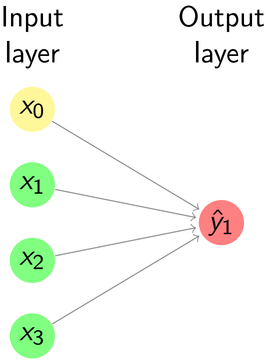
Mixing Old and New with Neural Networks in Communications

Peter Hartig

September 5, 2019

Overview

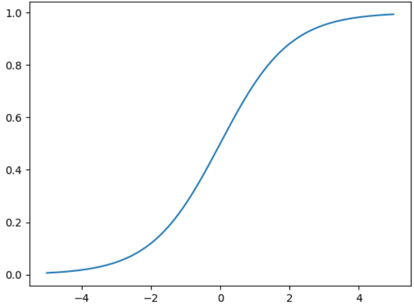
Building Blocks: Single Neuron



$$Output = f(\sum_{n=1}^{3} w_i * i_i + bias)$$

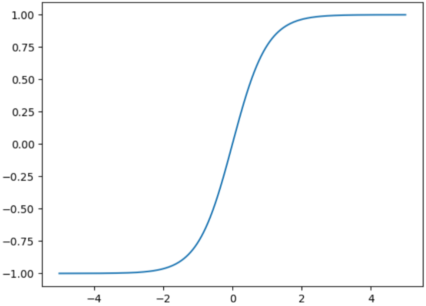
Activation function incorporates non-linearity so long as there is tractable $\Delta f()$ with respect to $w_{i,j}$. Without non-linear functions, a network of arbitrary depth can be collapsed to a single layer . [1]

Activation Functions: Sigmoid



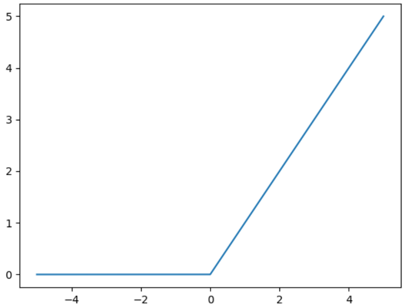
$$\sigma(x) = \frac{1}{1 + e^{w_i^T x + b}} \tag{1}$$

Activation Functions: Tanh



$$tanh(x)_i = \frac{e^{x_i} - e^{-x_i}}{e^{x_i} + e^{+x_i}} \tag{2}$$

Activation Functions: ReLU

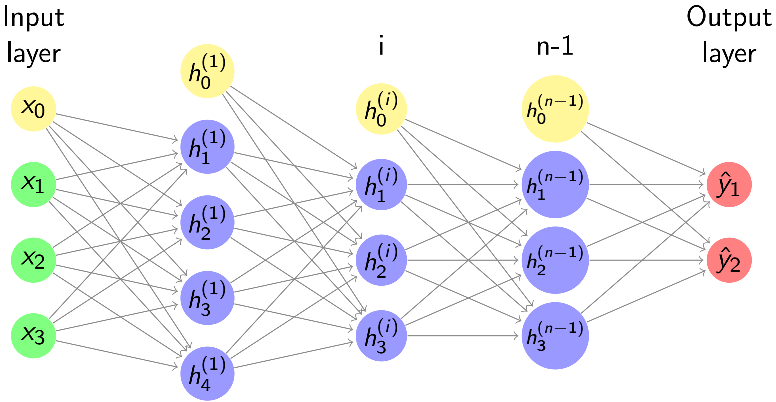


$$f(x) = max(0, Wx + b) \tag{3}$$

Activation Functions: Softmax

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{4}$$

Full Network



Prior to the activation function of nodes. Each layer undergoes a transformation into a |l+1| dimensional space.

Training

$$J(W,b;x,y) = [\frac{1}{m} * \sum_{i=1}^{m} J(W,b,x^i,y^i)] + \frac{\lambda}{2} * \sum_{l=1}^{n_l-1} * \sum_{l=1}^{s_l} * \sum_{l=1}^{s_{l+1}} (W_{ji}^{(i)})^2 \quad (5)$$

Figure: Cost function over a batch of m training examples (x_i,y_i) [2]

No analytic solution -> Iteratively update weights

- Gradient descent w.r.t weights and bias (batch/stochastic)
- Newton's Method (requires Hessian)
- Momentum learning ... lots of tweaks

Cost Function

$$J(W,b;x,y) = [\frac{1}{m} * \sum_{i=1}^{m} J(W,b,x^{i},y^{i})] + \frac{\lambda}{2} * \sum_{l=1}^{n_l - 1} * \sum_{l=1}^{s_l} * \sum_{l=1}^{s_{l+1}} (W_{ji}^{(i)})^2 \tag{6}$$

- Quadratic

$$\frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}||y_i - a_i^l||^2 \tag{7}$$

- ML parametarized model (minimum KL Divergence) [2]

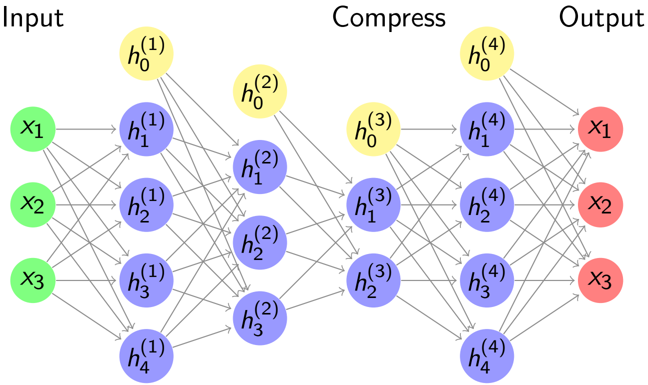
$$(W,b)_{ML} = argmax_{W,b} Pr(Y|X,W,b) \tag{8}$$

Without regularization this searches for the ML model for the **training** data [3]

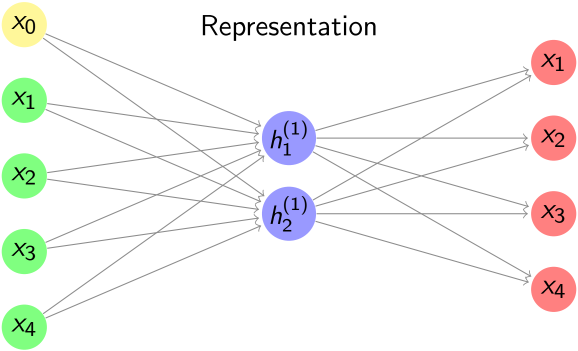
Regularization (Training Data is Expensive)

ML converges asymptotically but we usually don't have infinite data so we enforce priors

Autoencoders and Source Coding



Priciple Component Analysis as an Autoencoder



Of interest here is the Matrix W^1 with elements W^1_{ij}. If we use the cost function $||h_{W,b}(x) - y||^2$, this matrix will span the eigenspace of the dataset similarly to the way PCA selects the span of eigenvectors corresponding to the largest eigenvalues.

Interpretting Regularizing with Autoencoders

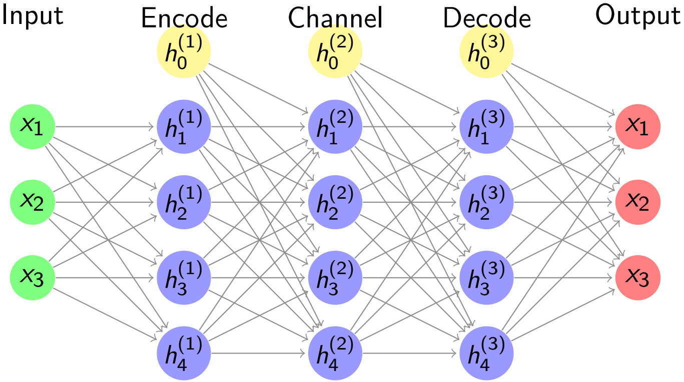
Classification

Regularization corresponds to priors on parameters of the training data pdf model.

Autoencoder

Autoencoders learn latent variables so regularization corresponds priors on these latent variables.

Denoising Autoencoder



Channel model can impact how we train [4]

Training Deep Autoencoders

Individual layers are often "pre-trained" to learn an idea of the latent variables they should be moving towards. [14.3 3]

Dividing Channel Resources

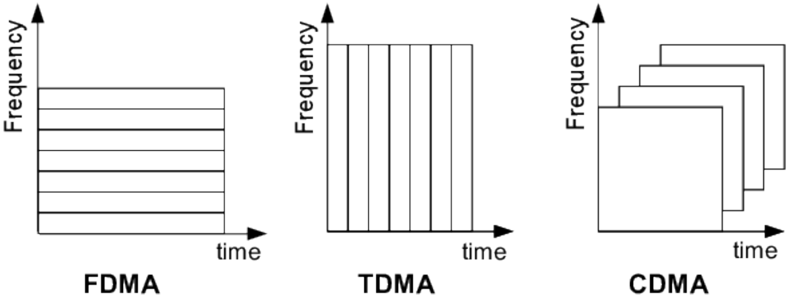


Figure: Multiple Access Schemes (include SDMA/NOMA?)

Sinusoids in LTI systems

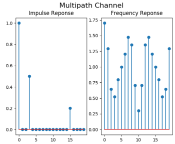


Figure: Multipath Channel

$$x(t) = e^{j2\pi f_c t}$$

$$\sum_{n=1}^{n} c_n * \delta(t - \tau_n)$$

Solution step 1

Truncate the sinusoid with a rectangular window in time...
Convolves frequency with sinc

Solution step 2

In discrete time, a finite sinusoid can still be represented by a delta in discrete frequency.

OFDM Outline

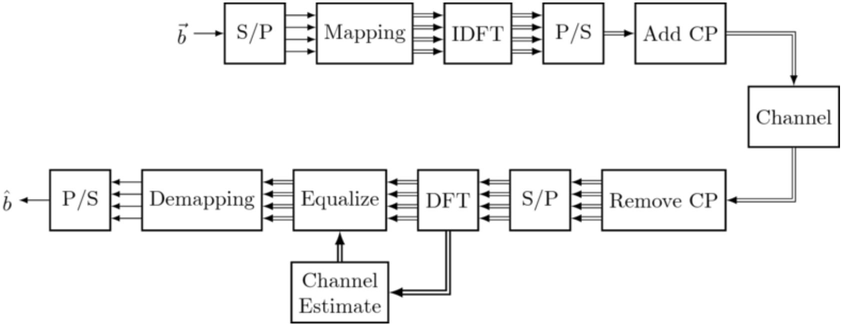


Figure:[1]

Discrete frequency multiplied by channel frequency response -> circular convolution in time.

[1] 5.

Multiplexing

ORTHOGONAL Frequency Division MULTIPLEXING

N-point time sample contains N frequencies at modulated phase/magnitude. The result can carry information on all N orthogonal sinusoids.

*Users are assigned to sets of frequencies within an N-point DFT.

Synchonization

Orthogonality of the N "sub-carriers" depends on equally spaced samples.
Sampling period and total number of samples need to be synchronized to preserve sub-carrier orthogonality. This results in strict requirements on transmit/receive oscillator synchronization.
-Show how synchronization mismatch can cause problems Just include this with mention that it will be important for next section [6]

Benefits and drawbacks

- Large Peak to Average Power Ratio (**PAPR**)
  - Variance of single die roll: PAPR = 2.37

$$E[|x|^2] = 15.166 \tag{9}$$

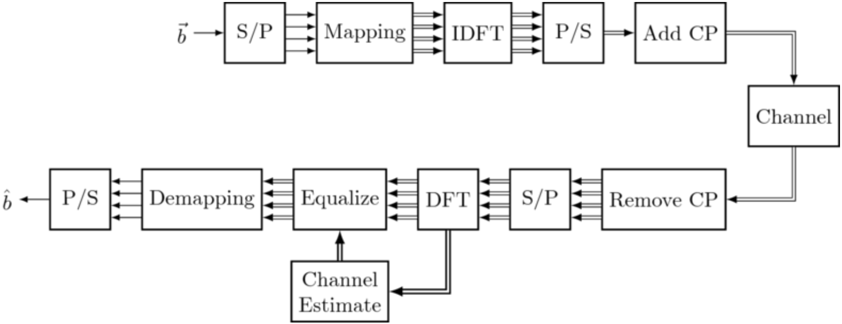
  - Variance of sum of two dice rolled: PAPR = 2.62

$$E[|x_1 + x_2|^2] = 54.833 \tag{10}$$

- Cycle Prefix overheard proportional to channel response length
- Sensitive to asynchronous transmission

Current Deployment of OFDM

OFDM Outline



End-to-End Autoencoder

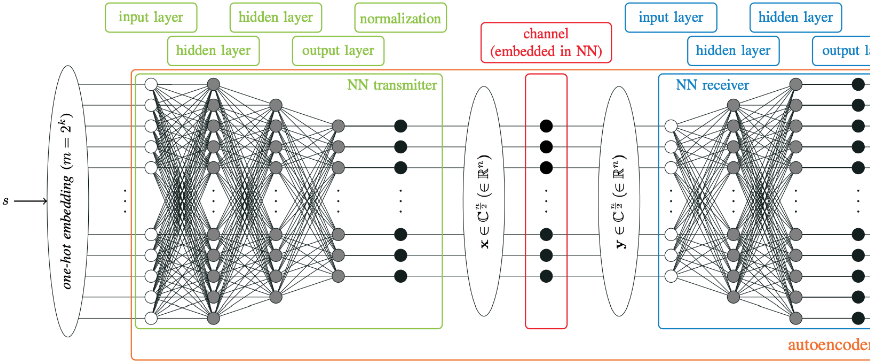


Figure:[2]

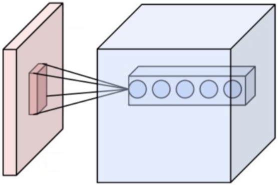
Naive Modeling

Intuitively this gives the network freedom to learn the best channel equalization. Classifying will require the network to perform a marginalization over all variables in the channel.

Synchonization

Why this might be difficult. Resolved in previous work using additional NN. OFDM allows for simple sampling synchronization using autocorrelation with the cyclic prefix.

Applying domain knowledge to NN Architectures



- Reduce required training data and increase training speed
- Improved generalization

RTN – OFDM

We take length w_{fft} inverse fft over w_{fft} messages with each message being length $\frac{n}{2}$ long. The result is that we send $\frac{n}{2}$ symbols over each carrier in the multicarrier system. All $\frac{n}{2}$ can be equalized with the same pilot.

Channel Model

Training

Discuss the two stages of training

Sequence Detector

Equalization

- Pilot tones with MMSE Equalizer
- Pilot tones without explicit equalizer
- No pilot or explicit equalizer

Normally we have models for the channels and these models usually incorporate random variables as parameters.The current state of the channel is one realization of these random parameters

Pure ML approach no longer assumes a specific model until trained

This study considers the case in which we assume a model but allow for estimation of model parameters

Each decoded block provides information with which to update the model

End result achieves similar performance to CSI based (?? dB)

ViterbiNet outperforms standard Viterbi when there is CSI uncertainty (uncoded)

$$\hat{s}(y) = argmin_s \sum_{i=1}^{t} -log(P_{Y[i]|s}(y[i]|s) \tag{11}$$

Knowing $-log(P_{Y[i]|s}(y[i]|s)$ provides metrics for Viterbi trellis

Discuss the two stages of training

Further work for online training utilizing FEC to provide a new

Future Directions

Question?

P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural networks*, vol. 2, no. 1, pp. 53–58, 1989.

A. Ng *et al.*, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

A. Felix, S. Cammerer, S. Dörner, J. Hoydis, and S. Ten Brink, "Ofdm-autoencoder for end-to-end learning of communications systems," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, IEEE, 2018, pp. 1–5.

https://dspillustrations.com/pages/posts/misc/python-ofdm-example.html.

M. Sandell, J. v. d. Beek, and P. O. Börjesson, "Timing and