BIML 2017 Feb. 16, 2017

MINING AND LEARNING BIO-BIG DATA

Sael Lee
Department of Computer Science,
SUNY Korea, Incheon 21985, Korea

OUTLINE

- Part 1:
  - Big Data Characteristics in Bioinformatics Data
  - Matrix Factorization Based Mining
  - Tensor Factorization Based Mining
- **Part 2:**
  - **Deep Neural Network and Bio-Big(?) Data**
    - **NN Basics & Types of DNN**
    - **Bio-Big Data Applications**
  - **Convolution Neural Network**
    - **Theory**
    - **Practice**
    - **TensorFlow**

PART 2.1:

DEEP NEURAL NETWORK AND BIO-BIG DATA

- NN BASICS & TYPES OF DNN
- BIO-BIG DATA APPLICATIONS

ARTIFICIAL NEURAL NETWORKS

- Neural networks
- Perceptrons
- Multilayer perceptrons
- Applications of neural networks

Part 2.1.1 Slides are mostly made from AIMA resources and
Andrew W. Moore's tutorials: http://www.cs.cmu.edu/~awm/tutorials

EXAMPLE APPLICATIONS

ANN is robust to error in the training data and has been successfully applied to various real problems

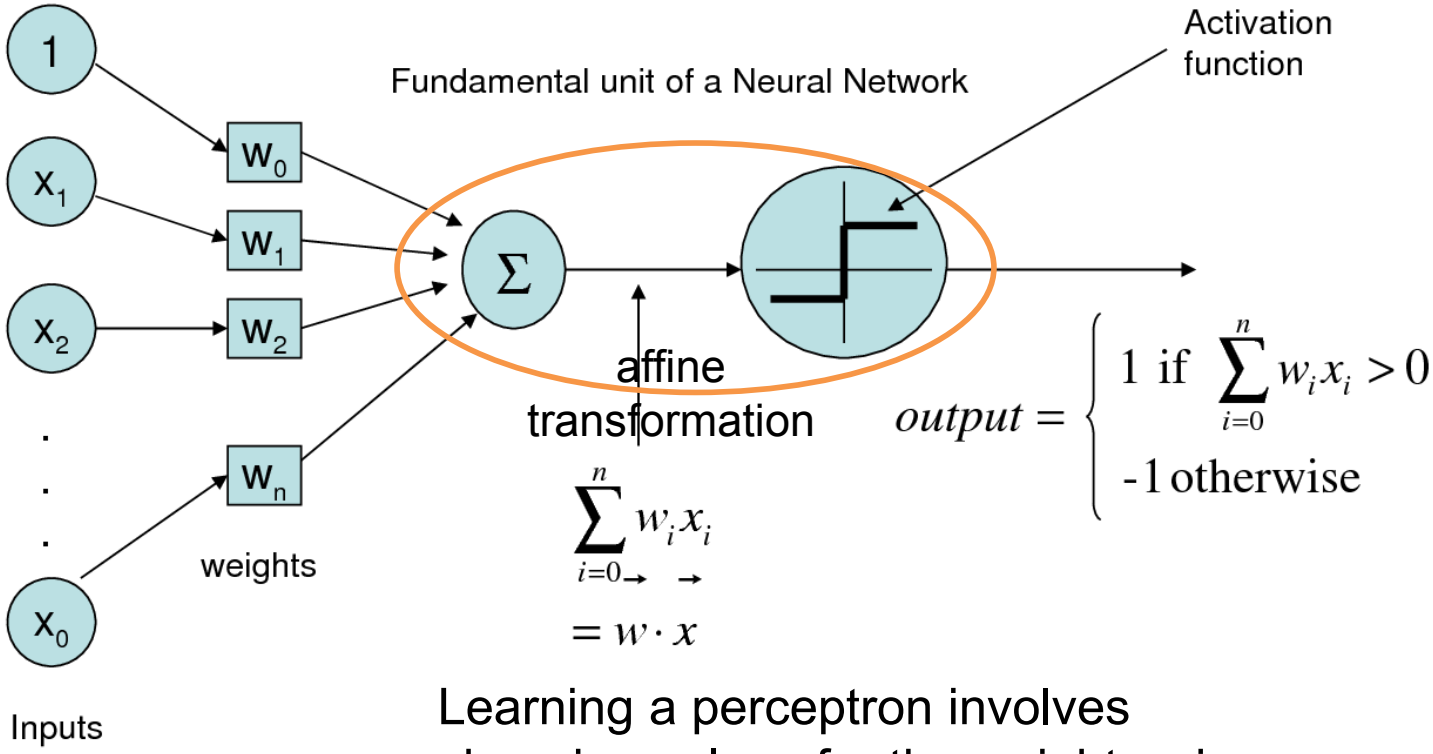
- Speech/voice recognition
- Face recognition
- Handwriting recognitions
- It can also be used where symbolic representations are used as cases for Decision tree learning

CHARACTERISTICS OF ANN

- Instances are represented by many attribute-value pair (supervised)
- The target function output may be discrete, real, or vector
- Training data may contain error
- Long training times are acceptable
- Fast evaluation of the learning target function may be required
- The ability of humans to understand the learned target function is not important.

PRIMITIVE UNITS THAT MAKE UP ANN

Perceptron



Learning a perceptron involves choosing values for the weights wi

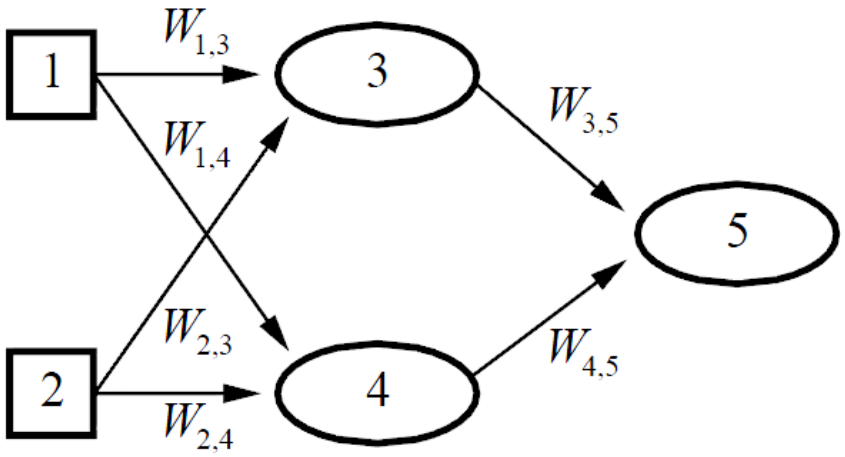
- Other Units:
  - Linear units
  - Sigmoid units
  - Rectified linear units

ANN STRUCTURE: CONNECTING UNITS

- **Feed-forward networks**: connections only in one direction (directed acyclic graph)
  - Feed-forward network implement functions, have no internal state
  - Examples:
    - single-layer perceptrons (output is 0 or 1)
    - multi-layer perceptrons
    - Convolution neural network

- **Recurrent networks:**
  - Have directed cycles (feedback loops) with delays ⇒ have internal state (like flip-flops), can oscillate etc.
  - Interesting models of the brain but more difficult to understand.

FEED-FORWARD EXAMPLE



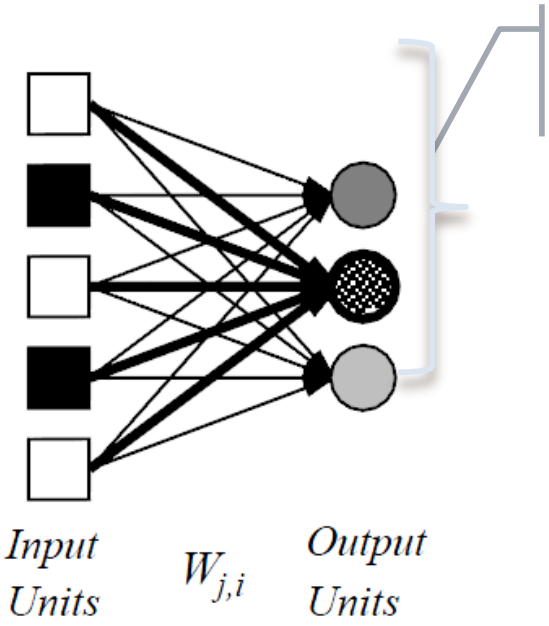
Feed-forward network = a parameterized family of nonlinear functions:

$$
\begin{aligned}
a_5 &= g(W_{3,5} \cdot a_3 + W_{4,5} \cdot a_4) \\
&= g(W_{3,5} \cdot g(W_{1,3} \cdot a_1 + W_{2,3} \cdot a_2) + W_{4,5} \cdot g(W_{1,4} \cdot a_1 + W_{2,4} \cdot a_2))
\end{aligned}
$$

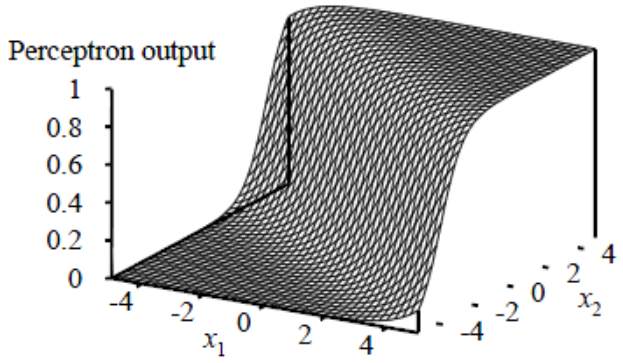
Adjusting weights changes the function: do learning this way!

SINGLE LAYER FEED-FORWARD NEURAL NETWORKS: PERCEPTRON NETWORK

Every unit connects directly form the network's inputs to it's output



3 separate learning tasks



Output units all operate separately — no shared weights
Adjusting weights moves the location, orientation, and steepness of cliff.

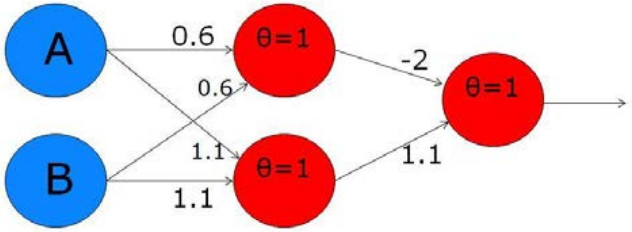
EXPRESSIVENESS OF SINGLE LAYER PERCEPTRONS

- Consider a perceptron with g = step function (Rosenblatt, 1957, 1960)
- Can represent AND, OR, NOT, majority, etc., but not XOR
- Represents a **linear separator** in input space:

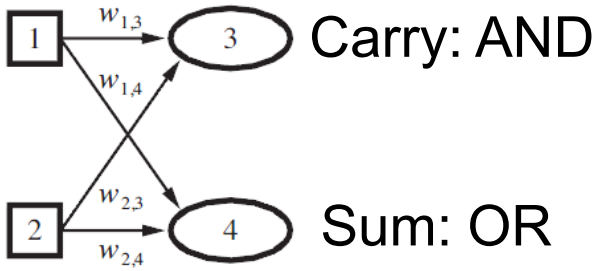
EX> Two bit adder

Two separate component

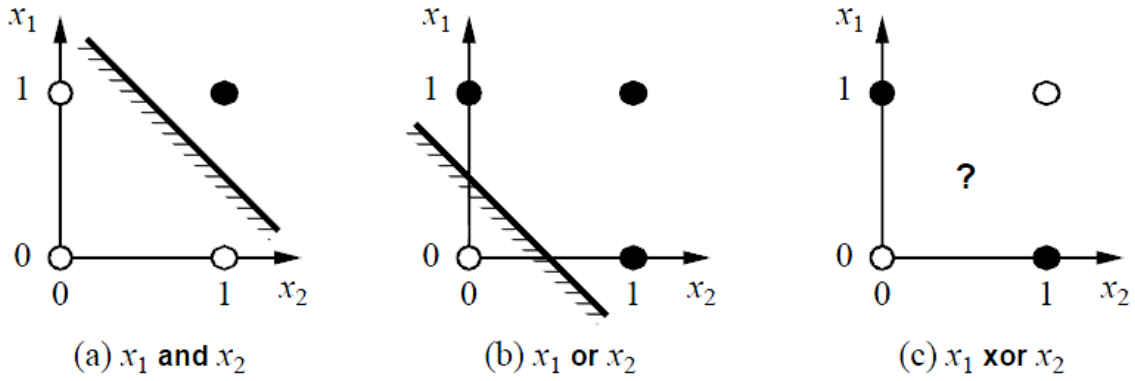
1. Carry 2. sum



X1	X2	Y3 (carry)	Y4 (sum)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1



$\sum_j W_j x_j > 0$ or $\mathbf{W} \cdot \mathbf{x} > 0$



PERCEPTRON LEARNING

Learn by adjusting weights to reduce error on training set
The squared error for an example with input x and true output y is

$$E = \frac{1}{2}Err^2 \equiv \frac{1}{2}(y - h_{\mathbf{W}}(\mathbf{x}))^2$$

Perform optimization search by **gradient descent**
(just like logistic regression)

$$\frac{\partial E}{\partial W_j} = Err \times \frac{\partial Err}{\partial W_j} = Err \times \frac{\partial}{\partial W_j}(y - g(\Sigma_{j=0}^{n} W_j x_j))$$
$$= -Err \times g'(in) \times x_j$$

Simple weight update rule:

$$W_j \leftarrow W_j + \alpha \times Err \times g'(in) \times x_j$$

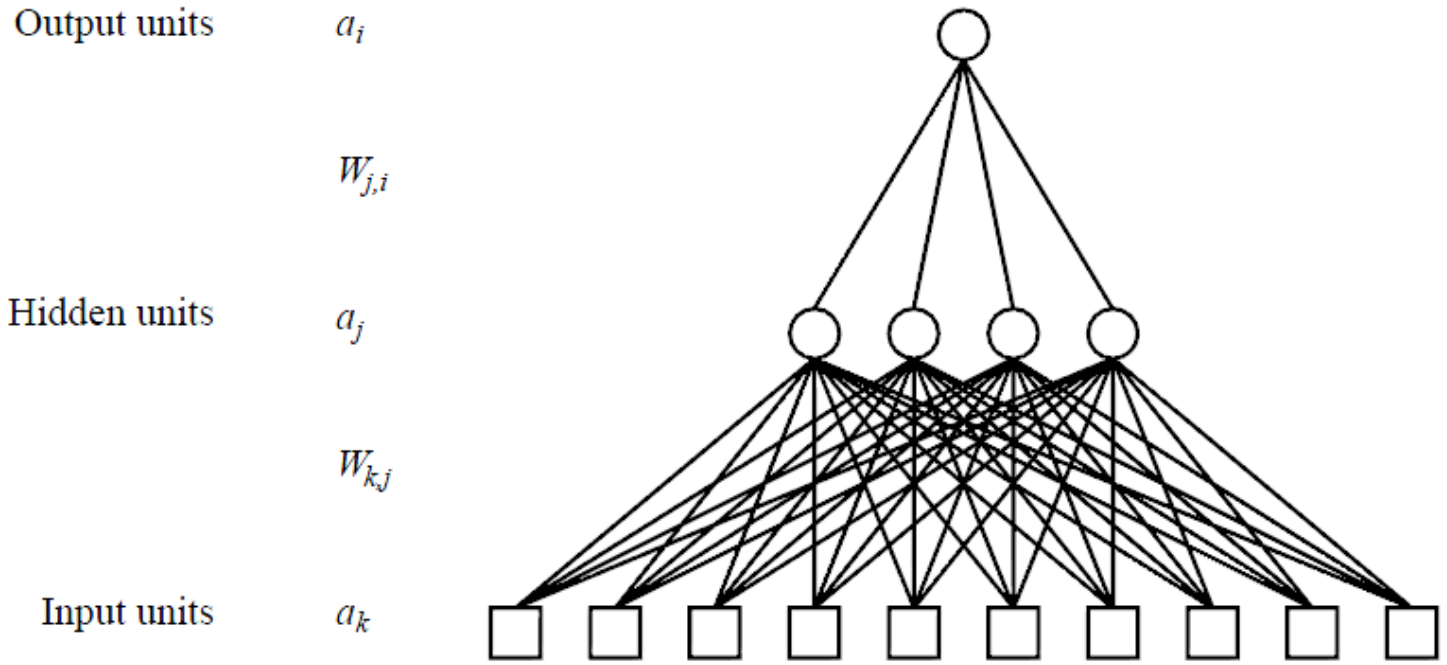
E.g., +ve error ⇒ increase network output
⇒ increase weights on +ve inputs, decrease on -ve inputs

* Chain rule:

$$\frac{\partial g(f(x))}{\partial x} = \frac{g'(f(x))\partial f(x)}{\partial x}$$

MULTILAYER PERCEPTRONS

Layers are usually fully connected;
numbers of hidden units typically chosen by hand



EXPRESSIVENESS OF MLPS

All continuous functions w/2 layers, all functions w/3 layers

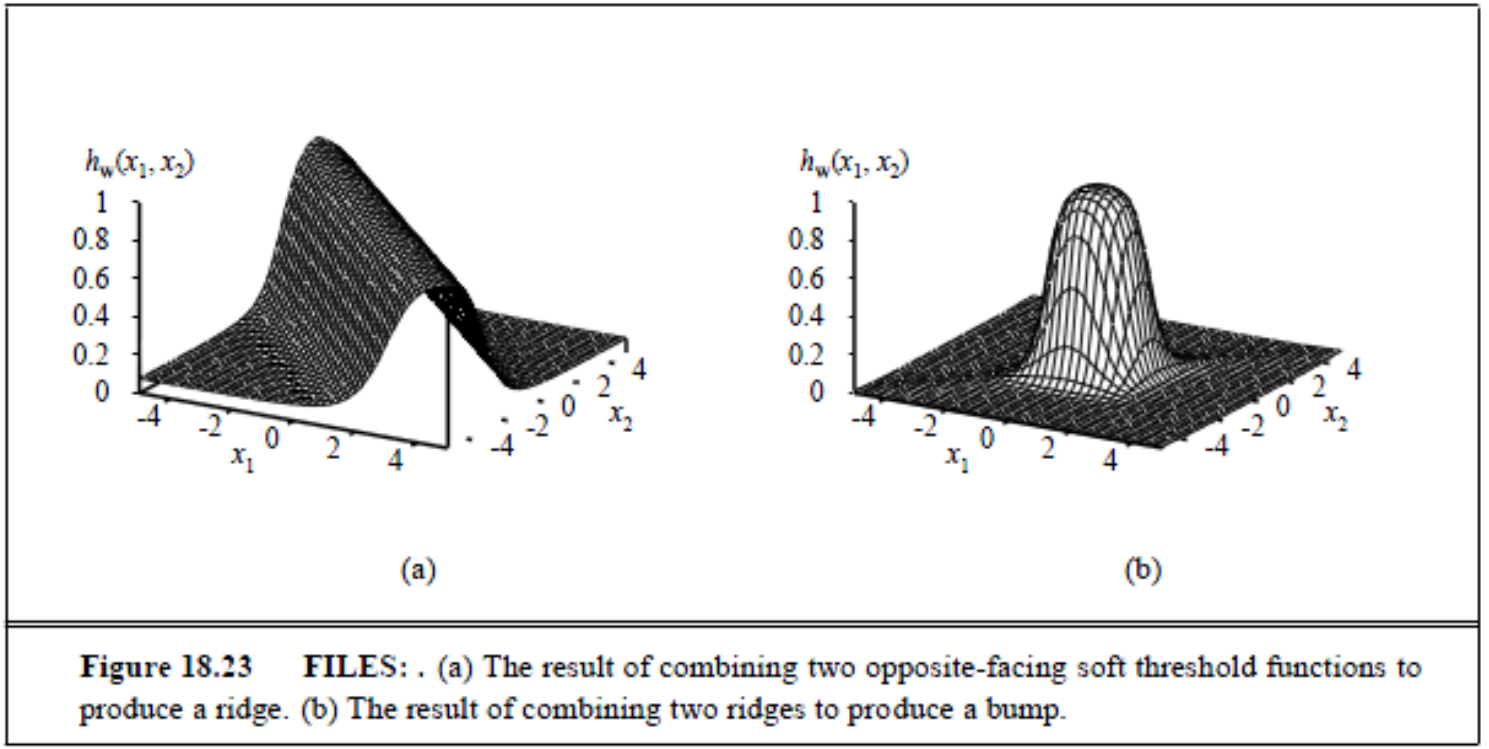
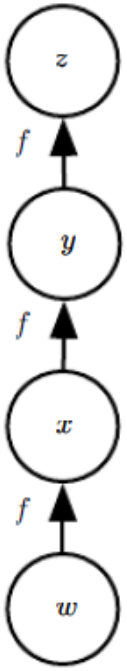


Figure 18.23 **FILES:** . (a) The result of combining two opposite-facing soft threshold functions to produce a ridge. (b) The result of combining two ridges to produce a bump.

Combine two opposite-facing threshold functions to make a ridge
Combine two perpendicular ridges to make a bump
Add bumps of various sizes and locations to fit any surface

BACK-PROPAGATION

- **Back propagation** allows the information from the cost to then flow backwards through the network, in order to **compute the gradient**.
  - Refers only to the method for computing the gradient,
  - Different from other algorithms, e.g., stochastic gradient descent, used to perform learning using this gradient.
- Chain Rule
  - Basic rule of backprop



$$\frac{\partial z}{\partial w} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\frac{\partial x}{\partial w}$$

$$= f'(y)f'(x)f'(w)$$

$$= f'(f(f(w)))f'(f(w))f'(w)$$

Have sufficient memory

memory is limited.

BACK-PROPAGATION LEARNING FOR MLP

1. Output layer: **weight update rules** are same as for single-layer perceptron,

$$W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \Delta_i$$

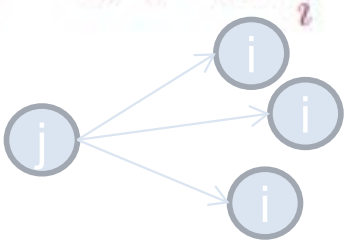
where $\Delta_i = Err_i \times g'(in_i)$

$$W_j \leftarrow W_j + \alpha \times Err \times g'(in) \times x_j$$

2. Hidden layer: **Error back-propagation rule**

back-propagate the error from the output layer:

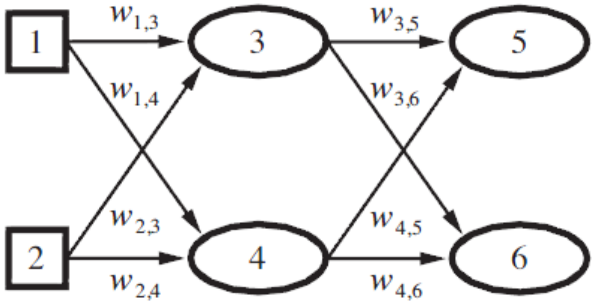
$$\Delta_j = g'(in_j) \sum_i W_{j,i} \Delta_i$$



Hidden layer is responsible for Δ_i portion of error according to strength of the connection.

3. Update rule for weights in hidden layer:

$$W_{k,j} \leftarrow W_{k,j} + \alpha \times a_k \times \Delta_j$$



THE BACK-PROPAGATION ALGORITHM

function BACK-PROP-LEARNING($examples$, $network$) **returns** a neural network
  inputs: $examples$, a set of examples, each with input vector **x** and output vector **y**
    $network$, a multilayer network with L layers, weights $w_{i,j}$, activation function g
  local variables: Δ, a vector of errors, indexed by network node

  repeat
    for each weight $w_{i,j}$ in $network$ **do**
      $w_{i,j} \leftarrow$ a small random number
    for each example (**x**, **y**) **in** $examples$ **do**
      /* *Propagate the inputs forward to compute the outputs* */
      for each node i in the input layer **do**
        $a_i \leftarrow x_i$
      for $\ell = 2$ **to** L **do**
        for each node j in layer ℓ **do**
          $in_j \leftarrow \sum_i w_{i,j}\, a_i$
          $a_j \leftarrow g(in_j)$
      /* *Propagate deltas backward from output layer to input layer* */
      for each node j in the output layer **do**
        $\Delta[j] \leftarrow g'(in_j) \times (y_j - a_j)$
      for $\ell = L - 1$ **to** 1 **do**
        for each node i in layer ℓ **do**
          $\Delta[i] \leftarrow g'(in_i) \sum_j w_{i,j}\, \Delta[j]$
      /* *Update every weight in network using deltas* */
      for each weight $w_{i,j}$ in $network$ **do**
        $w_{i,j} \leftarrow w_{i,j} + \alpha \times a_i \times \Delta[j]$
  until some stopping criterion is satisfied
  return $network$

Compute the Δ values for the output units using the observed error

Propagate the Δ values back to the previous layer.

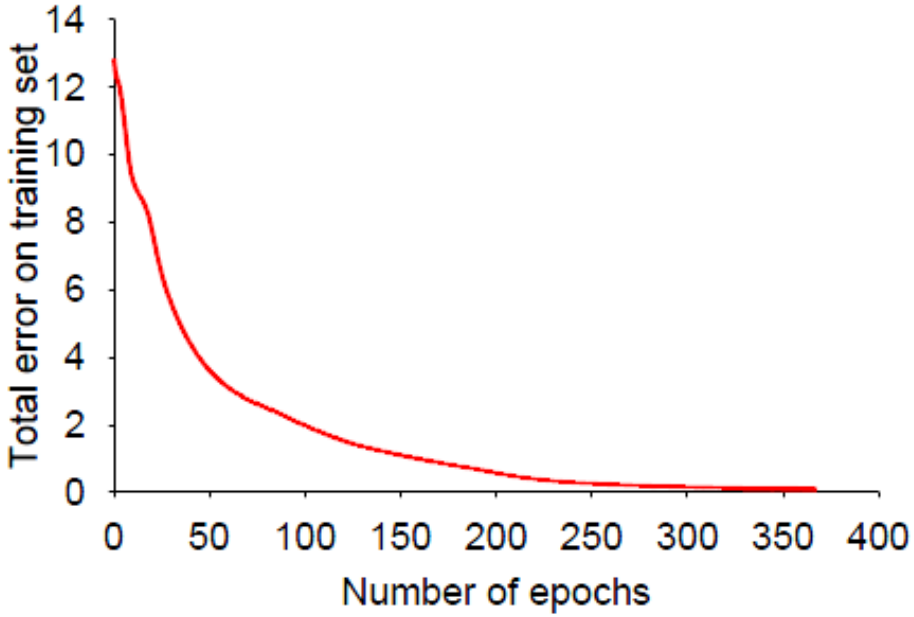
$$\Delta_j = g'(in_j) \sum W_{j,i} \Delta_i$$

Update the weights between the two layers.

Propagate the Δ values back to the previous layer.

BACK-PROPAGATION LEARNING CONT.

At each epoch, sum gradient updates for all examples and
Apply Training curve for 100 restaurant examples:
finds exact fit



Typical problems: slow convergence, local minima

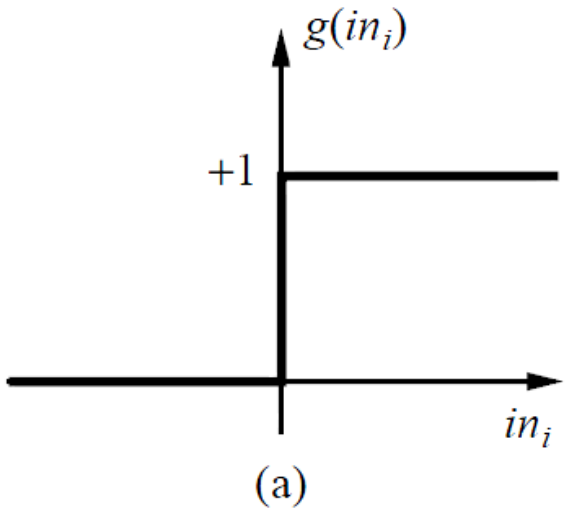
TYPES OF OUTPUT UNITS IN ANN

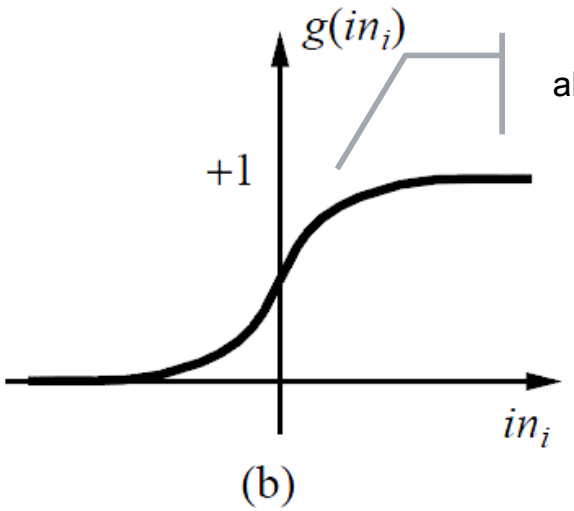
- Choice of cost function is tightly coupled with the choice of output units.
- The role of the output layer is to provide some additional transformation from the features to complete the task.
- Types of output units
  - Linear Units for Gaussian output (regression)
    - $\hat{y} = W^T h + b$
  - Sigmoid Units for Bernoulli output (classification)
  - Softmax Units for Multinomial output (multimodal classification)
  - Gaussian Mixture Units (multimodal regression)
  - Others

ACTIVATION FUNCTIONS *G*

$$a_i \leftarrow g(in_i) = g\left(\sum_j W_{j,i} a_j\right)$$

Activation function enables the model to be **nonlinear**





Sigmoid function allows the model to be **differentiable**

Hard threshold:
perceptron

Logistic function:
Sigmoid perceptron

(a) is a step function or threshold function
(b) is a sigmoid function $1/(1 + \exp(-W^T A))$
Changing the bias weight $W_{0,i}$ moves the threshold location

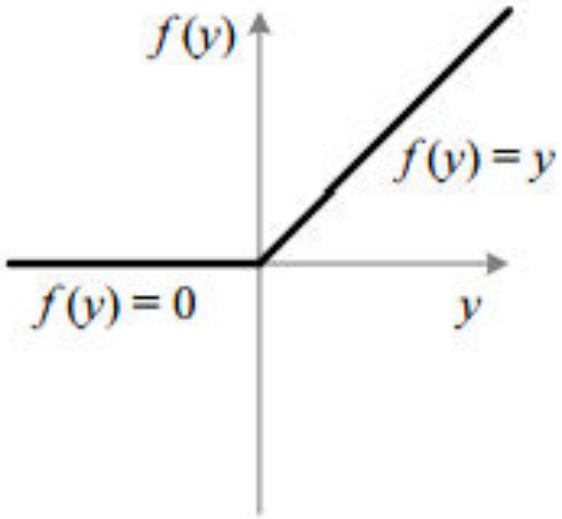
TYPES OF HIDDEN UNITS IN ANN

- Variants of Linear and sigmoid units
- Variants of hyperbolic tangent $h = tanh(W^T x + b)$
- Variants of Rectified linear units (ReLU) (the default choice of hidden unit for modern ANN.)
  - Derivative through a ReLU remain large whenever the unit is active
  - Gradients are more consistent
  - Typically used on top of a affine transformation : $h = \max\{0, (W^T x + b)\}$
  - they cannot learn via gradient based methods on examples for which their activation is zero.

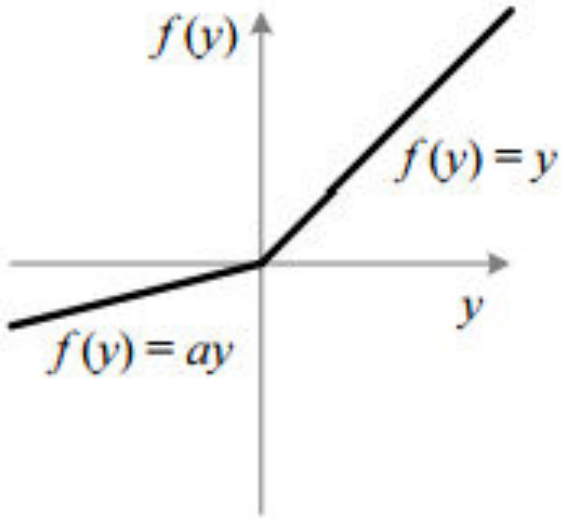
SELECTING HIDDEN UNITS

- The design of hidden units does not yet have many definitive guiding theoretical principles.
- It is difficult to determine which units will work prior to experiment
- Hidden units that are not differentiable at small number of points can still be used
  - Because we do not expect training to actually reach a point where the gradient is 0 , it is acceptable for the minima of the cost function to correspond to points with undefined gradient

RECTIFIED LINEAR UNITS



Rectified Linear Unit (ReLU)
f(y) = max{0,y}



Parametric Rectified Linear Unit (PReLU)

ANN ARCHITECTURE

- Structure of ANN:
  - How many layers
  - How many units in each layer
  - How these units should be connected to each other.

- NOTE: Deeper networks often are able to use far fewer units per layer and far fewer parameters and often generalize to the test set, but are also often harder to optimize.

DEEPER MODELS TEND TO PERFORM BETTER

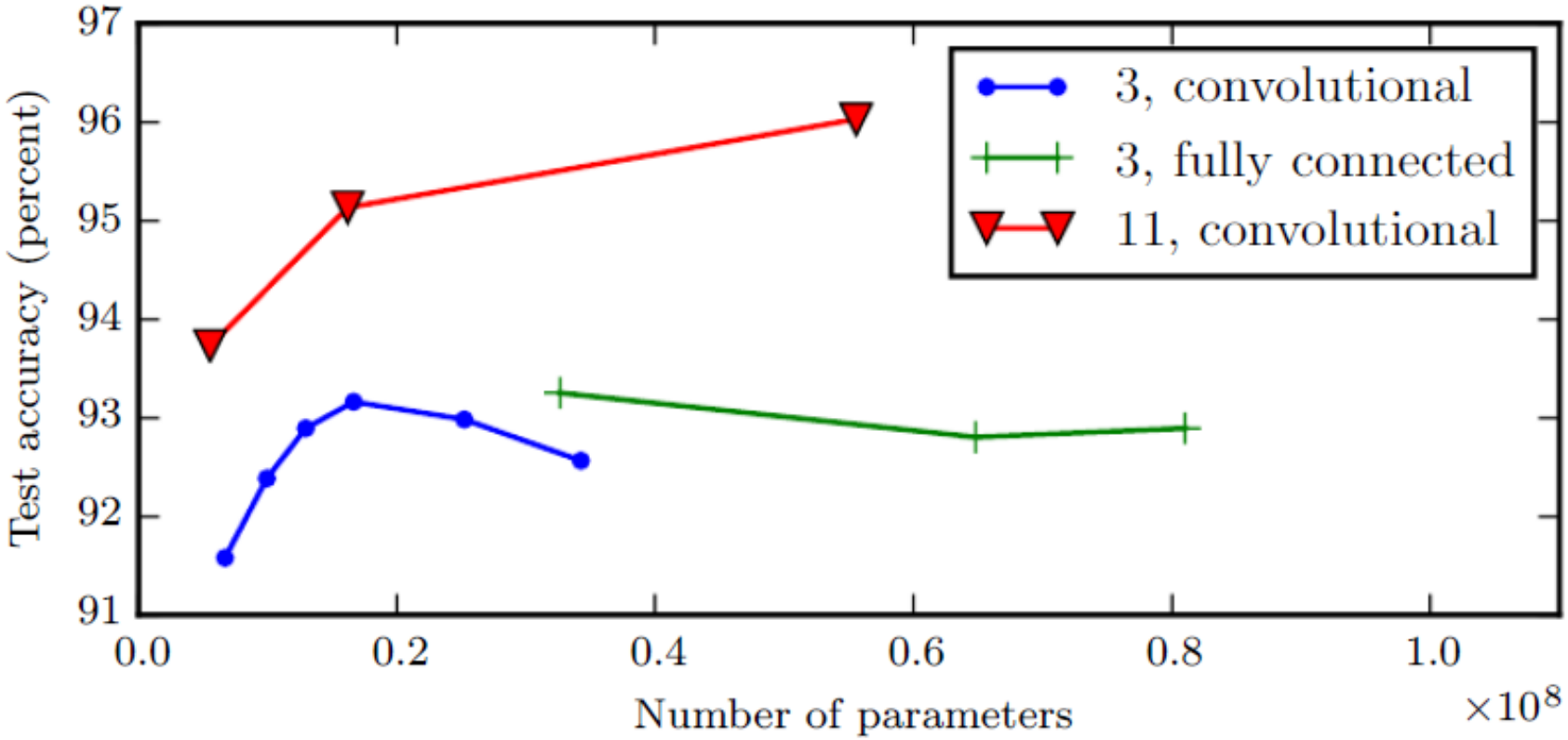


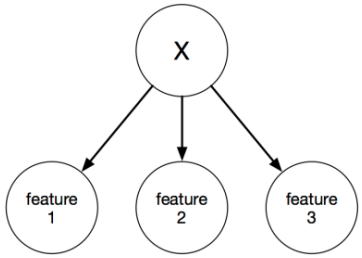
Figure from: Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. (2014). Multi-digit number recognition from Street View imagery using deep convolutional neural networks. In International Conference on Learning Representations .

LEARNING THE STRUCTURE

- Cross-validation
  - If we stay with fully connected networks, structural parameters to choose from are:
    - Number of hidden layers and their sizes.
- Optimal brain damage
  - Start with fully connected network and start removing links and units iteratively.
- Tiling
  - Starting from single unit and start adding units to take care of the examples that current units got wrong.

DEEP NEURAL NETWORK (DNN)

Deep Neural Network is a **graphical model** that can be placed somewhere between Naïve Bayes and Generalized Conditional Random Field.



Naïve Bayes

DNN

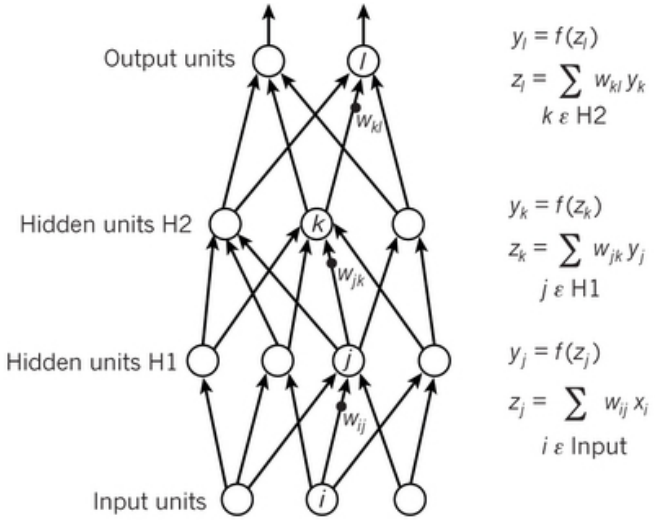
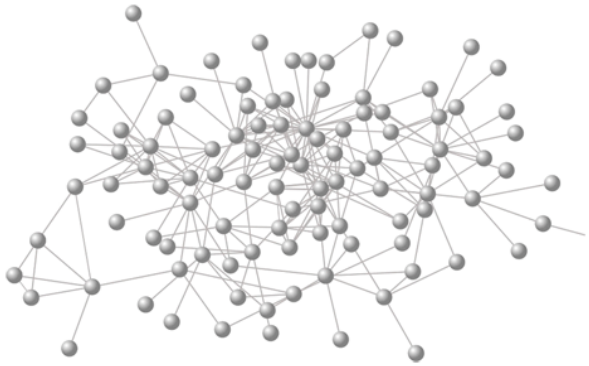


Fig from LeCun et al. 2015 Nature



Markov Random Field & Conditional Random Field

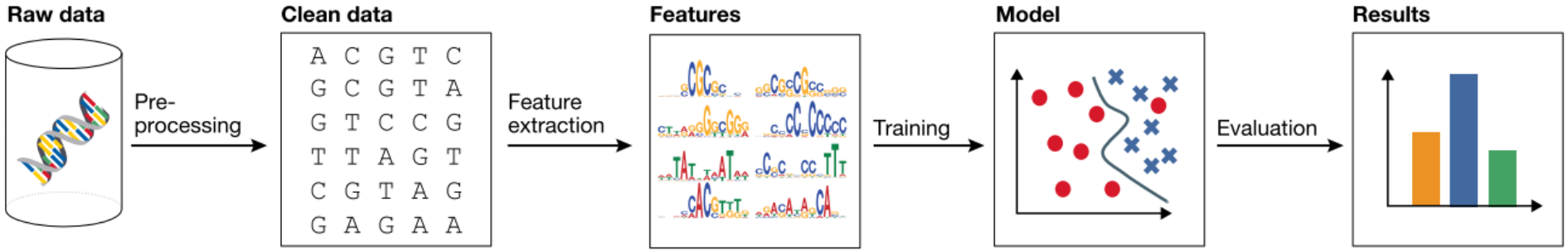
Easy to design; exact optimization can be done

Difficult to design; most rely on approx. algos

Increasing in Complexity

BENEFITS OF DNN LEARNING

Classical Machine Learning Pipeline in Comp Bio



Deep Learning in Comp Bio.

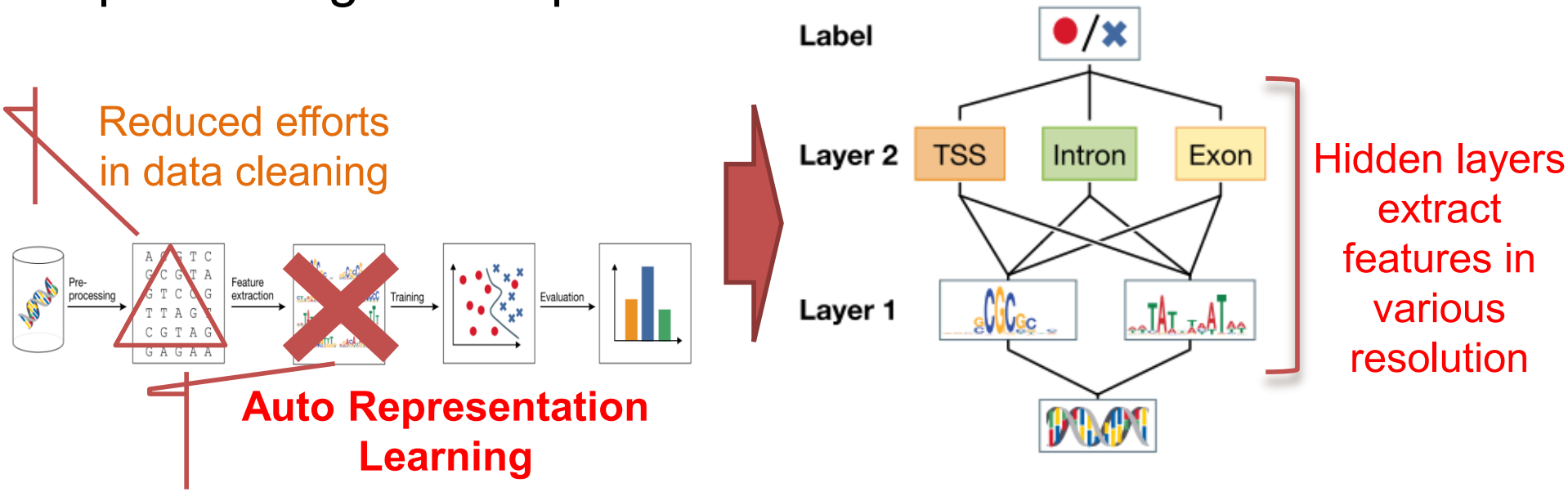


Fig 1A,D from Angermueller et al. (2016) *Molecular Systems Biology*, (12), 878.

VARIOUS APPLICATIONS

- Regulatory Genomics
  - Alternative Splicing (Leung et al 2014; Xiong et al, 2015)
  - Accessible Genome Analysis (Zhous & Troyanskaya, 2015; Kelley et al, 2016)
  - Protein-Nucleic Acid Binding Prediction (Alipananhi et al, 2015)
  - Variant Analysis
- Protein Structure Prediction
  - Secondary structure Prediction
  - Order/Disorder Region Prediction
  - Residue-Residue Contact Prediction
- Applications on High throughput Data
  - QSAR Prediction
  - Circadian Rhythms
- Other Topics Not Covered
  - Cellular Image Analysis
  - Medical Time Series Data

EARLY WORKS OF DNN IN ALTERNATIVE SPLICING

Deep Feedforward NN

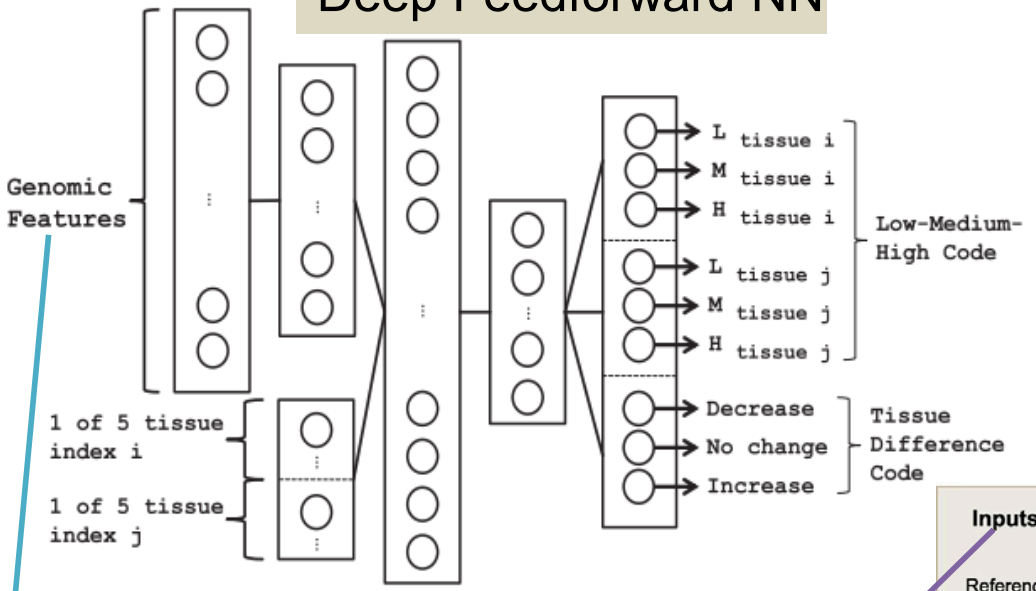


Fig 1 of Leung et al. (2014) Bioinformatics 30(12) 121-129

"Deep learning of the tissue-regulated splicing code"

1393 features extracted from each exon of 5 different tissue types

1000 predetermined features from candidate exon and adjacent introns

Early works still utilize **selected (large size) features**

Fully connected Feedforward NN (Bayesian Deep Learning)

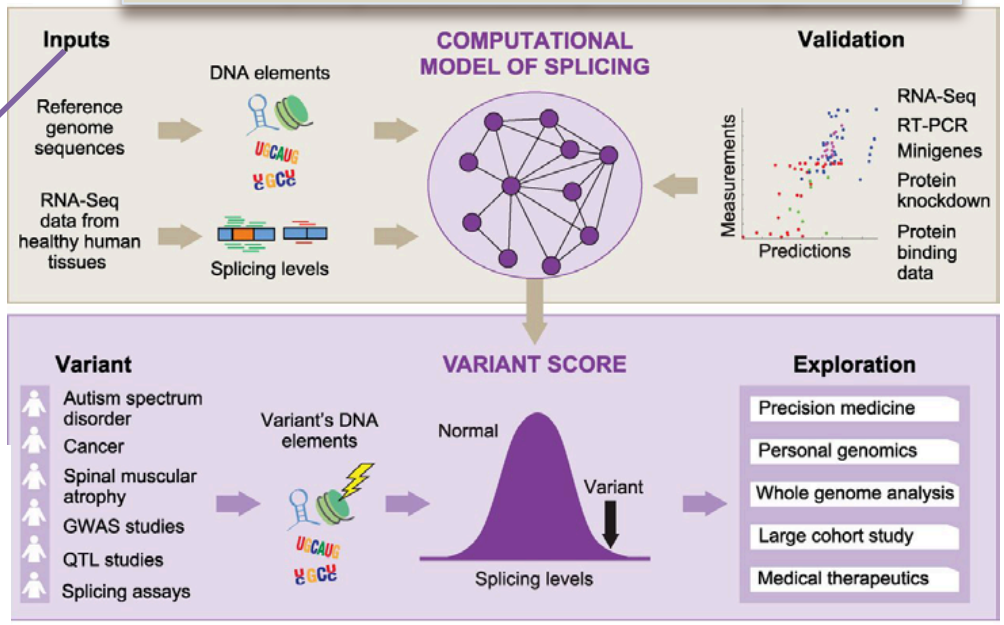


Fig 1 of Xiong et al. (2015) Science 347(6218):1254806

Feature listing

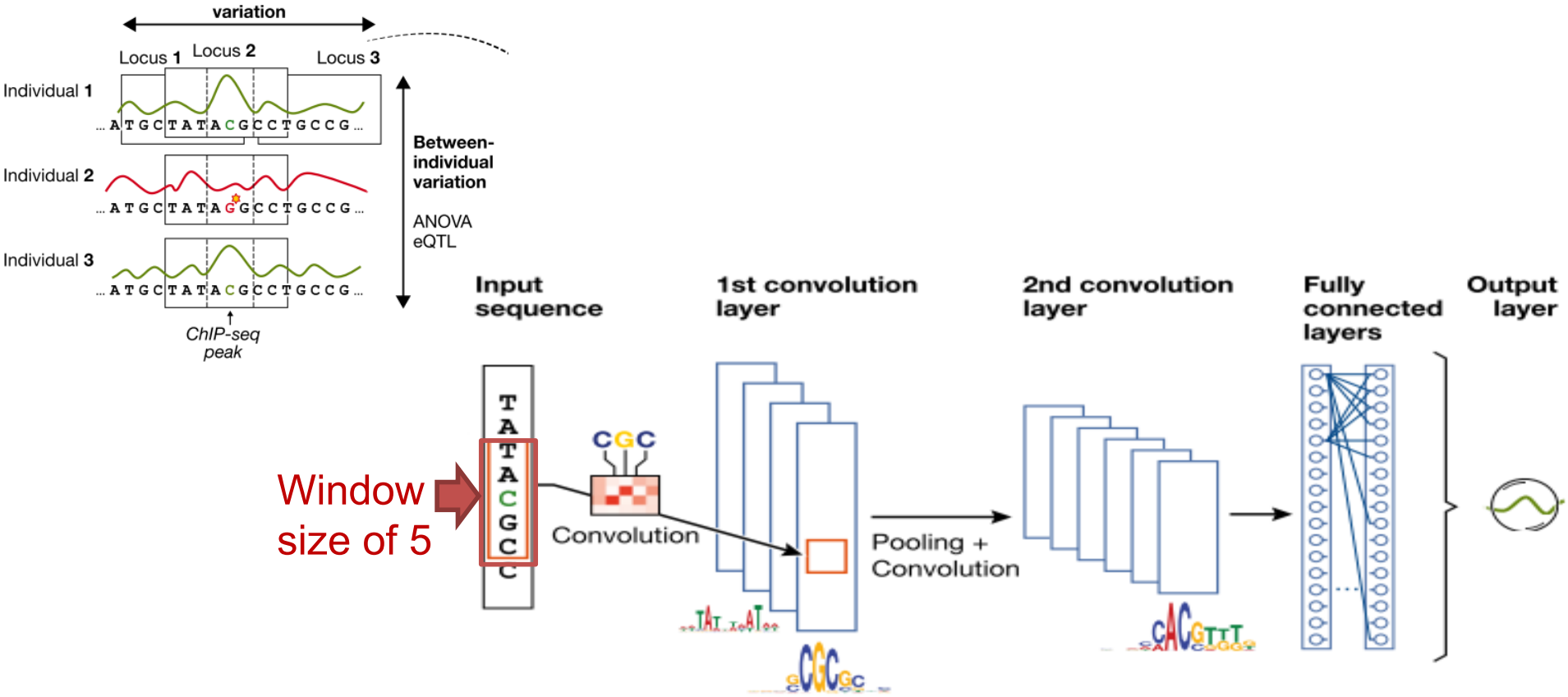
Leung et al. (2014)
Bioinformatics 30(12)
121-129

Group #	Name	Description	Type	# of Features
01	short-seq-1mer	Frequency of nucleotide patterns of different lengths (1 to 3).	real (0-1)	28
02	short-seq-2mer			112
03	short-seq-3mer			320
04	translatable-C1	Describes whether exons can be translated without a stop codon in one of three possible reading frames. For example, C1A means the exons of interest are C1 + A.	binary	1
05	translatable-C1A			1
06	translatable-C1AC2			1
07	translatable-C1C2			1
08	mean-con-score-AI2	Mean conservation score.	real (0-1)	1
09	mean-con-score-I1A			1
10	mean-con-score-I2C2			1
11	mean-con-score-C1I1			1
12	log-length	Log base 10 lengths of exons.	real	5
13	log-length-ratio	Log base 10 length ratios of exons.	real	3
14	acceptor-site-strength	Strength of acceptor and donor sites.	real	2
15	donor-site-strength			2
16	frameshift-exonA	Whether exon A introduces frame shift.	binary	1
17	rna-sec-struct	RNA secondary structures.	real (0-1)	32
18	5mer-motif-down	Counts of motif clusters of different lengths (5 to 7) weighted by conservation upstream and downstream from alternative exon.	real	54
19	6mer-motif-down			76
20	7mer-motif-down			28
21	5mer-motif-up			49
22	6mer-motif-up			78
23	7mer-motif-up			29
24	ese-ess-A	Counts of exonic splicing enhancers and silencers.	real	4
25	ese-ess-C1			4
26	ese-ess-C2			4
27	pssm-SC35	PSSM scores of SC35 splicing regulator protein.	real	5
28	pssm-ASF-SF2	PSSM scores of ASF/SF2 splicing regulator protein.		5
29	pssm-SRp40	PSSM scores of SRp40 splicing regulator protein.		10
30	nucleosome-position	Nucleosome positioning.	real	4
31	PTB	Phosphotyrosine-binding domain.	real	50
32	Nova-counts	Counts of Nova motif.	integer	27
33	Nova-cluster	Nova cluster score.	real	8
34	T-rich	Counts of motif with and without weighting by conservation.	real	24
35	G-rich			8
36	UG-rich			16
37	GU-rich			32
38	Fox	Counts of motif with and without weighting by conservation.	real	24
39	Quak			8
40	SC35			22
41	SRm160			11
42	SRrp20/30/38/40/55/75			77
43	CELF-like			2
44	CUGBP			16
45	MBNL			24
46	TRA2-alpha			22
47	TRA2-beta			22
48	hnRNP-A			44
49	hnRNP-H			22
50	hnRNP-G			22
51	9G8			22
52	ASF/SF2			11
53	Sugnet			2
54	alt-AG-pos	Position of the alternative AG and GT position.	integer	2
55	Alu-AI2	Counts of ALU repeats.	integer	12

C1 and *C2* denote the flanking constitutive exons ; *A* denotes the alternative exon ; *I1* denotes the intron between *C1* and *A* ; *I2* denotes the intron between *A* and *C2*

DNA/RNA SEQUENCE ANALYSIS WITH DEEP CNN

Convolution step in Deep CNN resembles traditional sequence "**windowing**" scheme

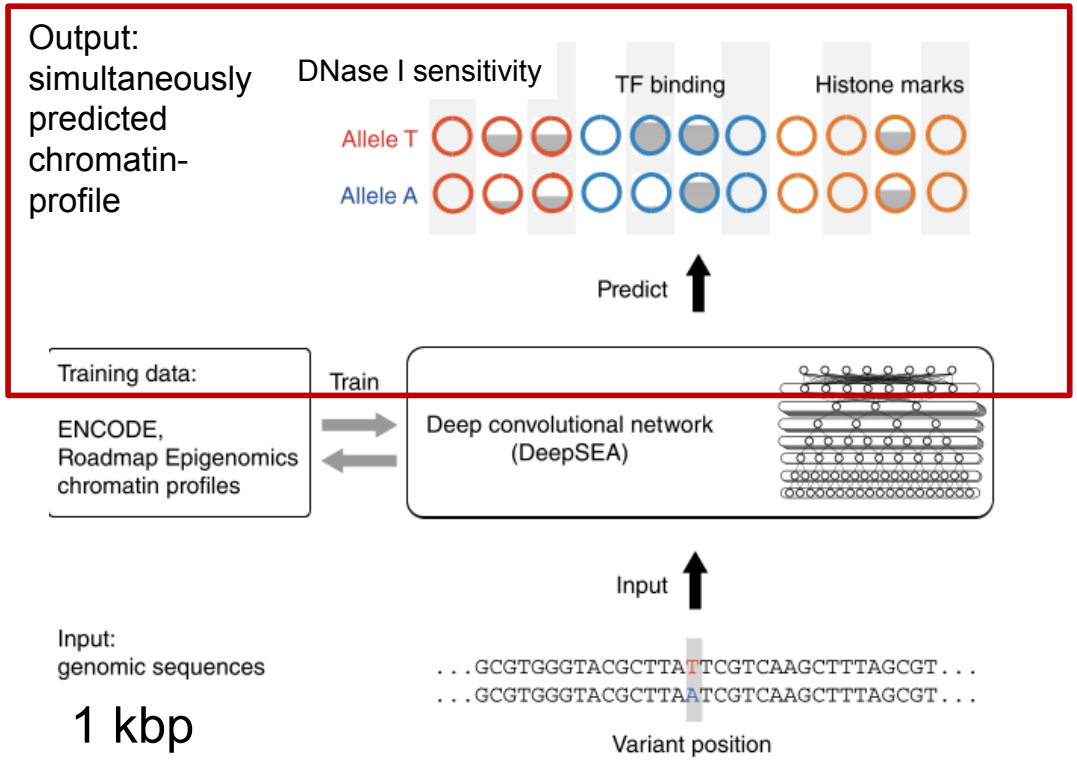


Angermueller et al. (2016) *Molecular Systems Biology*, (12), 878.

DEEPSEA: CNN-BASED NONCODING VARIANT EFFECT PREDICTION

GOAL: Identifying functional effects of noncoding variants

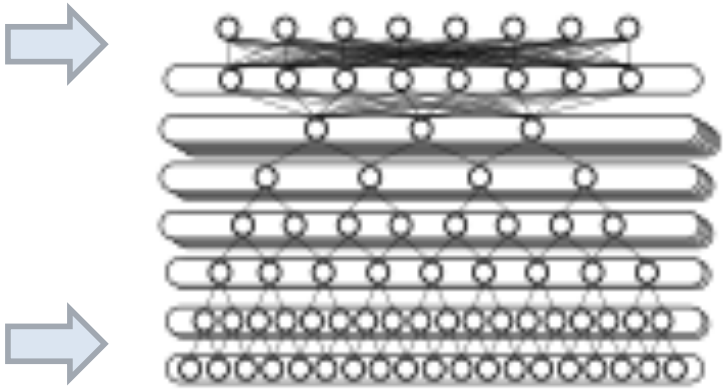
DeepSEA CNN structure



Innovative points:

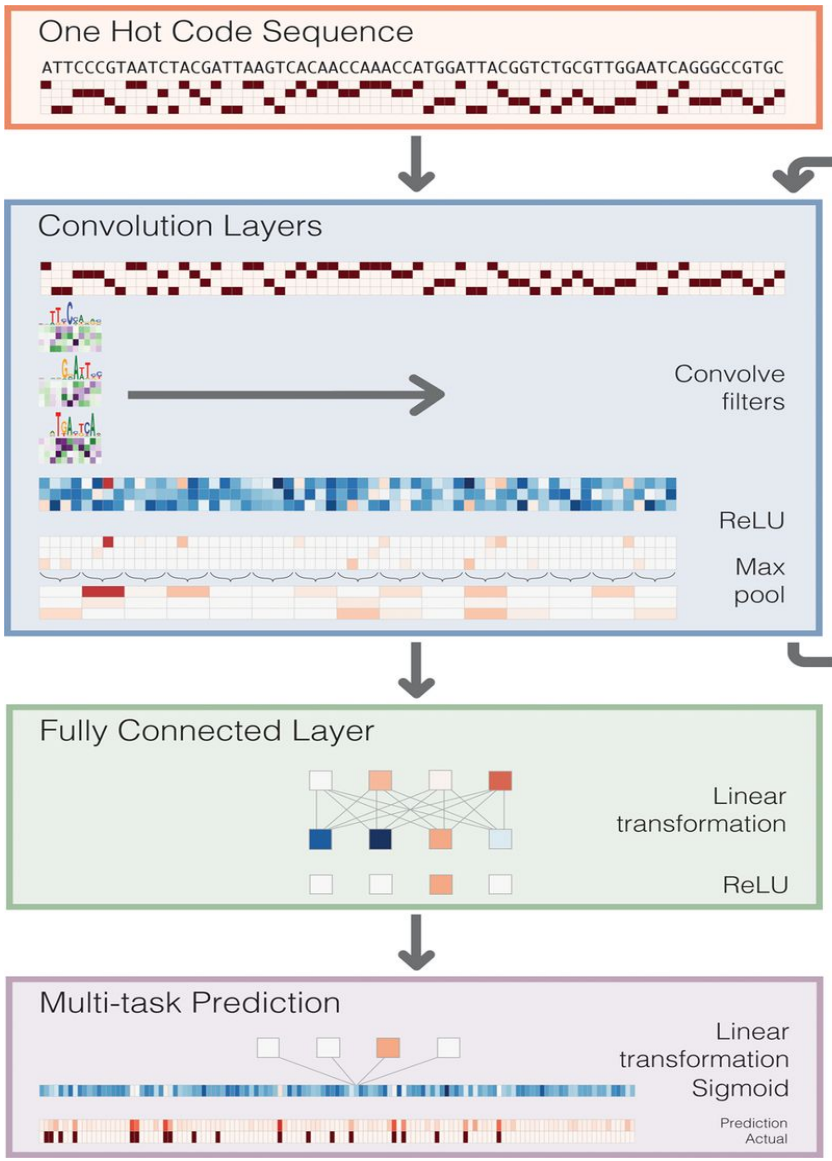
1. Use long seq. 1kbp
2. **multitask architecture**

-> multiple output variables
919 chromatin features (125 DNase features, 690 TF features, 104 histone features)



Zhou, J., & Troyanskaya, O. G. (2015). *Nature Methods*, *12*(10),

BASSET: CNN-BASED ACCESSIBLE GENOME ANALYSIS



1. convert the sequence to a "one hot code" representation

2. scanning weight matrices across the input matrix to produce an output matrix with a row for every convolution filter and a column for every position in the input

3. linear transformation of the input vector and apply a ReLU.

4. linear transformation to a vector of 164 elements that represents the target cells

Kelley et al. (2016). *Genome Research*, *26*(7), 990–

DEEPBIND: PROTEIN–NUCLEIC ACID BINDING SITE PREDICTION

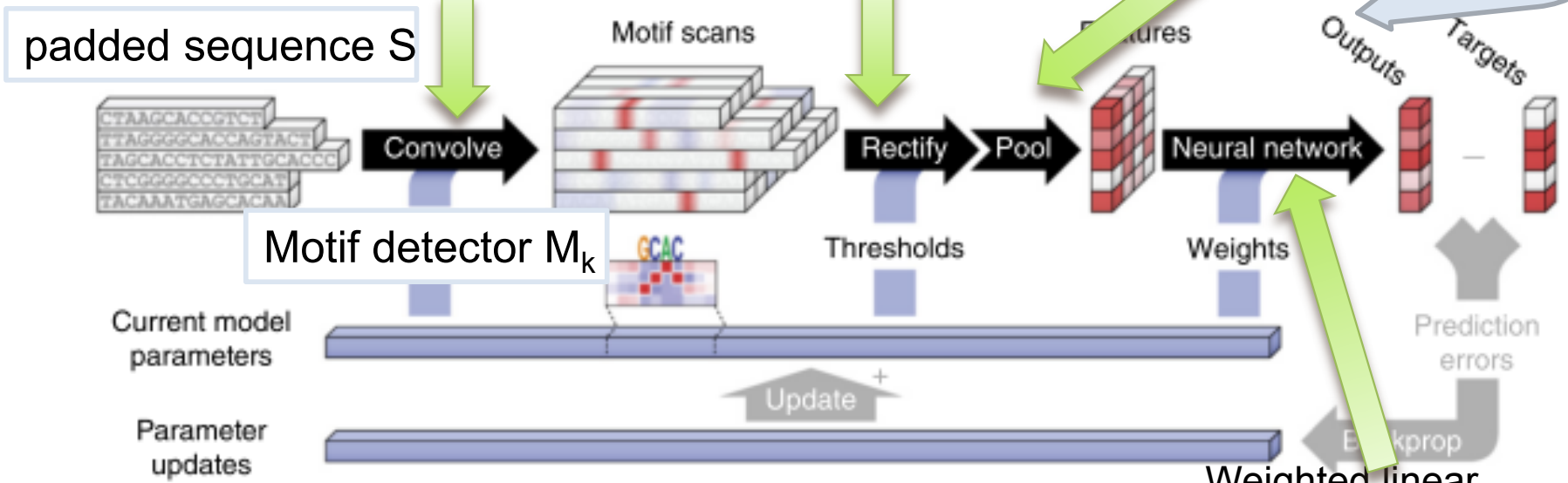
DeepBind is a CNN based supervised learning where
Input: segments of sequences and
labels (output): experimentally determined binding score (ex. ChIP-seq peaks)

$$f(s) = \text{net}_W\left(\text{pool}\left(\text{rect}_b\left(\text{conv}_M(s)\right)\right)\right)$$

$$X = \text{conv}_M(s) \qquad Y = \text{rectify}_b(X) \qquad z = \text{max\_pool}(Y)$$

$$X_{i,k} = \sum_{j=1}^{m}\sum_{l=1}^{4} S_{i+j,l} M_{k,j,l} \qquad Y_{i,k} = \max(0, X_{i,k} - b_k). \qquad z_k = \max(Y_{1,k}, \ldots, Y_{n,k}).$$

Threshold of each motif detector k

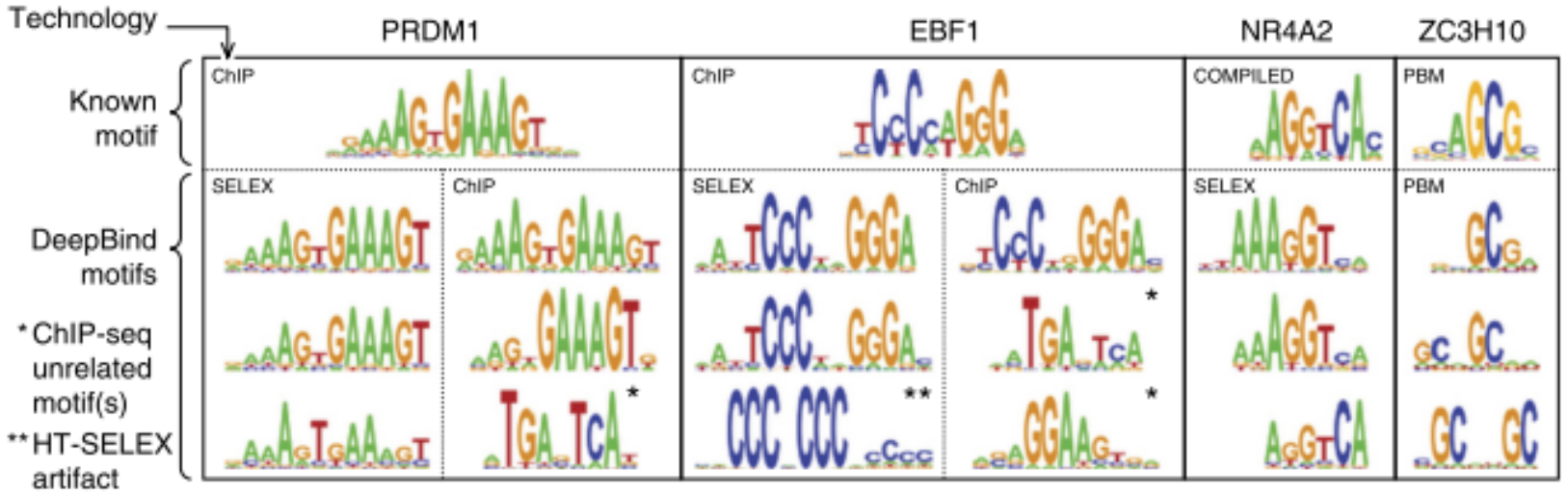


Weighted linear combination of pooled features

Alipanahi et al (2015) *Nature Biotechnology*, *33*(8), 831–

MOTIF EXTRACTION CAPABILITY OF DEEP BIND

The trained motif detector M_k and visualization with sequence logo



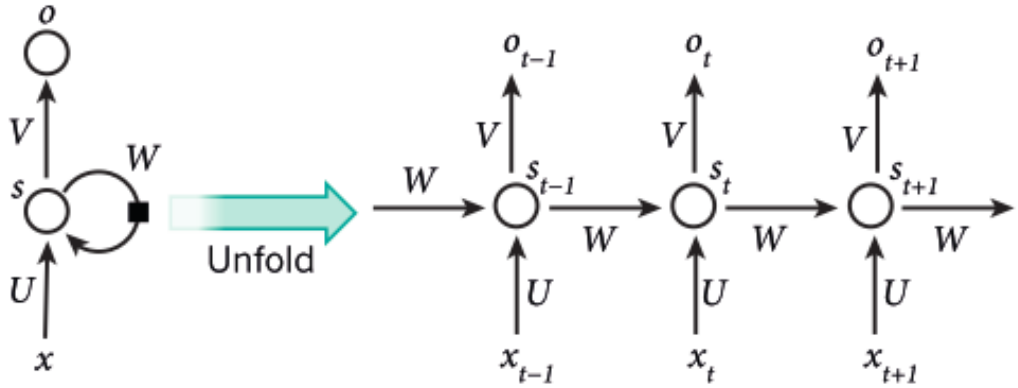
Generating sequence logo to find motifs

1. Feed all sequences from the test set through the convolutional and rectification stages of the DeepBind model,
2. Align all the sequences that passed the activation threshold for at least one position *i*.
3. Generate a position frequency matrix (PFM) and transform it into a sequence logo.

Alipanahi et al (2015) *Nature Biotechnology*, *33*(8), 831–

RNN FOR VARIABLE LENGTH SEQ. INPUT

- Recurrent Neural Network
  - Able to work with sequence input of variable length
  - Capture long range interactions within the input sequences and across outputs
    .
  - Difficult to work with and train



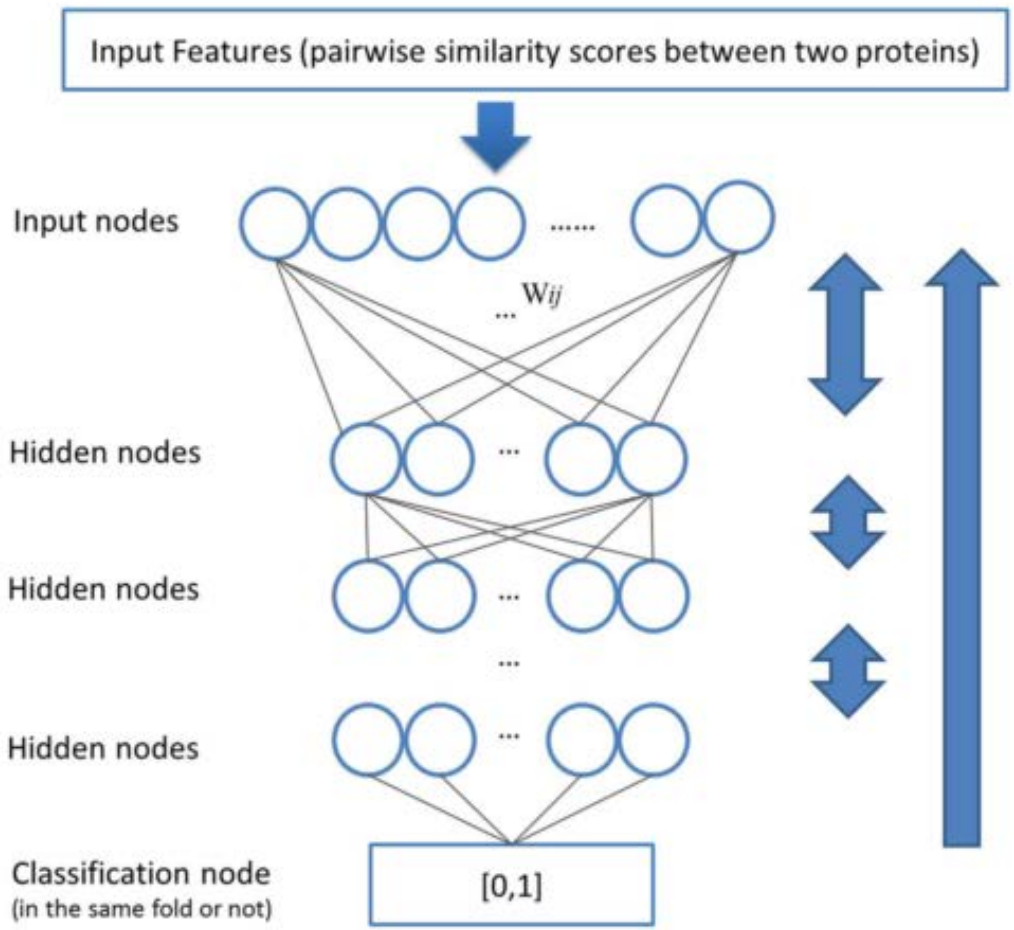
A recurrent neural network and the unfolding in time of the computation involved in its forward computation. (fig 5 of LeCun et al. 2015 Nature)

  - Not many success here

PROTEIN STRUCTURE PREDICTION

- Protein structure prediction methods tend to apply unsuper vised method or combination of NN methods
- Types of unsupervised DNN methods:
  - Restricted Boltzmann Machines (RBM)
  - Deep Belief Networks
- Combination methods
  - Deep Conditional Neural Fields

STACKING RBM IN PROTEIN FOLD RECOGNITION



84 features from five types of sequence alignment and/or protein structure prediction tools

Layer by layer learning with restricted Boltzmann machine (RBM).

Same fold or not

Jo et al. (2015). *Scientific Reports*, *5*, 17573.

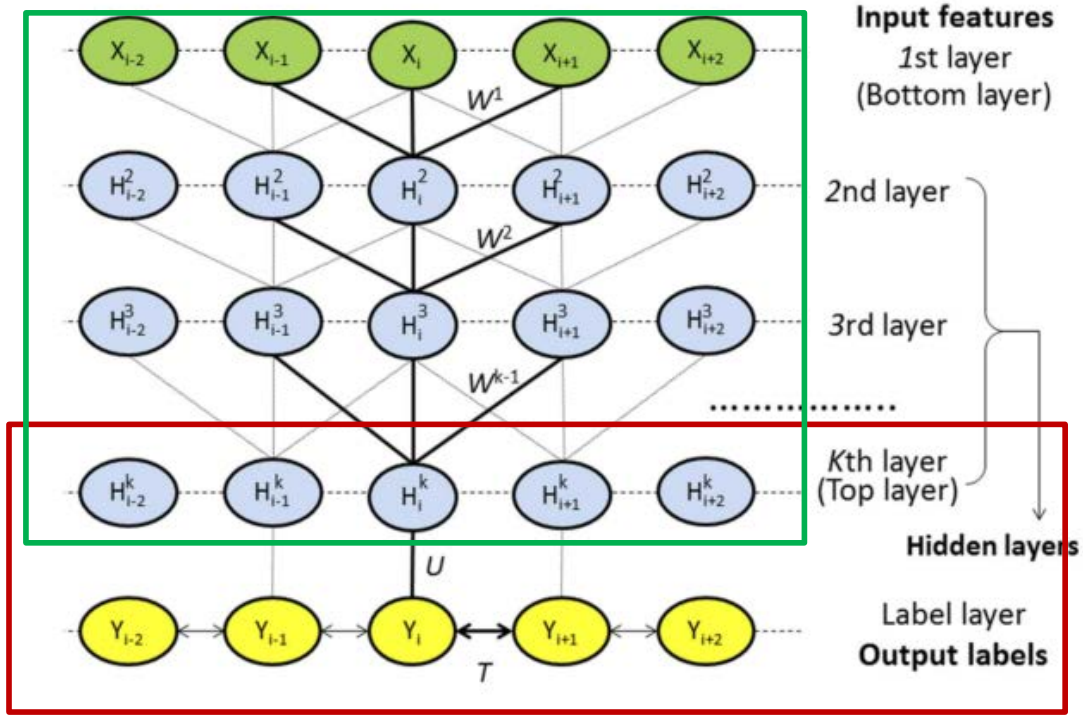
DEEPCNF: SECONDARY STRUCTURE PREDICTION

The architecture of Deep Convolutional Neural Field

fixed window size of 11:
average length of an alpha helix is around eleven residues and that of a beta strand is around six

Xi the associated input features of residue i.

5-7 layer CNN



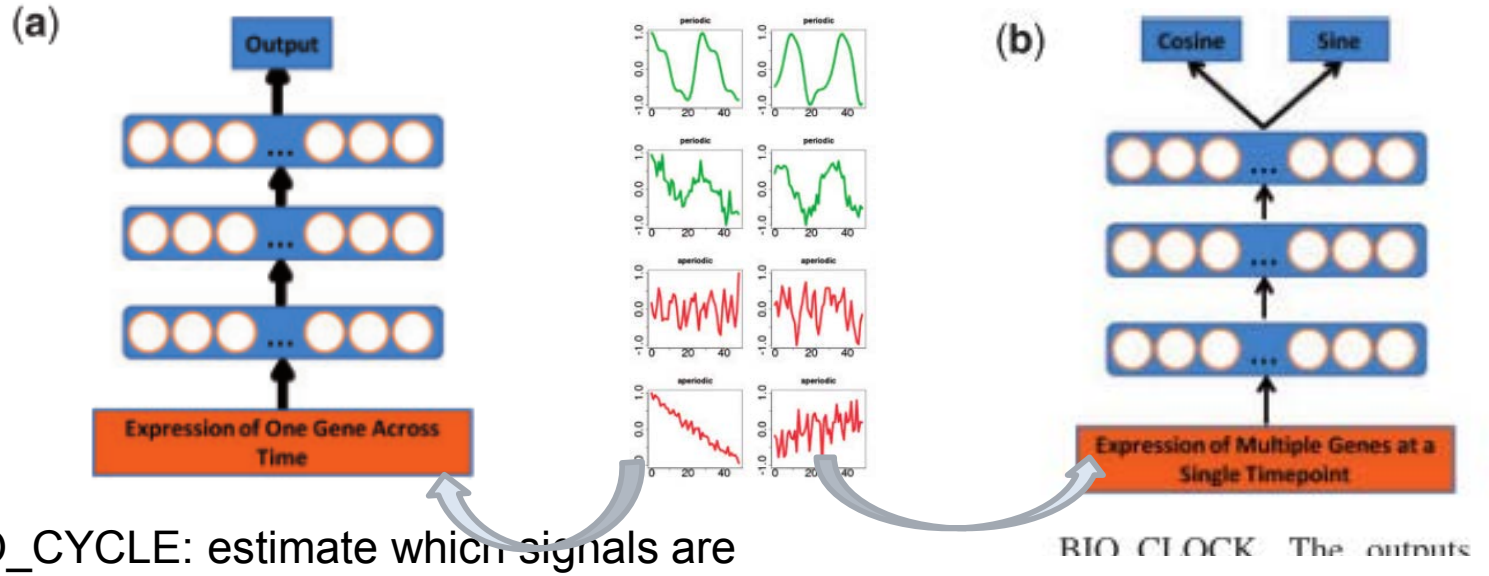
conditional random field (CRF) with U and T being the model parameters.

Calculates conditional probability of SS labels on input features

Wang et al. (2016) *Scientific Reports*, *6*(January), 18962.

CIRCADIAN RHYTHMS

GOAL: inferring whether a given genes oscillate in circadian fashion or not and inferring the time at which a set of measurements was taken



BIO_CYCLE: estimate which signals are periodic in high-throughput circadian experiments, producing estimates of amplitudes, periods, phases, as well as several statistical significance measures.
DATA: data sampled over 24 and 48h

BIO_CLOCK: estimate the time at which a particular single-time-point transcriptomic experiment was carried

Agostinelli, et al. (2016). *Bioinformatics*, *32*(12), i8–i17.

REFERENCE

1. Alipanahi, B., Delong, A., Weirauch, M. T., & Frey, B. J. (2015). Predicting the sequence specificities of DNA- and RNA -binding proteins by deep learning. *Nature Biotechnology*, *33*(8), 831–838.
2. Dahl, G., Jaitly, N., & Salakhutdinov, R. (2014). Multi-task Neural Networks for QSAR Predictions. *arXiv Preprint arXiv: 1406.1231*, 1–21.
3. Eickholt, J., & Cheng, J. (2012). Predicting protein residue-residue contacts using deep networks and boosting. *Bioinfo rmatics*, *28*(23), 3066–3072.
4. Eickholt, J., & Cheng, J. (2013). DNdisorder: predicting protein disorder using boosting and deep networks. *BMC Bioinf ormatics*, *14*(1), 88.
5. Gawehn, E., Hiss, J. A., & Schneider, G. (2016). Deep Learning in Drug Discovery. *Molecular Informatics*, *35*(1), 3–14.
6. Jo, T., Hou, J., Eickholt, J., & Cheng, J. (2015). Improving Protein Fold Recognition by Deep Learning Networks. *Scien tific Reports*, *5*, 17573.
7. Kelley, D. R., Snoek, J., & Rinn, J. L. (2016). Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Research*, *26*(7), 990–999.
8. Leung, M. K. K., Xiong, H. Y., Lee, L. J., & Frey, B. J. (2014). Deep learning of the tissue-regulated splicing code. *Bioin formatics*, *30*(12), 121–129.
9. Sønderby, S. K., & Winther, O. (2014). Protein Secondary Structure Prediction with Long Short Term Memory Network s. Retrieved from http://arxiv.org/abs/1412.7828
10. Wang, S., Peng, J., Ma, J., & Xu, J. (2016). Protein Secondary Structure Prediction Using Deep Convolutional Neural Fields. *Scientific Reports*, *6*(January), 18962.
11. Wang, S., Weng, S., Ma, J., & Tang, Q. (2015). DeepCNF-D: Predicting Protein Order/Disorder Regions by Weighted Deep Convolutional Neural Fields. *International Journal of Molecular Sciences*, *16*(8), 17315–17330.
12. Zhang, S., Zhou, J., Hu, H., Gong, H., Chen, L., Cheng, C., & Zeng, J. (2015). A deep learning framework for modeling structural features of RNA-binding protein targets. *Nucleic Acids Research*, *44*(4), 1–14.
13. Zhou, J., & Troyanskaya, O. G. (2015). Predicting effects of noncoding variants with deep learning-based sequence m odel. *Nature Methods*, *12*(10), 931–4.

REFERENCE TO REVIEWS

1. **Angermueller, C., Pärnamaa, T., Parts, L., & Oliver, S. (2016). Deep Learning for Computational Biology. *Molecular Systems Biology*, (12), 878.**
2. Gawehn, E., Hiss, J. A., & Schneider, G. (2016). Deep Learning in Drug Discovery. *Molecular Informatics*, *35*(1), 3–14.
3. Mamoshina, P., Vieira, A., Putin, E., & Zhavoronkov, A. (2016). Applications of Deep Learning in Biomedicine. *Molecular Pharmaceutics*, acs.molpharmaceut.5b00982.