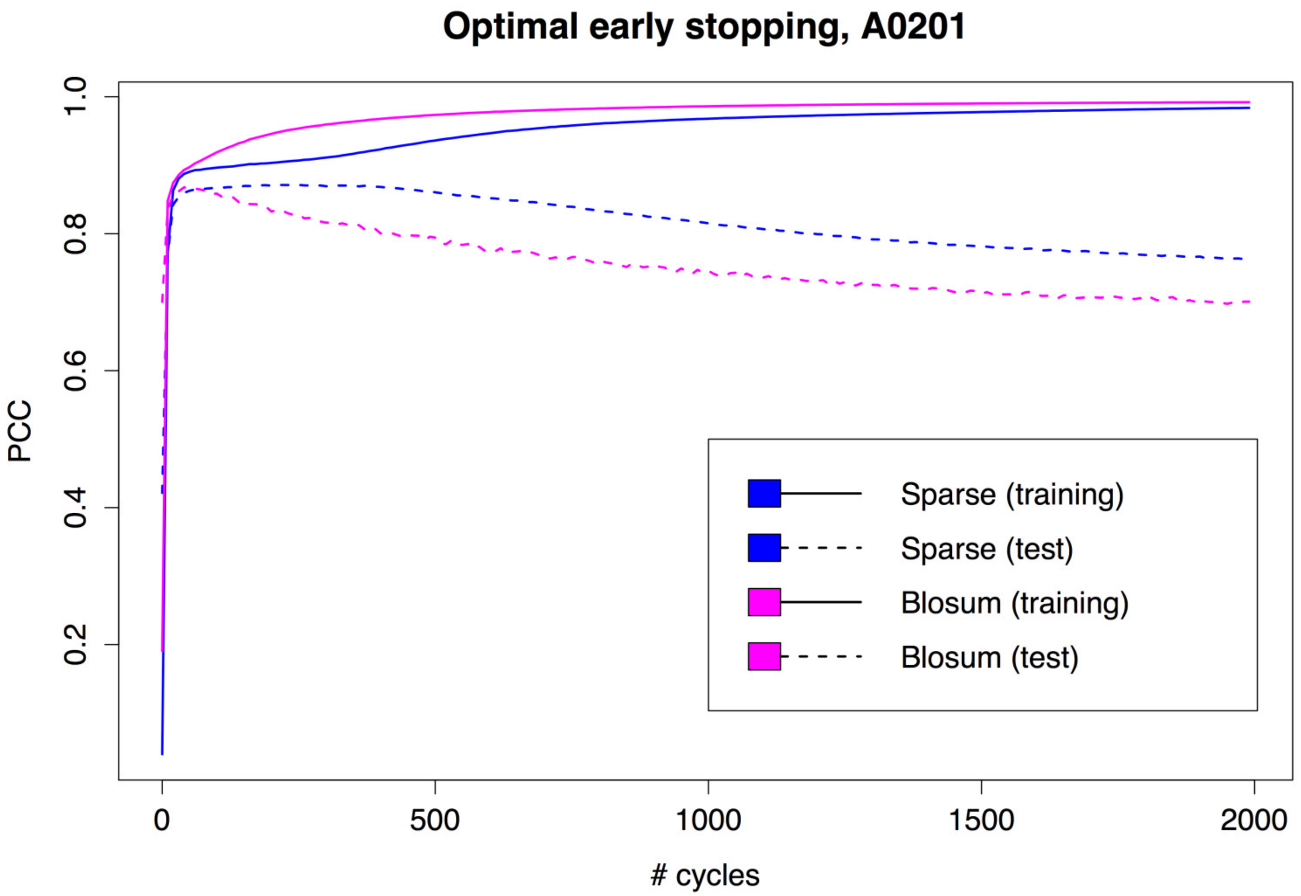
Step 4: Update weights

$$\Delta w_j = -\varepsilon \cdot \frac{\partial E}{\partial w_j} ; \Delta v_{jk} = -\varepsilon \cdot \frac{\partial E}{\partial v_{jk}}$$

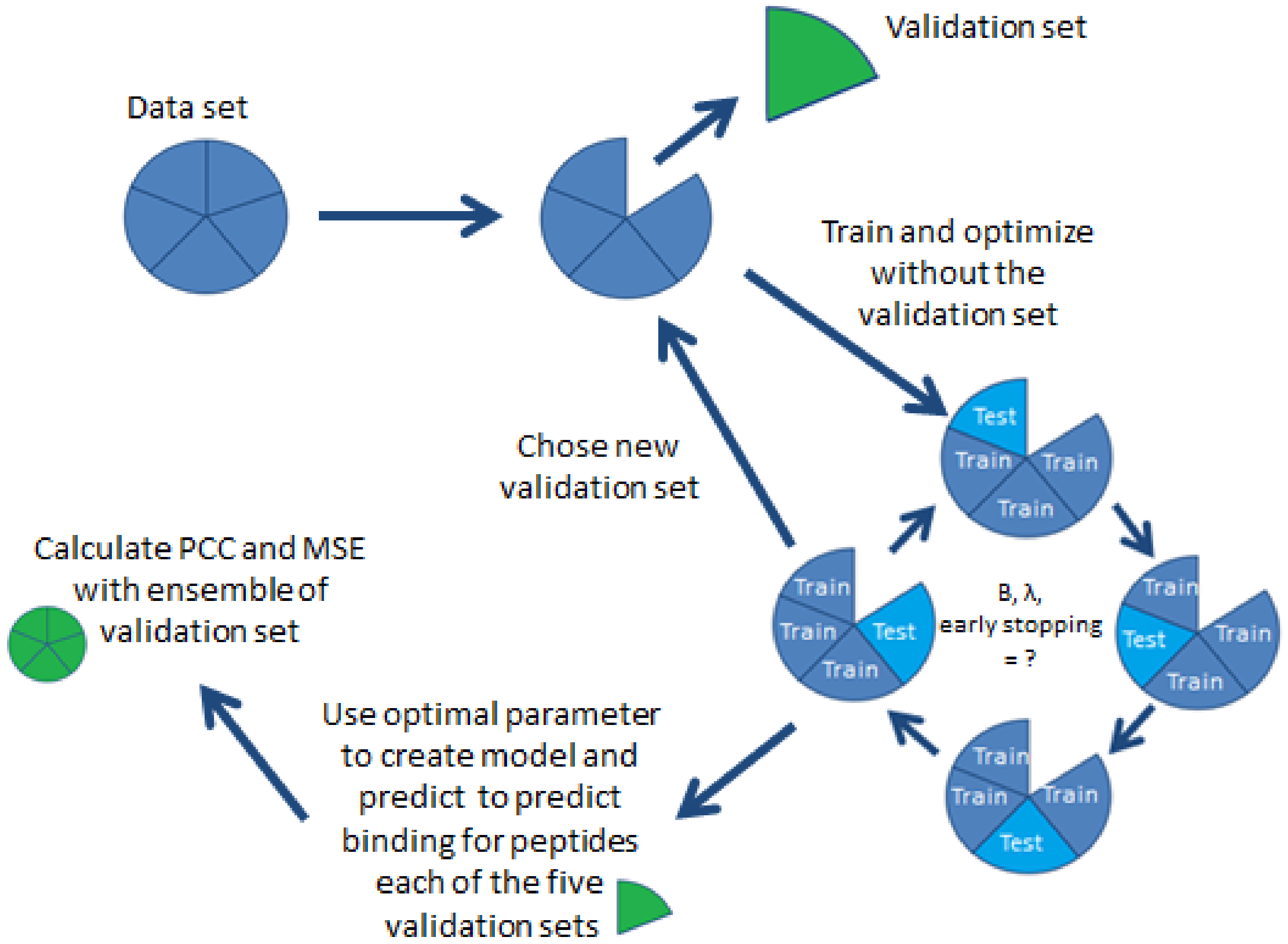
$$\frac{\partial E}{\partial w_j} = (O - t) \cdot g'(o) \cdot H_j$$

$$\frac{\partial E}{\partial v_{jk}} = g'(h_j) \cdot I_k \cdot (O - t) \cdot g'(o) \cdot w_j$$

ANN – early stopping

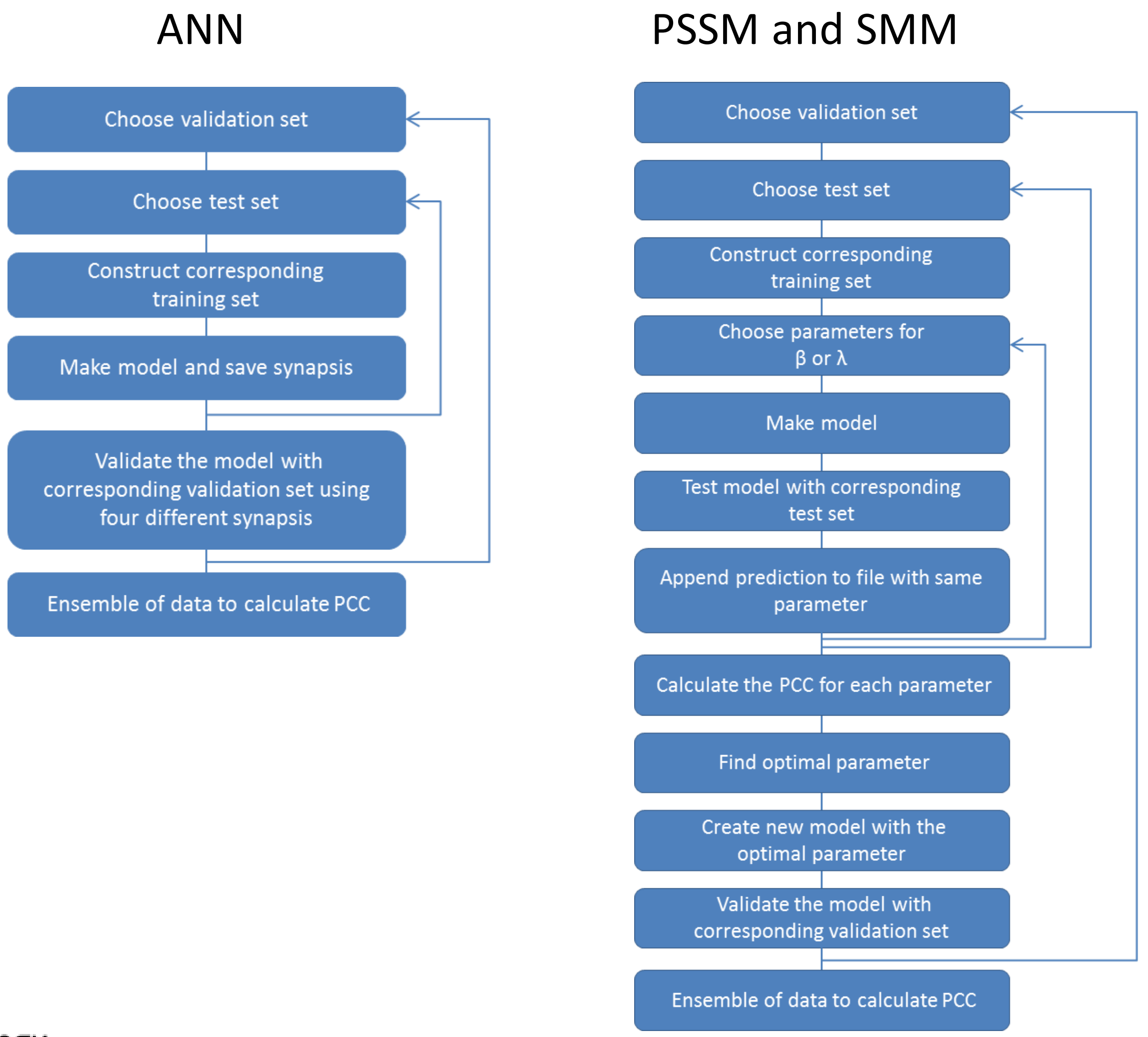


Cross-validation


Validation set
Data set
Train and optimize without the validation set
Test
Train
Train
Train
Chose new validation set
Calculate PCC and MSE with ensemble of validation set
Train
Train
Test
Train
B, λ, early stopping = ?
Train
Test
Train
Train
Train
Train
Train
Test
Use optimal parameter to create model and predict to predict binding for peptides each of the five validation sets

Technical University of Denmark

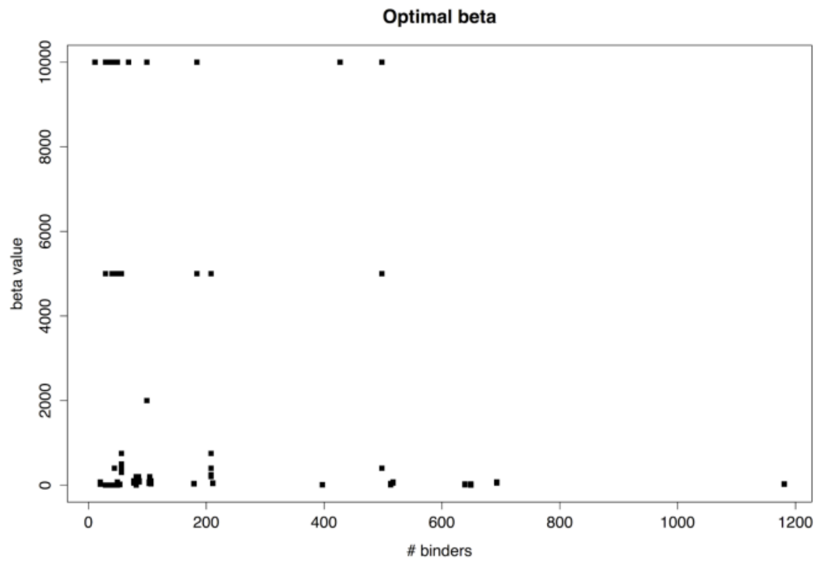
Program


ANN
Choose validation set
Choose test set
Construct corresponding training set
Make model and save synapsis
Validate the model with corresponding validation set using four different synapsis
Ensemble of data to calculate PCC
PSSM and SMM
Choose validation set
Choose test set
Construct corresponding training set
Choose parameters for β or λ
Make model
Test model with corresponding test set
Append prediction to file with same parameter
Calculate the PCC for each parameter
Find optimal parameter
Create new model with the optimal parameter
Validate the model with corresponding validation set
Ensemble of data to calculate PCC

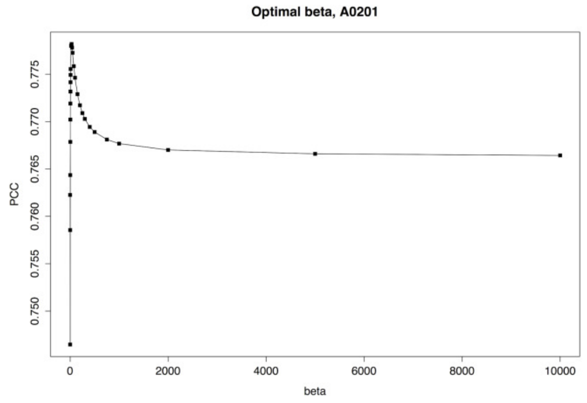
Technical University of Denmark DTU

Optimal parameters

- Expected:
  - Small dataset: large β
  - Large dataset: small β
- Uexpected:
  - Small dataset: small β
    - Sequence variance in dataset
    - Restricted binding motif

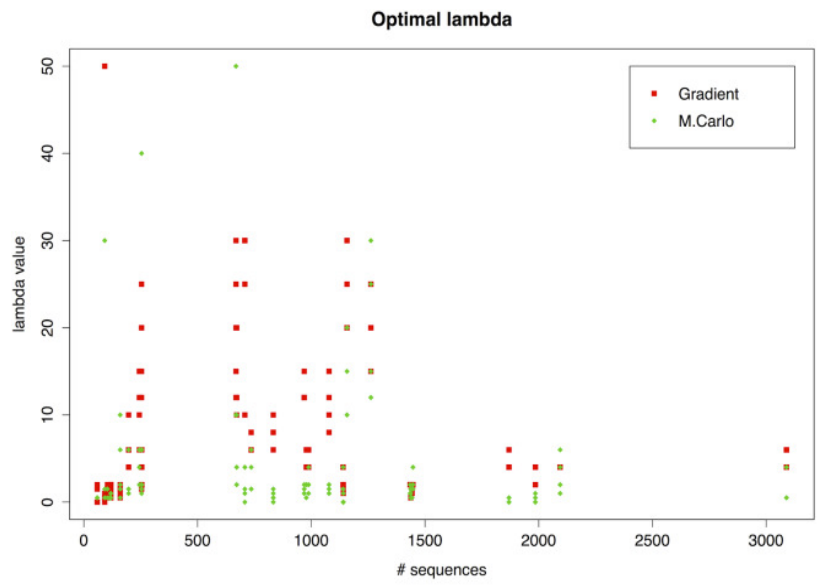


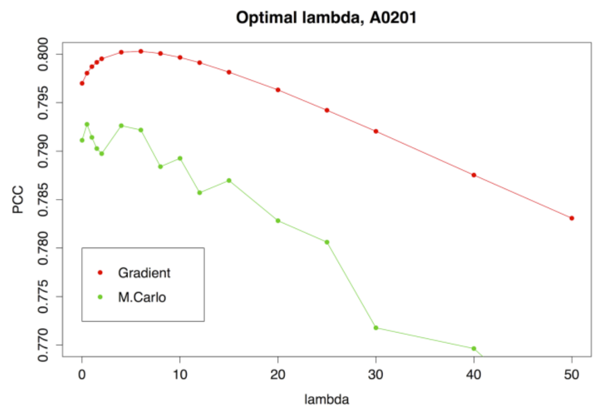
$$p_{ia} = \frac{\alpha \cdot f_{ia} + \beta \cdot g_{ia}}{\alpha + \beta}$$



Optimal parameters

- Large λ
  - Small dataset
  - Gradient descent based SMM



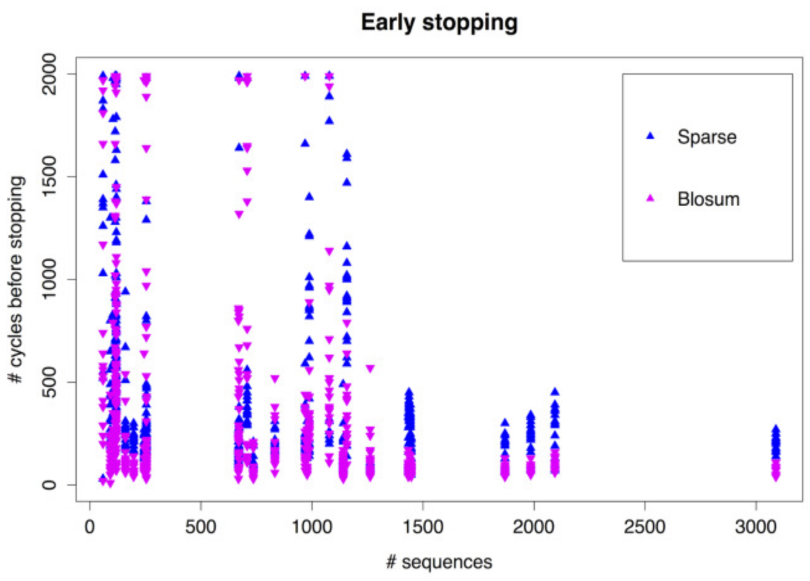


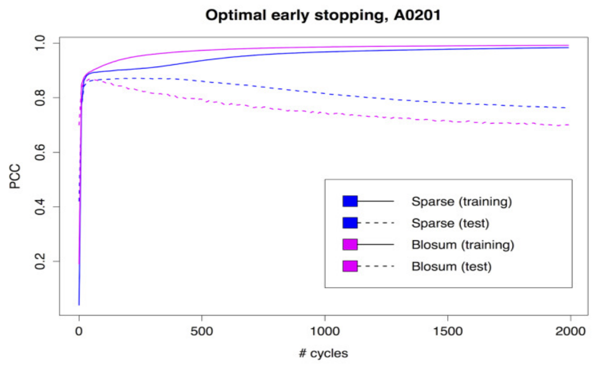
$$E_{per\ target} = \frac{1}{2} \cdot (O - t)^2 + \frac{\lambda}{N} \cdot \sum_i w_i^2$$

$$E = \frac{1}{2} \cdot \sum_i (O_i - t_i)^2 + \lambda \cdot \sum_i w_i^2 \ .$$

Optimal parameters

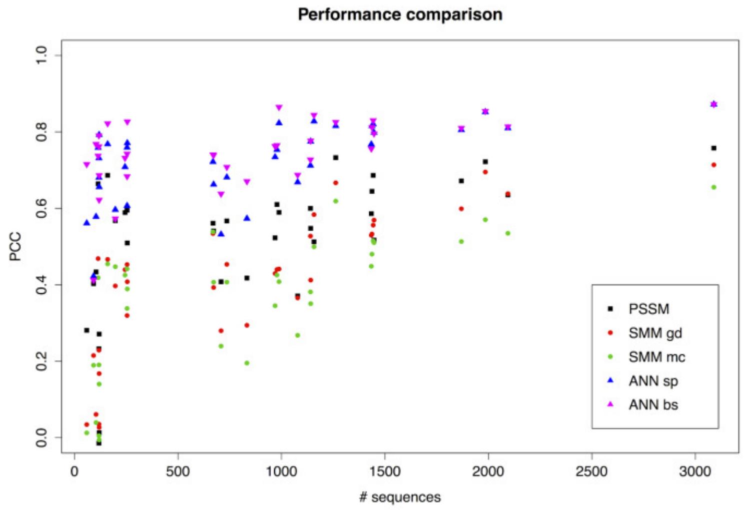
- Small dataset:
  large amount of cycles
  - Low information content
- BLOSUM encoding:
  small amount of cycles
  - High information content

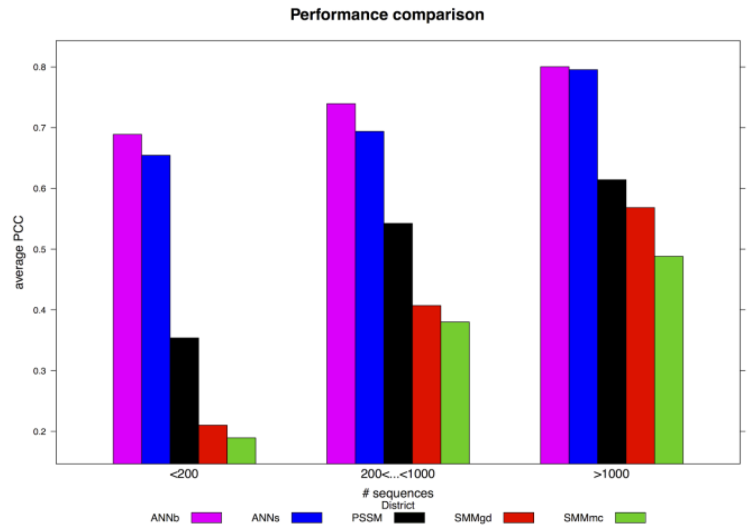




Performance comparison

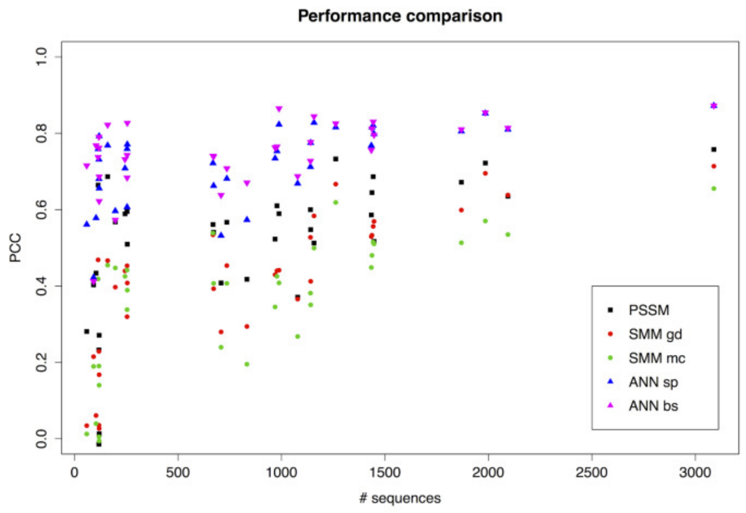
- Very low performance: small datasets
  - Not enought data to learn
- PSSM/SMM
  - Large increase in performace with the size of the datasets

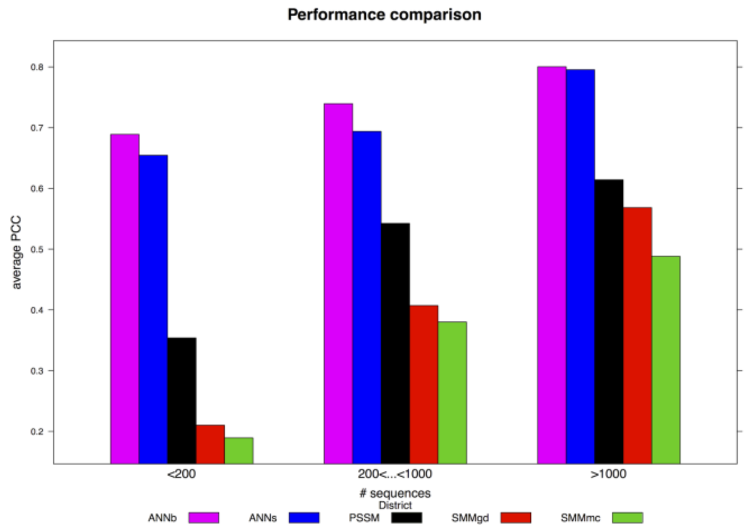




Performance comparison

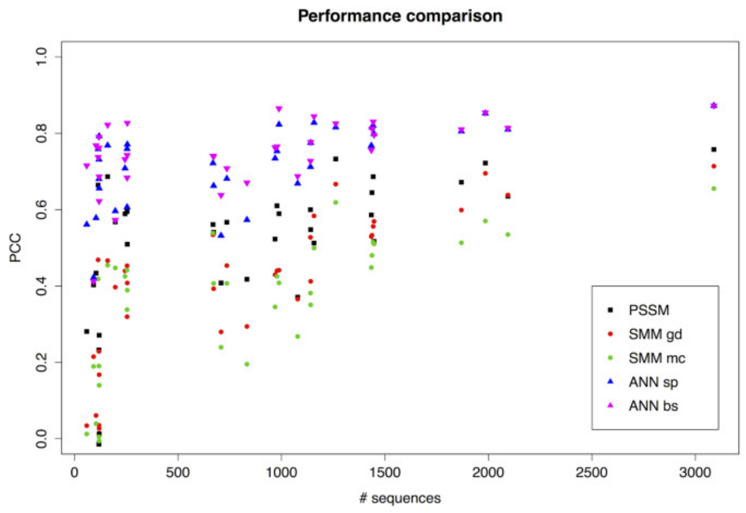
- ANN
  - Good performance on all datasets
  - BLOSUM encodig generally best
  - Sparse vs. BLOSUM: no pattern in dataset size

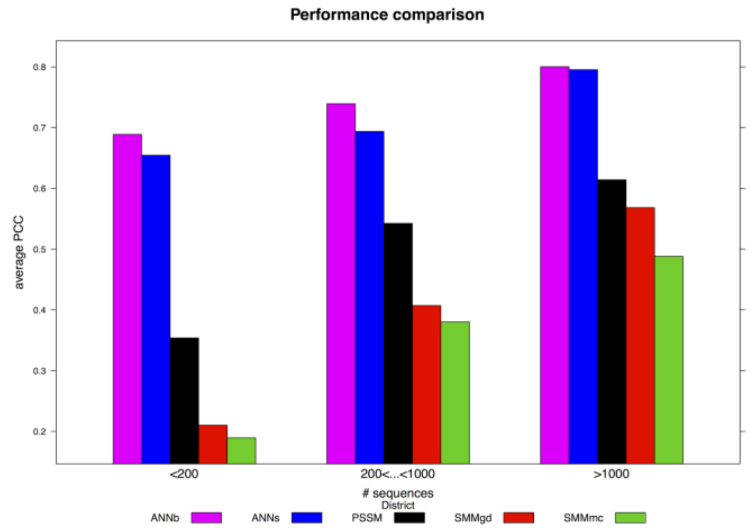




Performance comparison

- SMM underformance
  - Sparse vs. BLOSUM
  - Number of itereations for Monte Carlo
  - Divide by variance in sequences - Hobohm





Conclusion

- ANN > PSSM > SMM
- Performance generally better on larger datasets

Thank you for your attention