INSTITUTE *for* COMPUTATIONAL MEDICINE

Deep Learning

An introduction for proteins and DNA/RNA

ROHIT BHATTACHARYA

APRIL 16, 2018

DEPARTMENT OF COMPUTER SCIENCE

Table of contents

Goal

Roadmap

- Lay out the framework required for any machine learning algorithm
- Understand the framework in the context of neural networks
- Understand the limitations and strengths of network architectures
- Understand all of this in the context of protein/nucleotide sequences

Representation

Motivation

- Think back to the ESE sequences
- How did you analyze the “fitness” of these sequences

AGAAGA

ACGACT

AATCCA

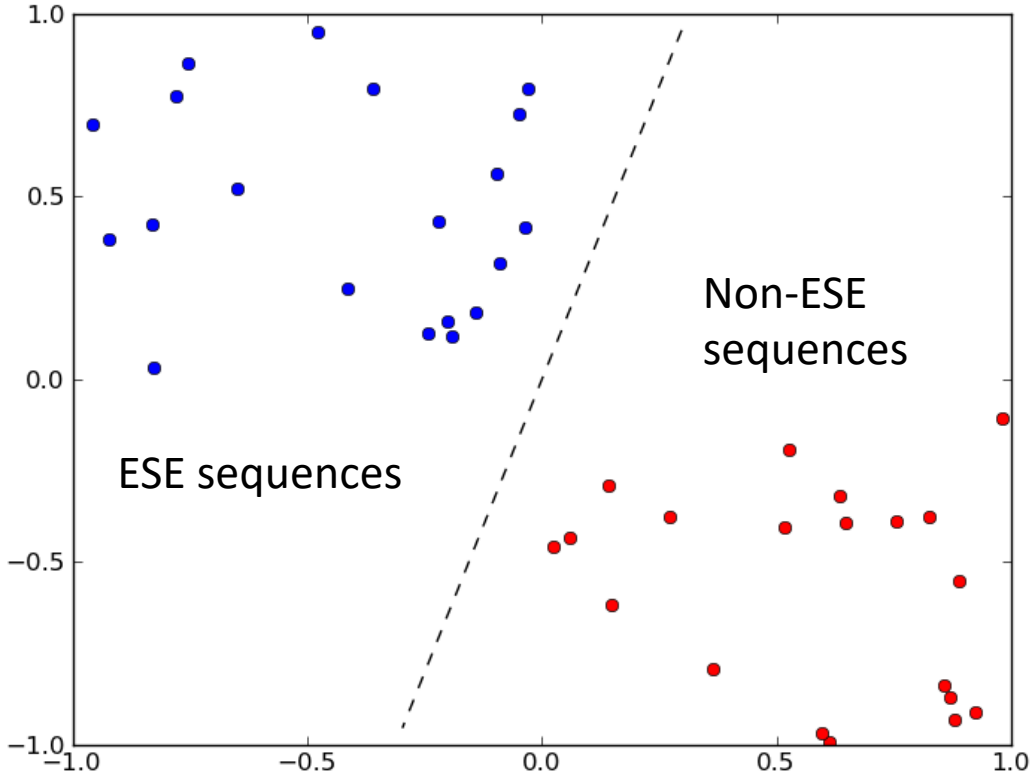
Position Specific Scoring Matrices

- Built a PSSM!
- Each entry corresponded to the probability of seeing a nucleotide at the given position
- What are the limitations/strengths of this?

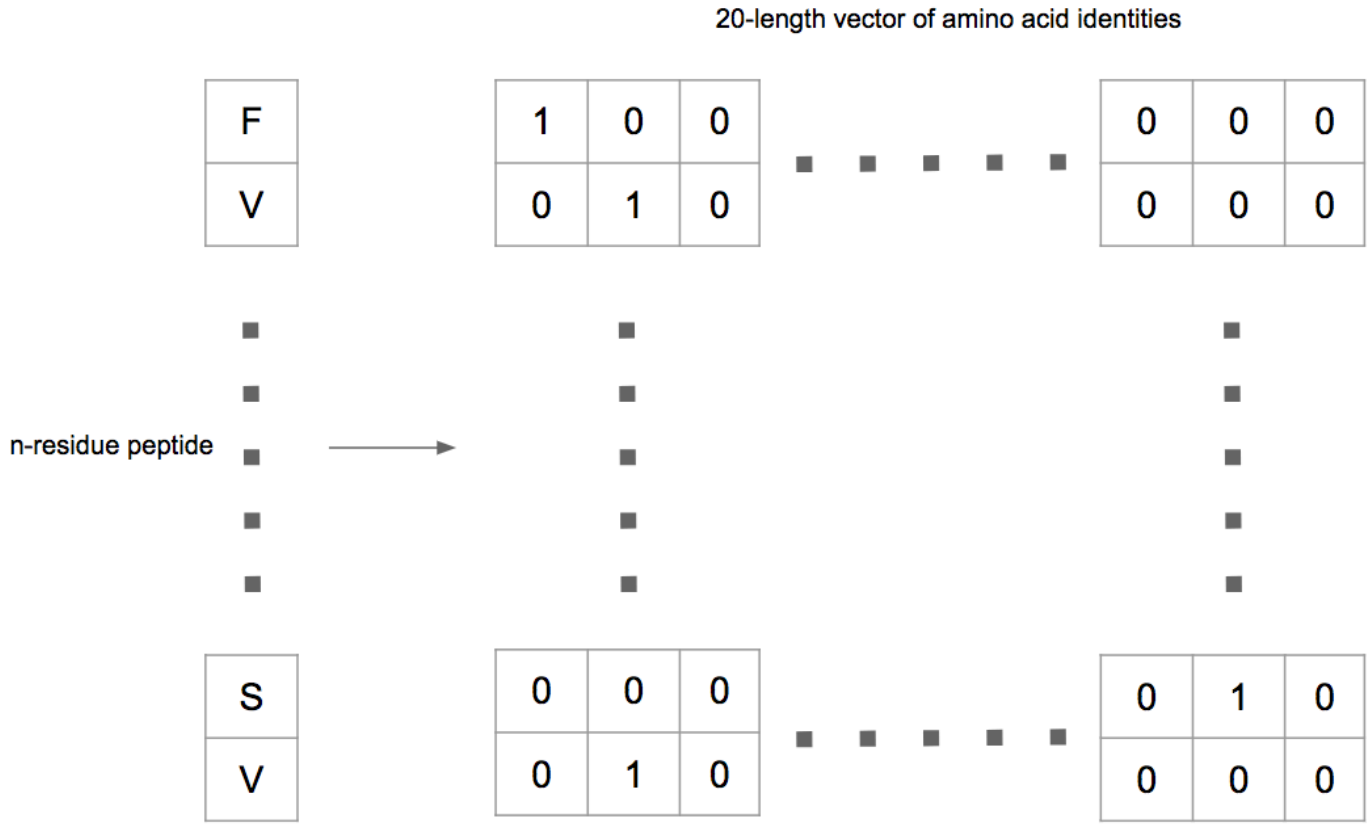
Position Specific Scoring Matrices

- Built a PSSM!
- Each entry corresponded to the probability of seeing a nucleotide at the given position
- What are the limitations/strengths of this?
  - Sequences must be uniform length
  - Assume all positions to be independent of each other
  - Simple model (Occam's razor is real!)

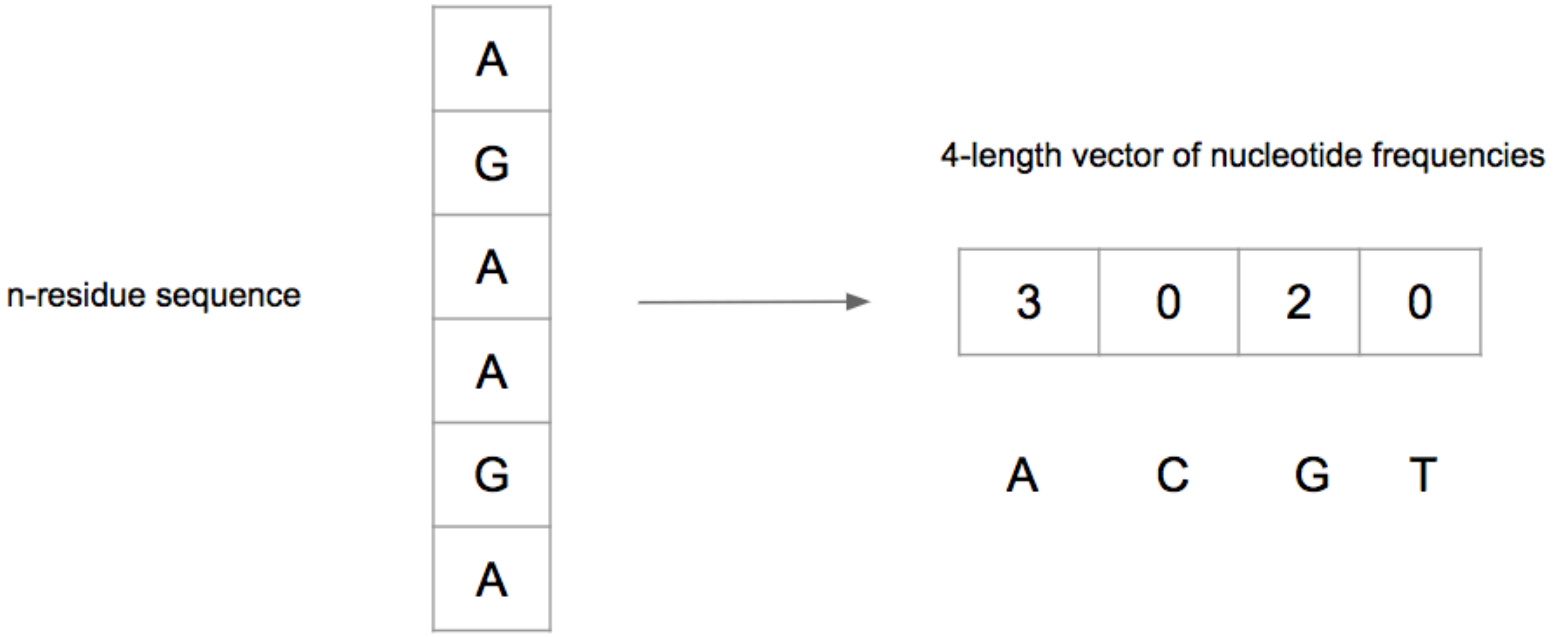
Classification



One-Hot Encoding of Peptides



Bag of Amino Acids/Nucleotides



Pairwise distances/similarities

- Not an explicit representation in vector space
  - But recall SVMs only need the pairwise similarities
- Can be achieved through pairwise sequence alignment
  - Needleman Wunsch or Smith Waterman
- Careful that these pairwise distances/similarities are **metric**
  - $d(x, y) = d(y, x)$
  - $d(x, y) \geq 0$
  - $d(x, y) + d(y, z) > d(x, z)$

Pairwise distances/similarities

	A	T	C	G
A	5	-4	-4	-4
T	-4	5	-4	-4
C	-4	-4	5	-4
G	-4	-4	-4	5

BLAST Similarity Matrix

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	5	-2	-1	-2	-1	-1	-1	0	-2	-1	-2	-1	-1	-3	-1	1	0	-3	-2	0
R	-2	7	-1	-2	-4	1	0	-3	0	-4	-3	3	-2	-3	-3	-1	-1	-3	-1	-3
N	-1	-1	7	2	-2	0	0	0	1	-3	-4	0	-2	-4	-2	1	0	-4	-2	-3
D	-2	-2	2	8	-4	0	2	-1	-1	-4	-4	-1	-4	-5	-1	0	-1	-5	-3	-4
C	-1	-4	-2	-4	13	-3	-3	-3	-3	-2	-2	-3	-2	-2	-4	-1	-1	-5	-3	-1
Q	-1	1	0	0	-3	7	2	-2	1	-3	-2	2	0	-4	-1	0	-1	-1	-1	-3
E	-1	0	0	2	-3	2	6	-3	0	-4	-3	1	-2	-3	-1	-1	-1	-3	-2	-3
G	0	-3	0	-1	-3	-2	-3	8	-2	-4	-4	-2	-3	-4	-2	0	-2	-3	-3	-4
H	-2	0	1	-1	-3	1	0	-2	10	-4	-3	0	-1	-1	-2	-1	-2	-3	2	-4
I	-1	-4	-3	-4	-2	-3	-4	-4	-4	5	2	-3	2	0	-3	-3	-1	-3	-1	4
L	-2	-3	-4	-4	-2	-2	-3	-4	-3	2	5	-3	3	1	-4	-3	-1	-2	-1	1
K	-1	3	0	-1	-3	2	1	-2	0	-3	-3	6	-2	-4	-1	0	-1	-3	-2	-3
M	-1	-2	-2	-4	-2	0	-2	-3	-1	2	3	-2	7	0	-3	-2	-1	-1	0	1
F	-3	-3	-4	-5	-2	-4	-3	-4	-1	0	1	-4	0	8	-4	-3	-2	1	4	-1
P	-1	-3	-2	-1	-4	-1	-1	-2	-2	-3	-4	-1	-3	-4	10	-1	-1	-4	-3	-3
S	1	-1	1	0	-1	0	-1	0	-1	-3	-3	0	-2	-3	-1	5	2	-4	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	2	5	-3	-2	0
W	-3	-3	-4	-5	-5	-1	-3	-3	-3	-3	-2	-3	-1	1	-4	-4	-3	15	2	-3
Y	-2	-1	-2	-3	-3	-1	-2	-3	2	-1	-1	-2	0	4	-3	-2	-2	2	8	-1
V	0	-3	-3	-4	-1	-3	-3	-4	-4	4	1	-3	1	-1	-3	-2	0	-3	-1	5

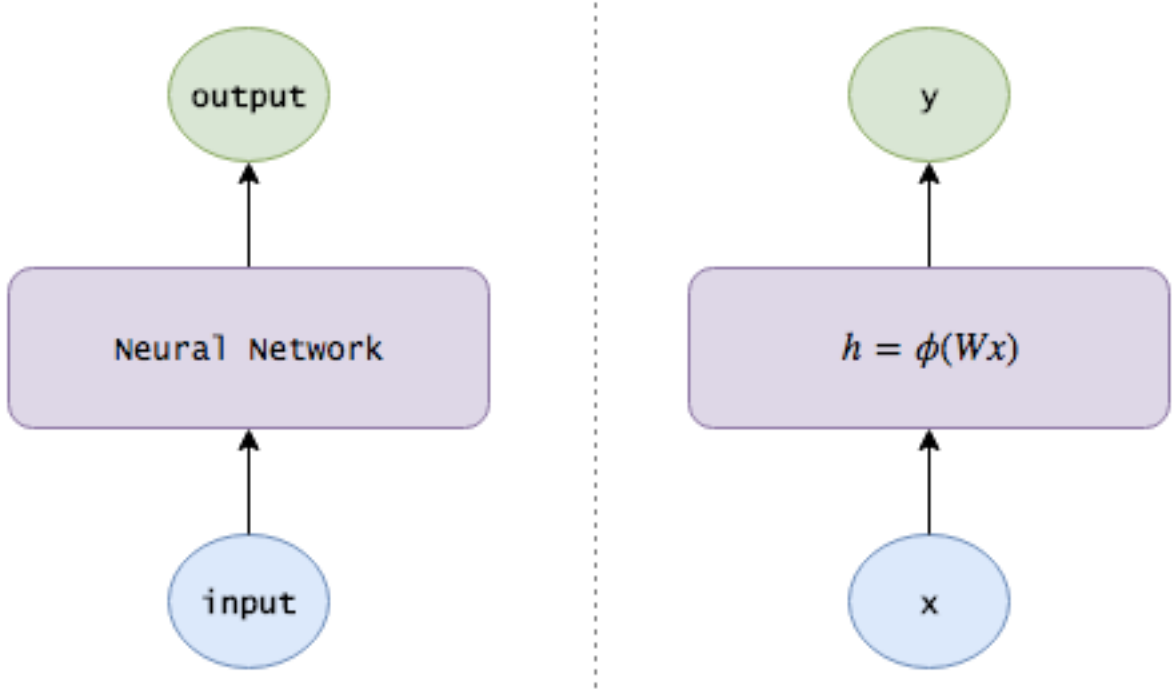
BLOSUM 50

Prediction

Forward Pass

- Some function ***f(x)*** that maps our *d* dimensions to scores for each class
- ***f(x)*** is parametrized by weights given by matrix ***W***
- Given *d* dimensions and *k* classes, $\boldsymbol{W}_{\boldsymbol{d \times k}}$
- Predictions are given by $\boldsymbol{W^T x + b}$ (or some chained version of this)
- This is called the forward pass of neural networks
- For the rest of the talk, assume ***x*** to be a one-hot encoded sequence

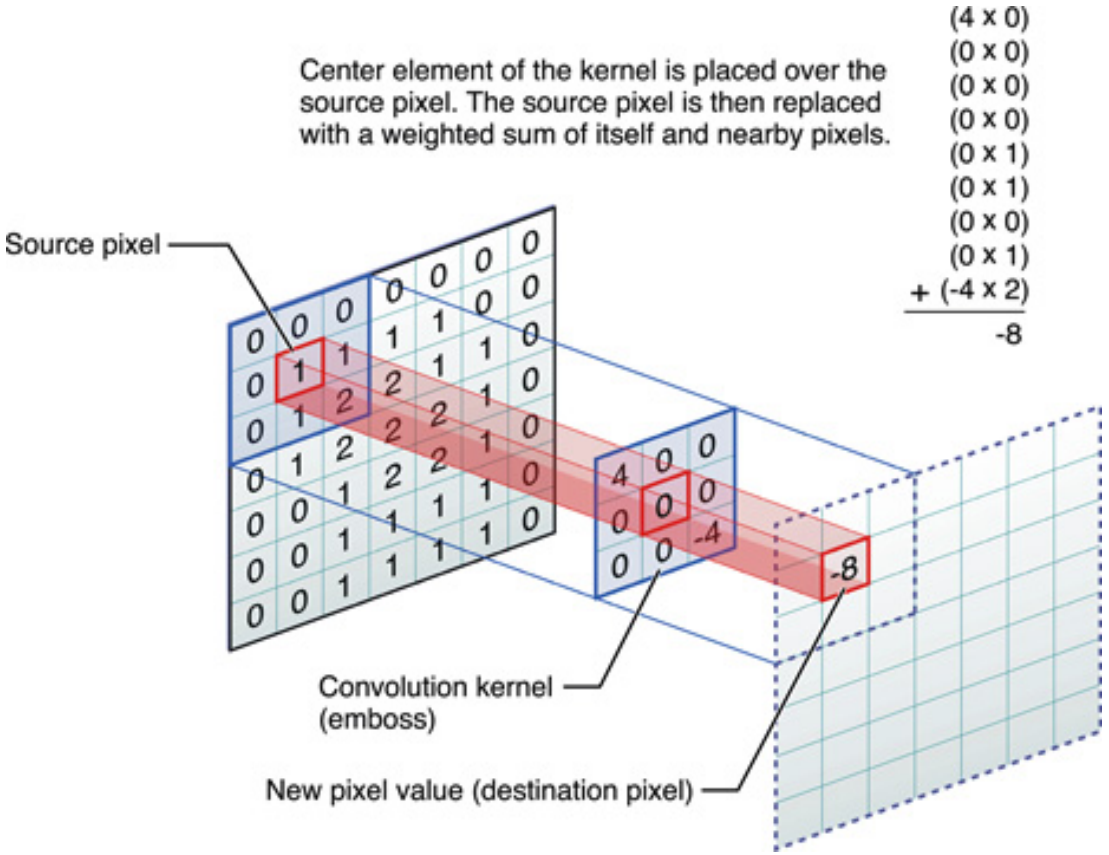
Fully Connected Neural Nets



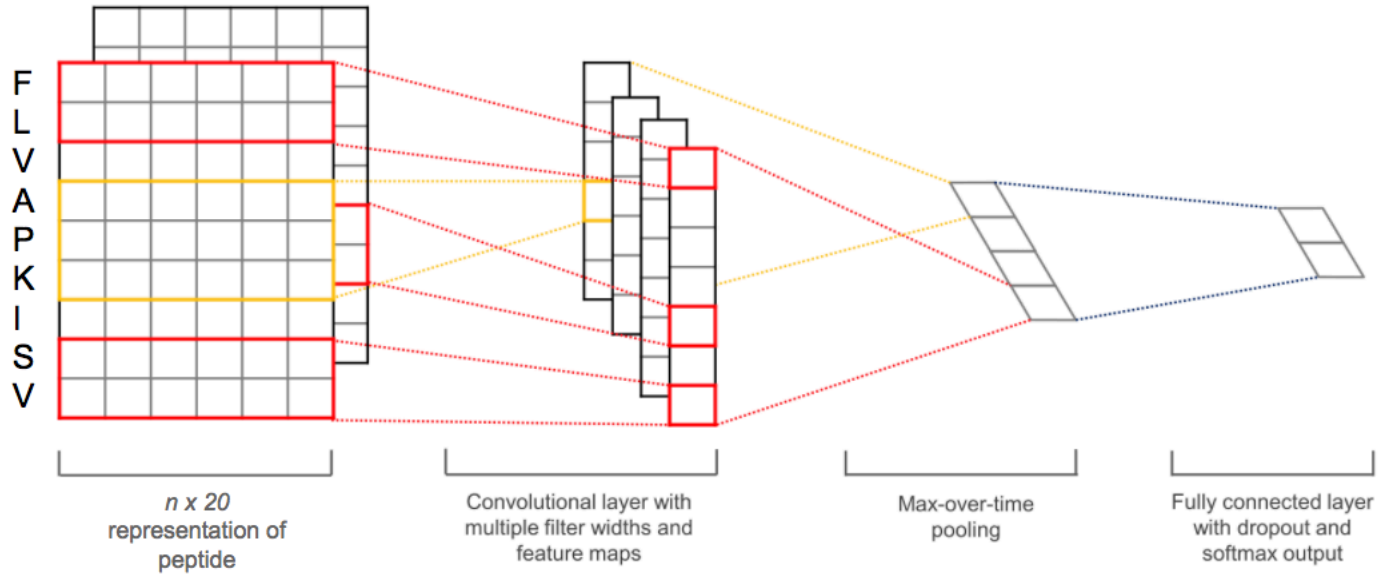
Pros/Cons

+ Easy to build and train
+ Preserve positional information of residues
- Fixed length inputs require various hacks to make inputs the same length
- No explicit encoding of interdependence between residues

2D Convolutional Neural Nets



1D Convolutional Neural Nets



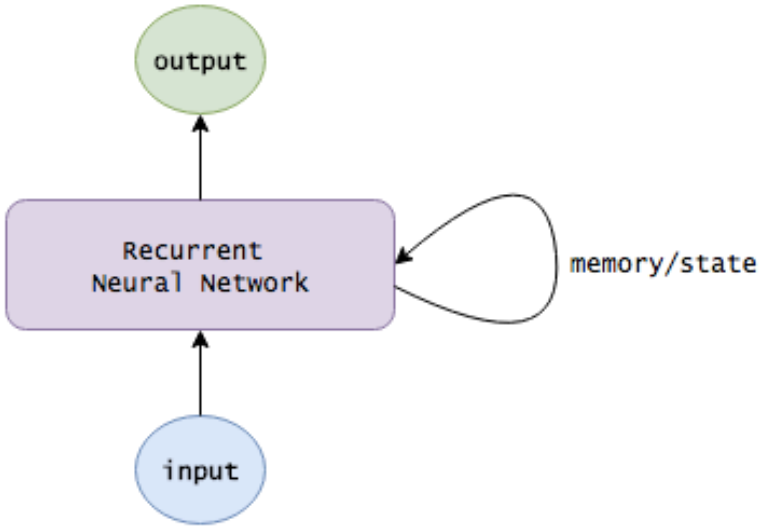
Pros/Cons

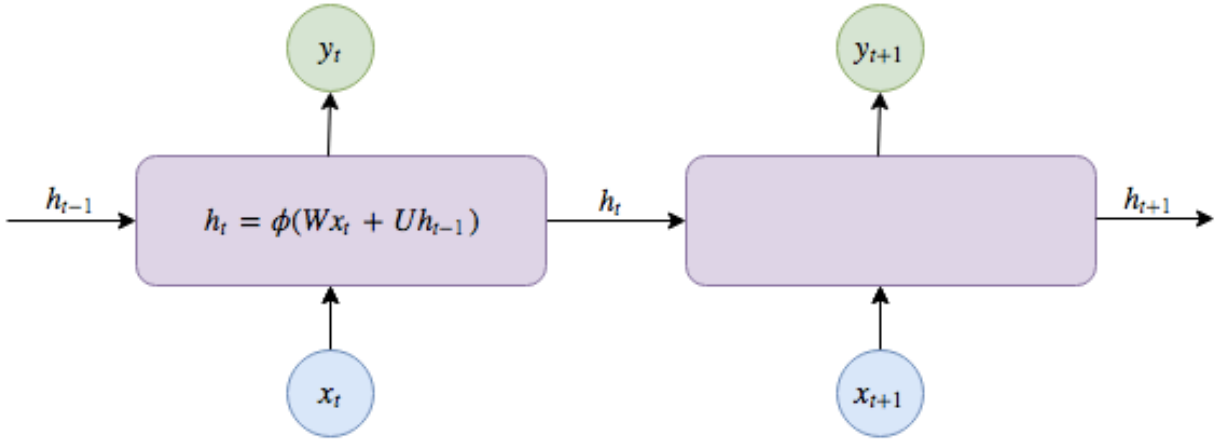
+ Considers motifs rather than individual residues
- Lose positional information

Side Note: 3D CNNs

- There's been some excitement surrounding 3D convolutions on protein structure
- Recent papers such as those by Ragoza et al (2017) show the use of 3D convolutions for scoring protein-ligand interactions

Recurrent Neural Nets

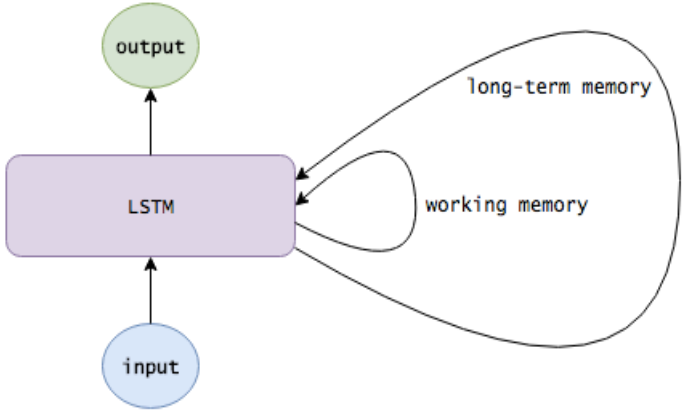


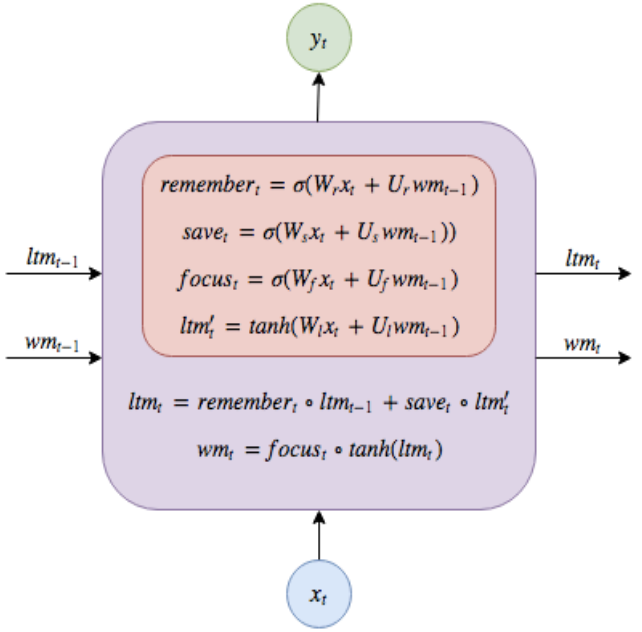


Pros/Cons

+ Encodes interdependence between residues
+ Interdependence includes long-range dependencies (theoretically)
+ Handles variable length sequences naturally
+ Maintains a sense of positional information
- Quite unstable to train and converge

Long Short-Term Memory Networks





Pros/Cons

+ Encodes interdependence between residues
+ Interdependence includes long-range dependencies (actually)
+ Handles variable length sequences naturally
+ Maintains a sense of positional information
- Harder to train than the CNN/FC but much easier than a vanilla RNN

Loss

Probabilities

- Predictions from the net are unnormalized scores
- We'd like to obtain normalized probabilities such that
  - $\mathbf{0} \leq \boldsymbol{p_k} \leq \mathbf{1}$
- $\boldsymbol{p_k} = \frac{e^{f_k}}{\sum_j e^{f_j}}$
- This is the softmax function

Data loss

- The loss for an example ***i*** is given by

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

- The total loss over the data for all examples is

$$\frac{1}{N}\sum_i L_i$$

Regularization loss

- Notice that an infinite number of $\boldsymbol{W}$ can minimize our loss
- Concretely, for any $\boldsymbol{W}$ that minimizes our loss so does $\boldsymbol{\lambda W}$ for $\boldsymbol{\lambda > 1}$
- In order to constrain this search we apply a **regularization loss**

$$\frac{1}{2}\sum_{k}\sum_{l} W_{k,l}^{2}$$

Total loss

- Thus the total loss is given by

$$L = \underbrace{\frac{1}{N}\sum_{i} L_i}_{\text{data loss}} + \underbrace{\frac{1}{2}\lambda \sum_{k}\sum_{l} W_{k,l}^2}_{\text{regularization loss}}$$

Minimization

Complete random search

- Each iteration pick a new random ***W***
- If the loss is lower than the previously seen best loss, use this as your new ***W***

Complete random search

```
W = np.random.randn(10, 3073) * 0.0001 # generate random parameters
loss = L(X_train, Y_train, W) # get the loss over the entire training set
if loss < bestloss: # keep track of the best solution
  bestloss = loss
  bestW = W
```

Local Random Search

- Each iteration pick a random direction to step in ***W***
- If the loss is lower than the previously seen best loss, use this as your new ***W***

Local Random Search

```
step_size = 0.0001
Wtry = W + np.random.randn(10, 3073) * step_size
loss = L(Xtr_cols, Ytr, Wtry)
if loss < bestloss:
  W = Wtry
  bestloss = loss
```

Gradient descent

- But we can do better!
- We know the **exact** direction of steepest descent
- It is the **negative** of the gradient

Numerical Gradient Descent

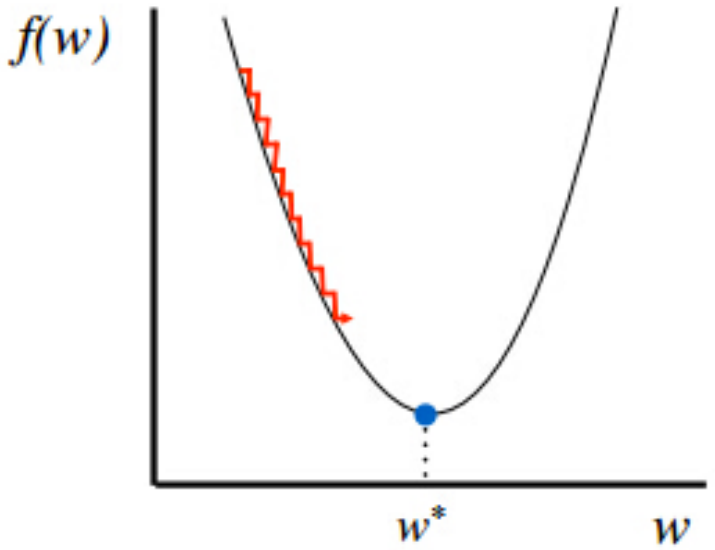
- Recall

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

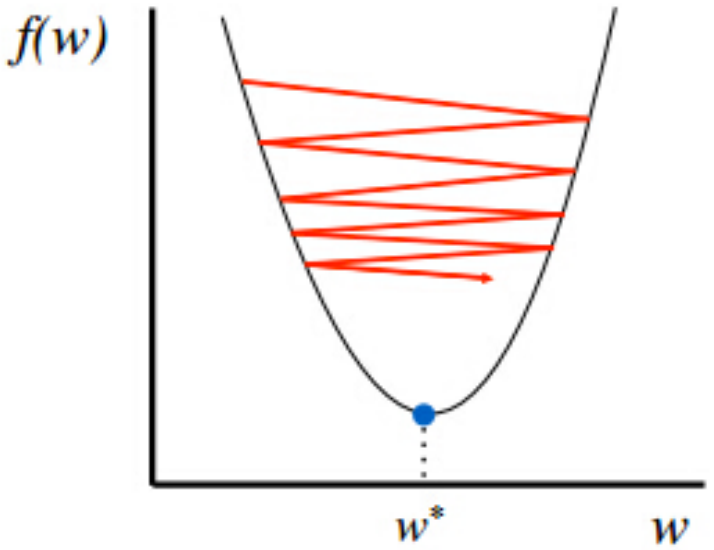
- Calculate the numerical gradient in each direction
- This results in a vector of partial derivatives
- Update the weights according to steepest descent

```
W_new = W - step_size * df # new position in the weight space
```

Step Size



Too small: converge very slowly



Too big: overshoot and even diverge

Analytical Gradient Descent

- Recall the probability and loss function derived earlier

$$p_k = \frac{e^{f_k}}{\sum_j e^{f_j}} \qquad L_i = -\log\left(p_{y_i}\right)$$

- Differentiating wrt $\boldsymbol{f_k}$

$$\frac{\partial L_i}{\partial f_k} = p_k - \mathbb{1}(y_i = k)$$

Analytical Gradient Descent

- Recall there's also a regularization term

$$L = \underbrace{\frac{1}{N}\sum_i L_i}_{\text{data loss}} + \underbrace{\frac{1}{2}\lambda \sum_k \sum_l W_{k,l}^2}_{\text{regularization loss}}$$

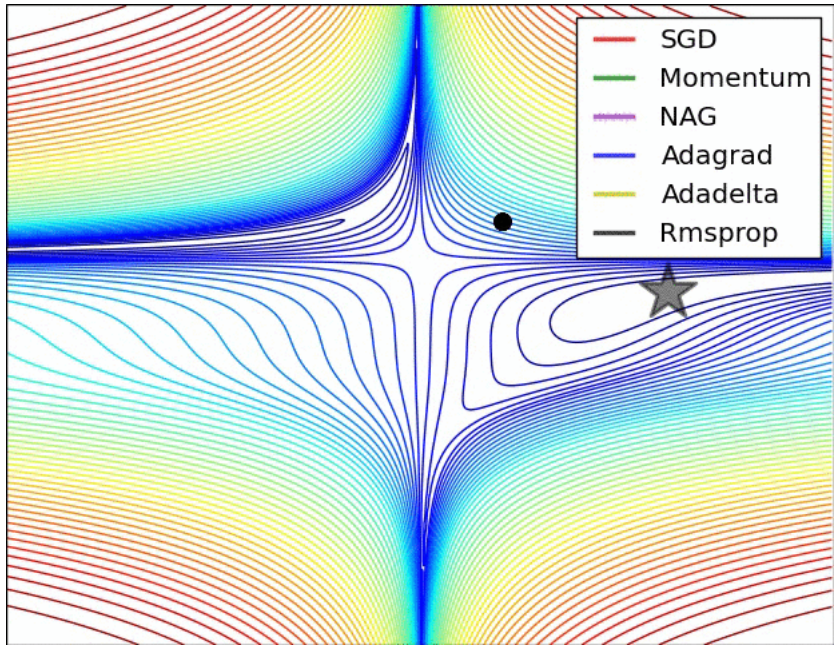
- Differentiating wrt ***w***

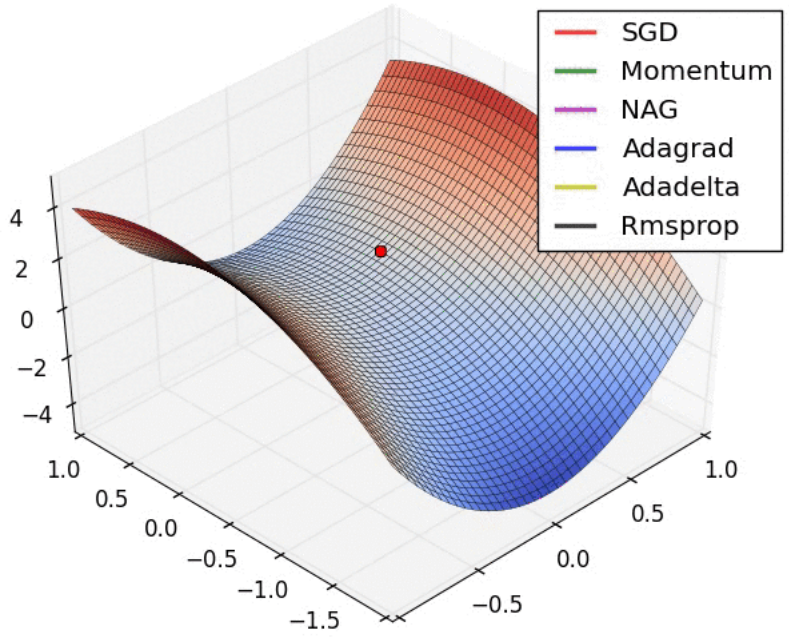
$$\frac{d}{dw}\left(\frac{1}{2}\lambda w^2\right) = \lambda w$$

Putting it all together

- Open iPython notebook
  - https://cs.stanford.edu/people/karpathy/cs231nfiles/minimal_net.html

Step Size Revisited





Other notes

- In practice, don’t perform the updates using all examples
- Do it in mini batches
- Works because samples are considered to be correlated
- ***SGD*** is extreme case of update per sample (online learning)
  - In practice SGD is still done in mini batches
- Batches are usually powers of 2: 32, 64, 128 etc.

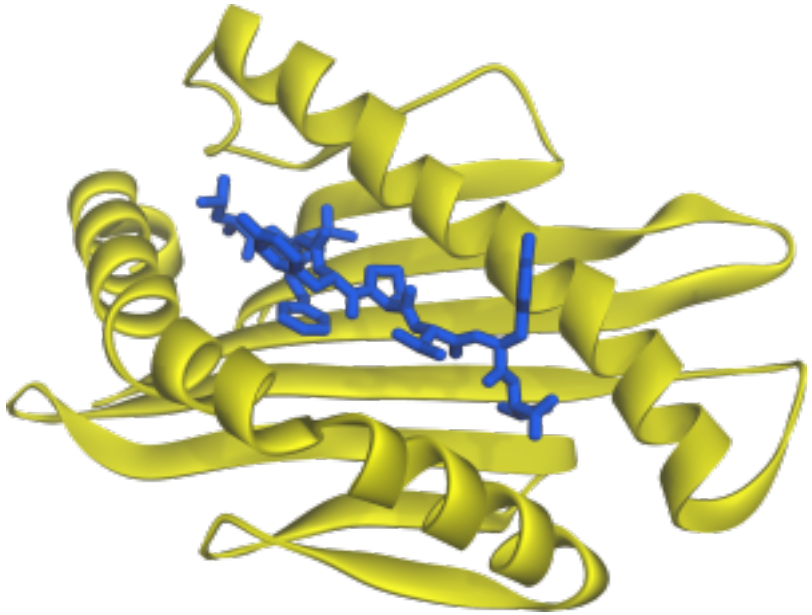
In Real Life (IRL)

Cancer Immunotherapy

- Utilize the patient's immune system to fight the tumour
- Identify antigens from the tumour (neoantigens) that bind the patient's MHC molecules as being foreign (as a result of mutations)
- Result in an activation of an immune response against the tumour

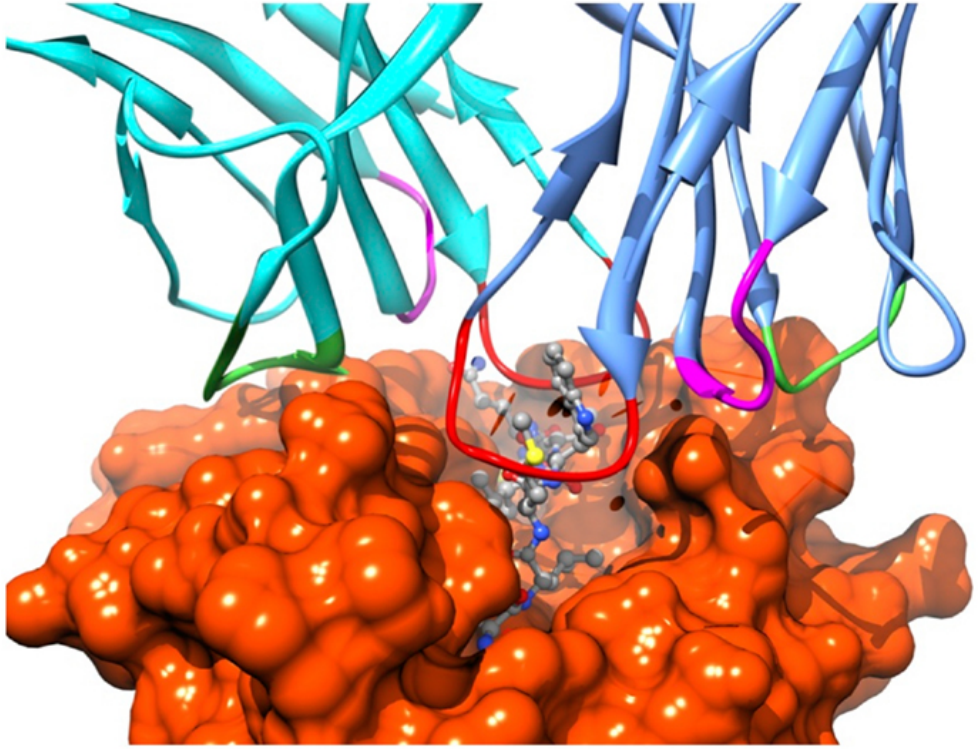
Peptide-MHC binding

Binding of a peptide to an MHC molecule is the first critical step in the formation of an immune response



BAKER LAB, UNIVERSITY OF NOTRE DAME

Peptide-MHC TCR complex

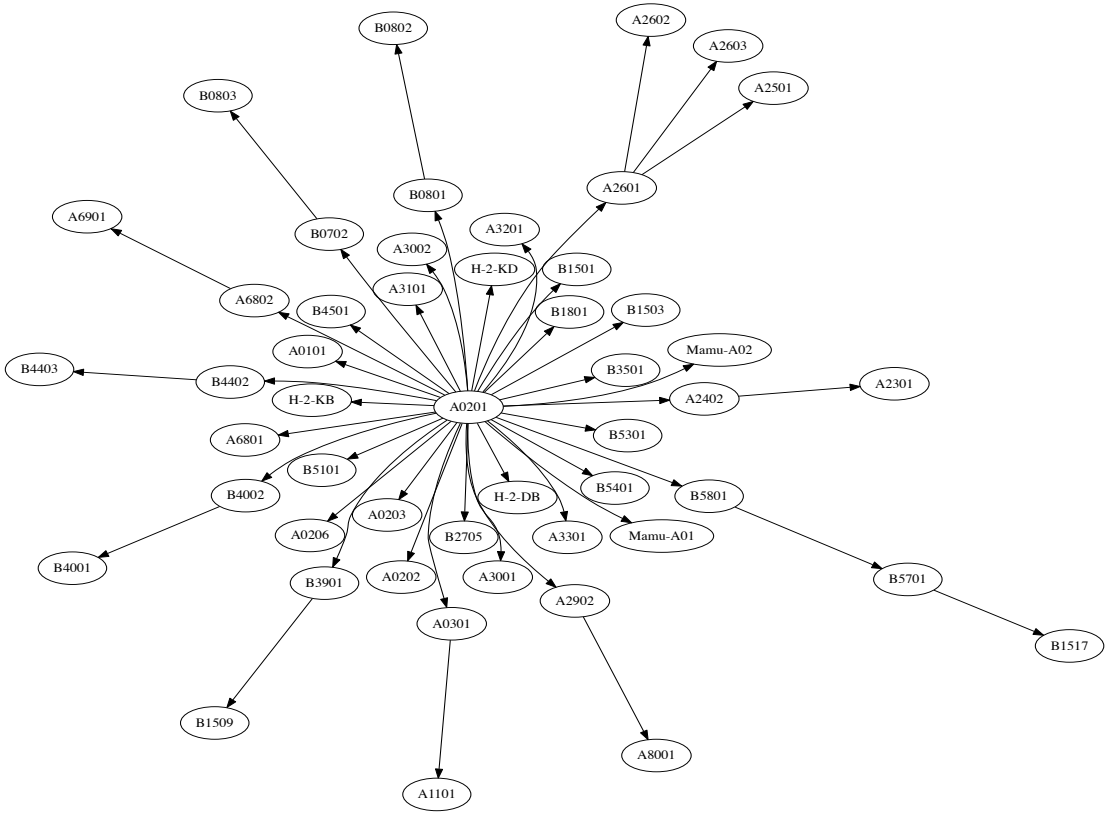


ZOETE ET AL, 2013

MHCnuggets

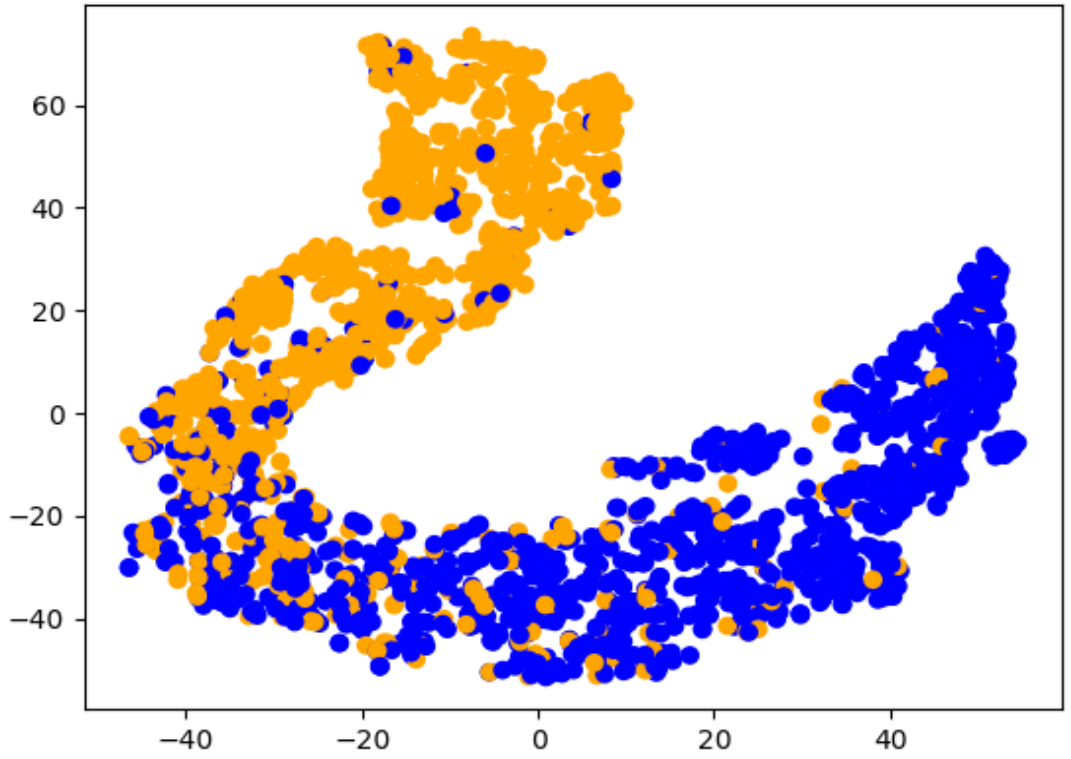
- One-hot encoded peptides
- Build a separate neural network for each MHC allele
- Each network is an LSTM layer of 64 units with an FC layer of 64 units stack on top
- Utilize a transfer learning protocol to generate better predictions for rare MHC alleles

Transfer learning



BHATTACHARYA ET AL, 2017

tSNE on feature space



Questions