Class is 170.

Announcements

Matlab Grader homework,
1 and 2 (of less than 9) homeworks Due 22 April **tonight,** Binary graded.
167, 165,164 has done the homework. (**If you have not done HW talk to me/TA!**)
Homework 3 due **5 May**
Homework 4 (SVM +DL) due ~24 May

Jupiter "GPU" home work released Wednesday. Due 10 May

Projects: 39 Groups formed. Look at Piazza for help.
Guidelines is on Piazza
May 5 proposal due. TAs and Peter can approve.

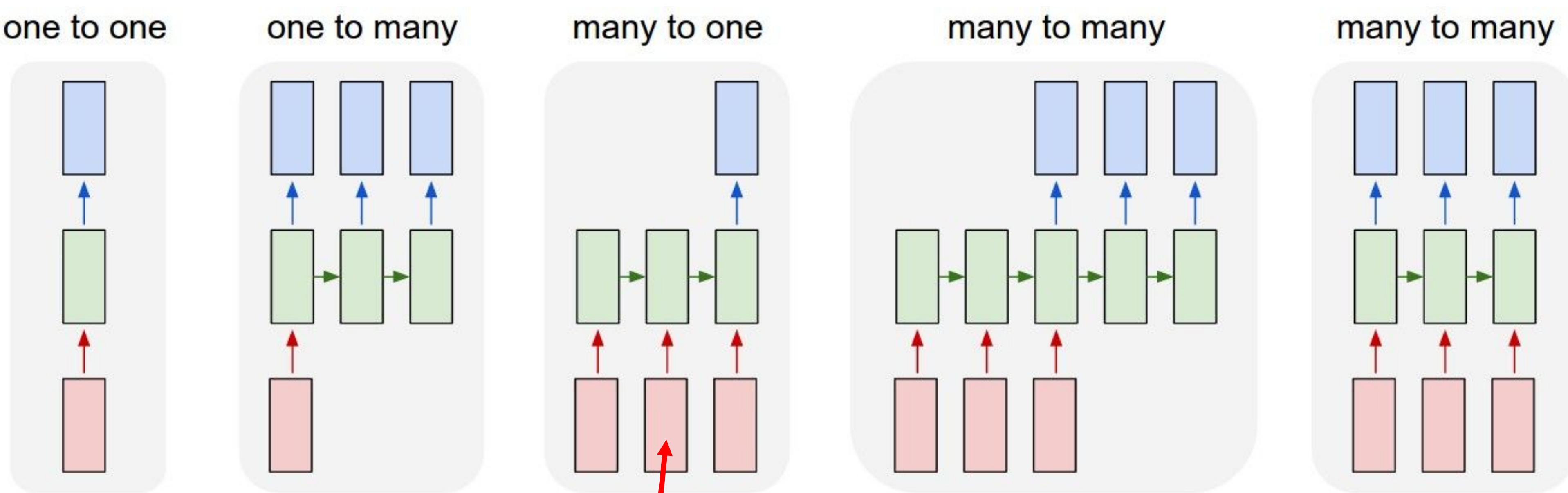
Today:

- Stanford CNN 10, CNN and seismics

Wednesday

- Stanford CNN 11, SVM, (Bishop 7),
- Play with Tensorflow playground before class http://playground.tensorflow.org
  Solve the spiral problem

Recurrent Neural Networks: Process Sequences



e.g. **Image Captioning**
image -> sequence of words

e.g. **Sentiment Classification**
sequence of words -> sentiment

e.g. **Machine Translation**
seq of words -> seq of words

Vanilla Neural Networks

Video classification on frame level

Recurrent Neural Network

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

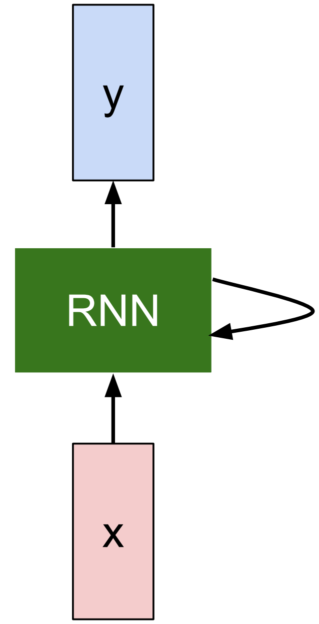
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

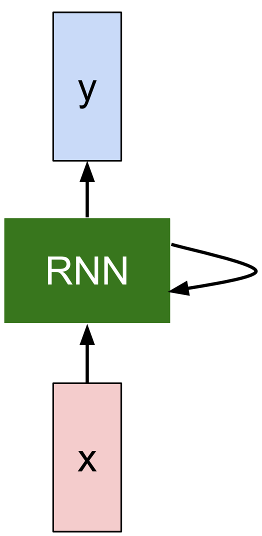
input vector at some time step

some function with parameters W



(Vanilla) Recurrent Neural Network

The state consists of a single *"hidden"* vector **h**:

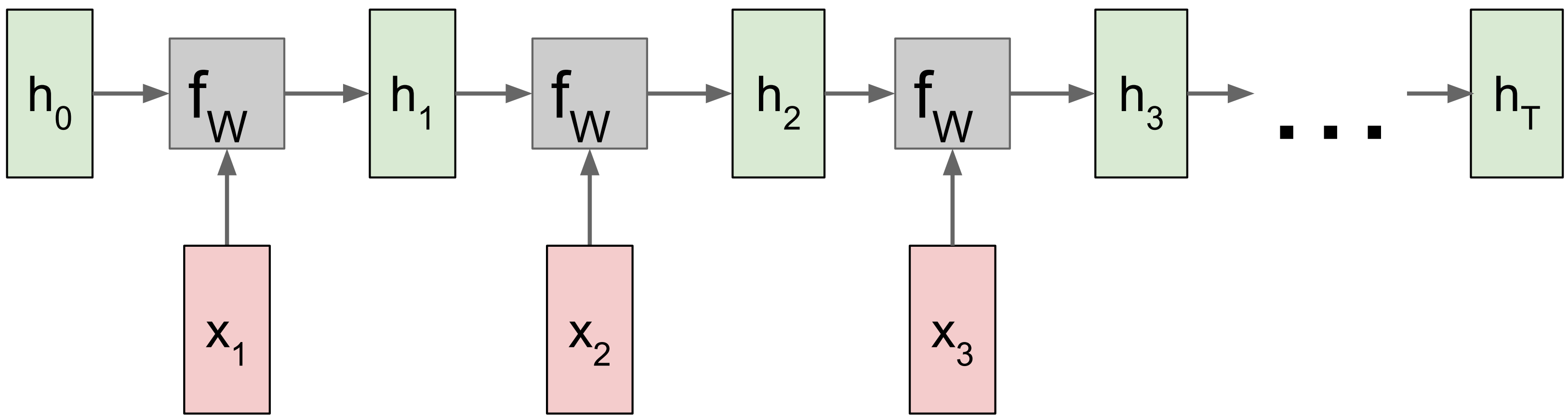


$$h_t = f_W(h_{t-1}, x_t)$$

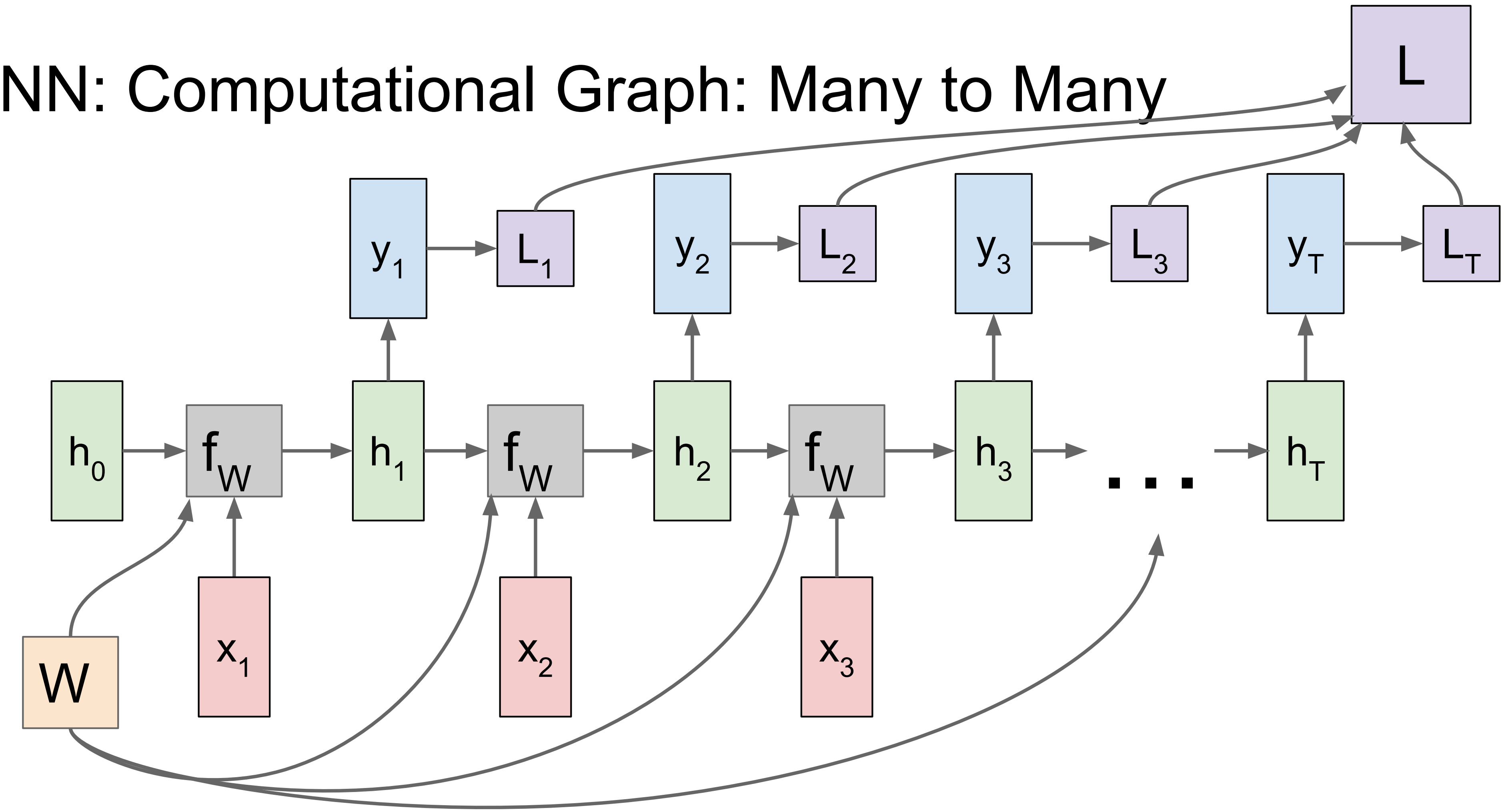
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

RNN: Computational Graph



RNN: Computational Graph: Many to Many



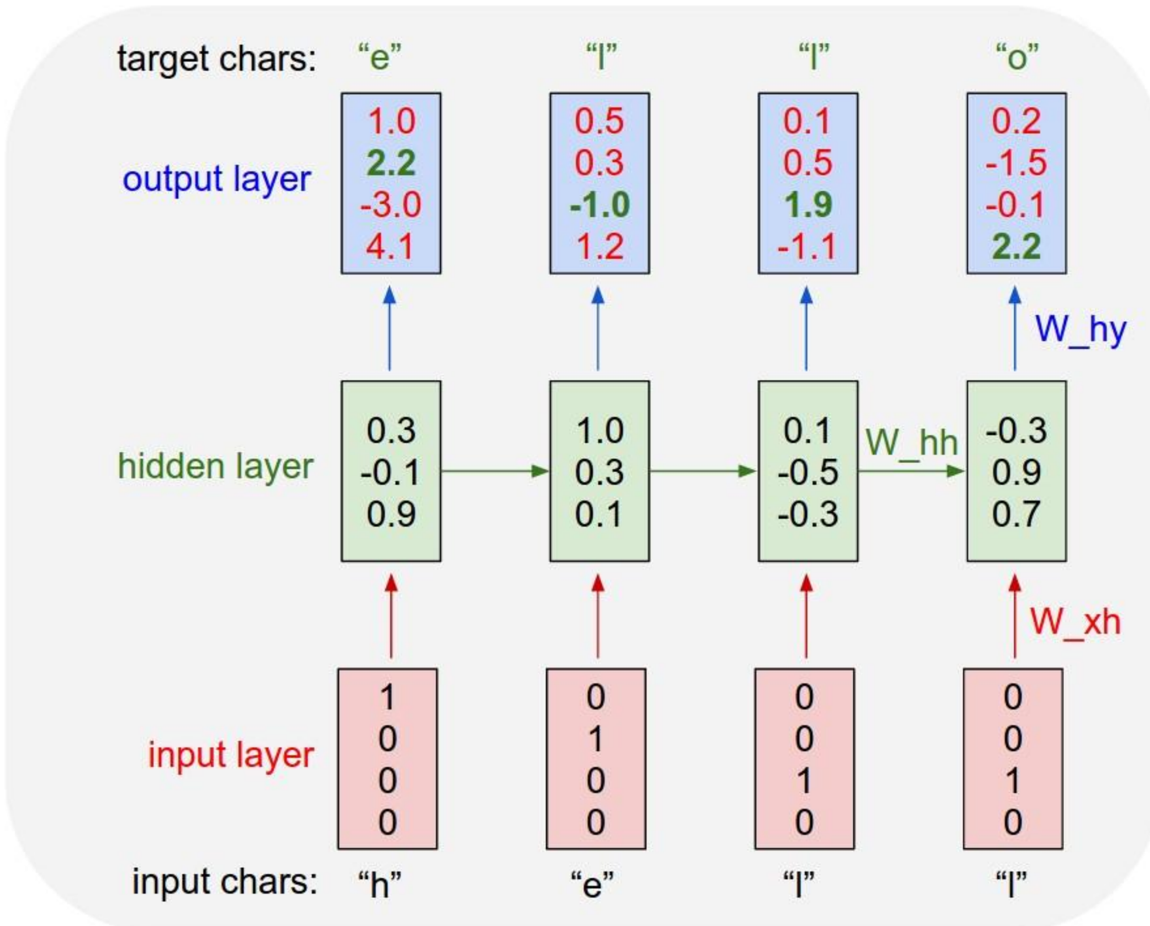
Example:
Character-level
Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

$$\boxed{h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)}$$

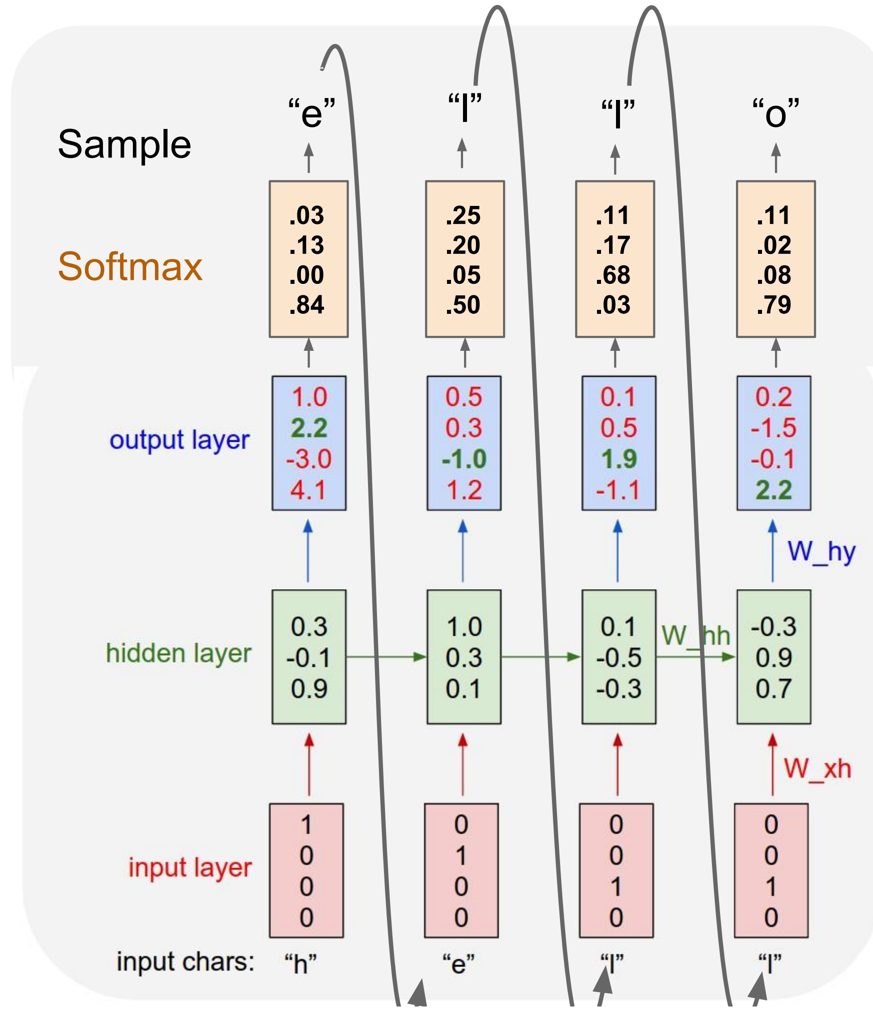
$$y_t = W_{hy}h_t$$



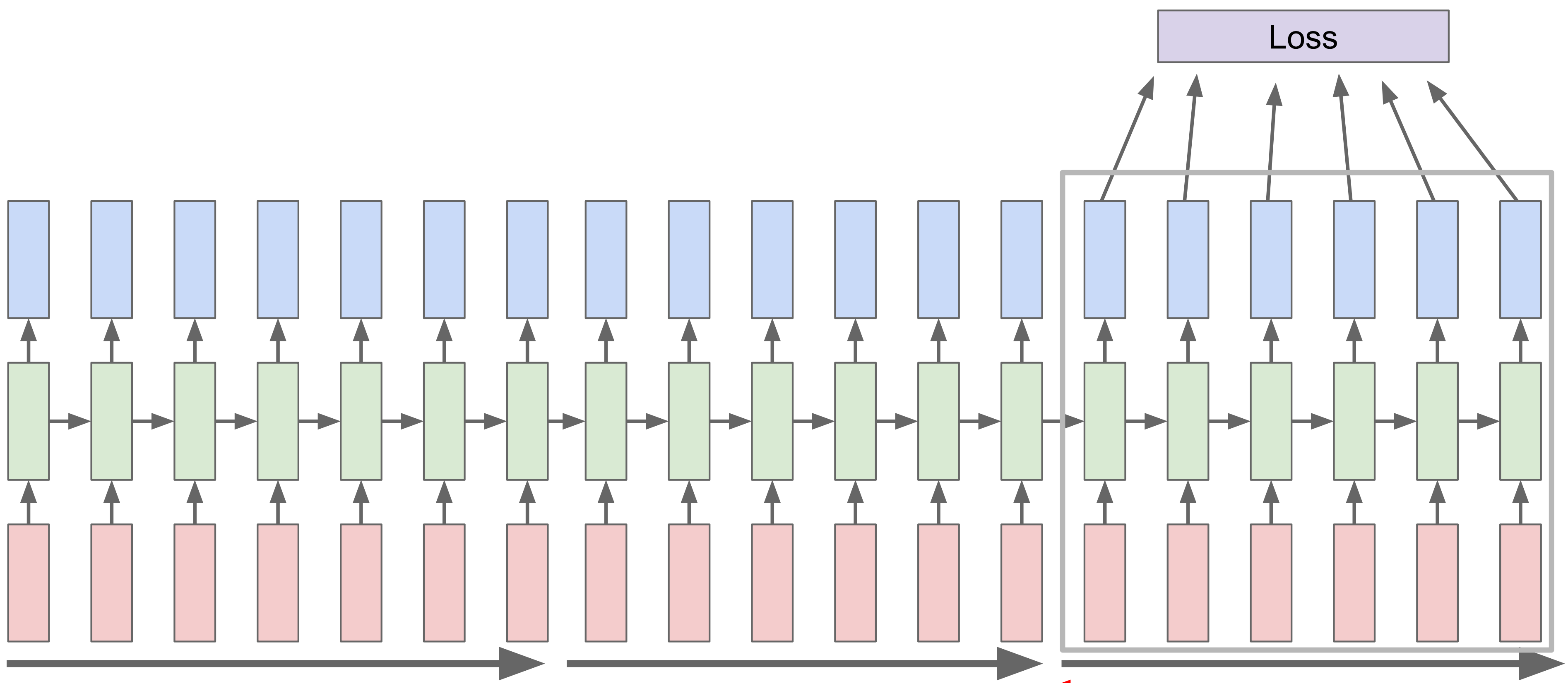
Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model



Truncated Backpropagation through time



Long Short Term Memory (LSTM)

Vanilla RNN

$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

LSTM

Cell state

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

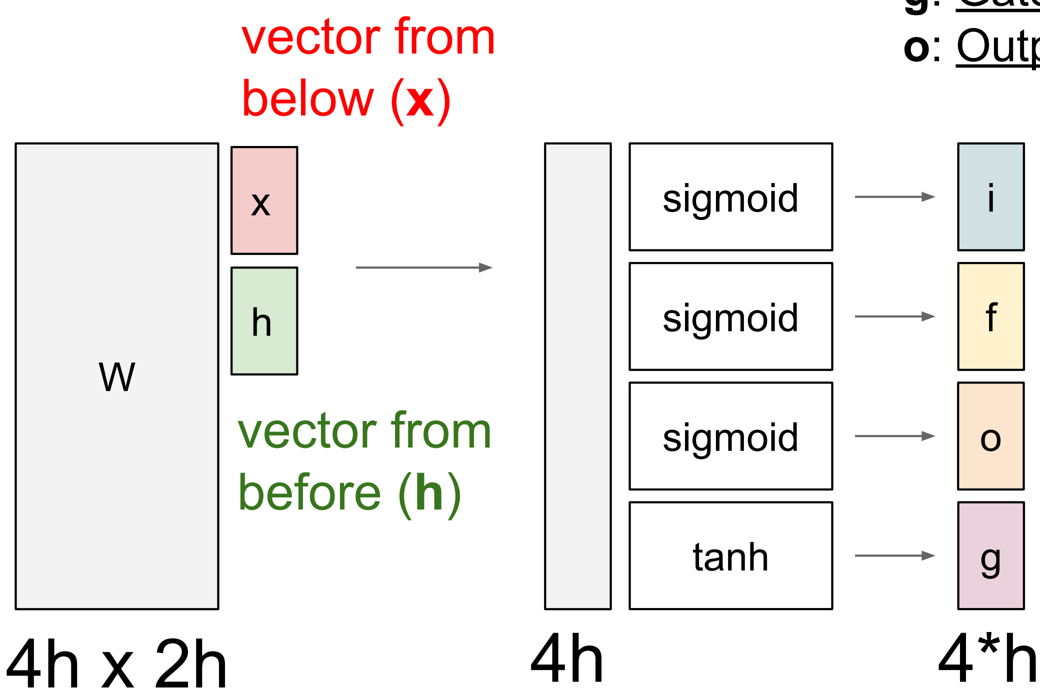
Hochreiter and Schmidhuber, “Long Short Term Memory”, Neural Computation 1997

Hidden state h(t)
Cell state c(t)

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]

f: Forget gate, Whether to erase cell
i: Input gate, whether to write to cell
g: Gate gate (?), How much to write to cell
o: Output gate, How much to reveal cell



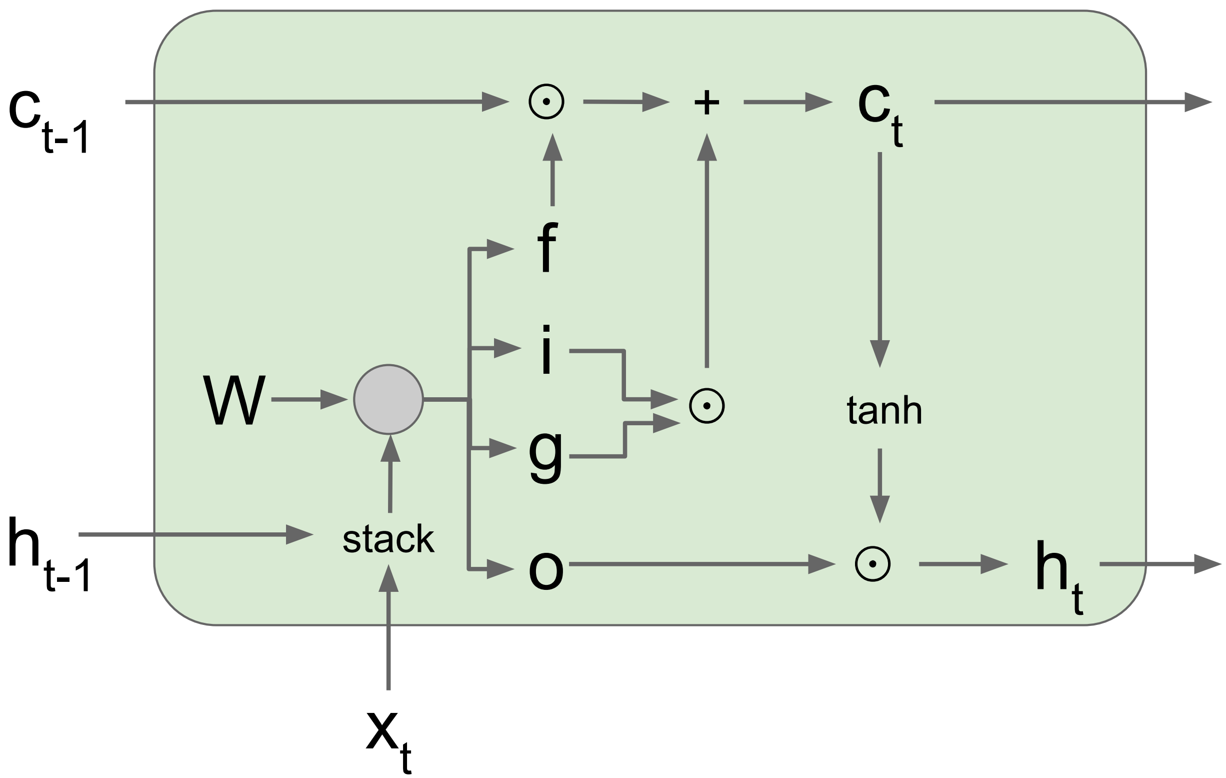
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Long Short Term Memory (LSTM)

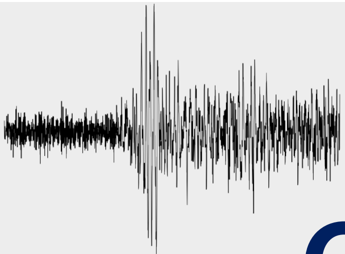
[Hochreiter et al., 1997]

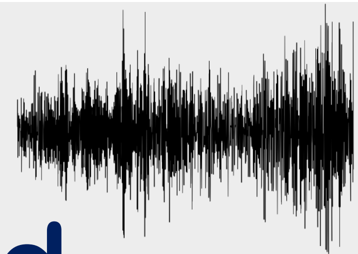


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$





Classifying emergent and impulsive seismic noise in continuous seismic waveforms

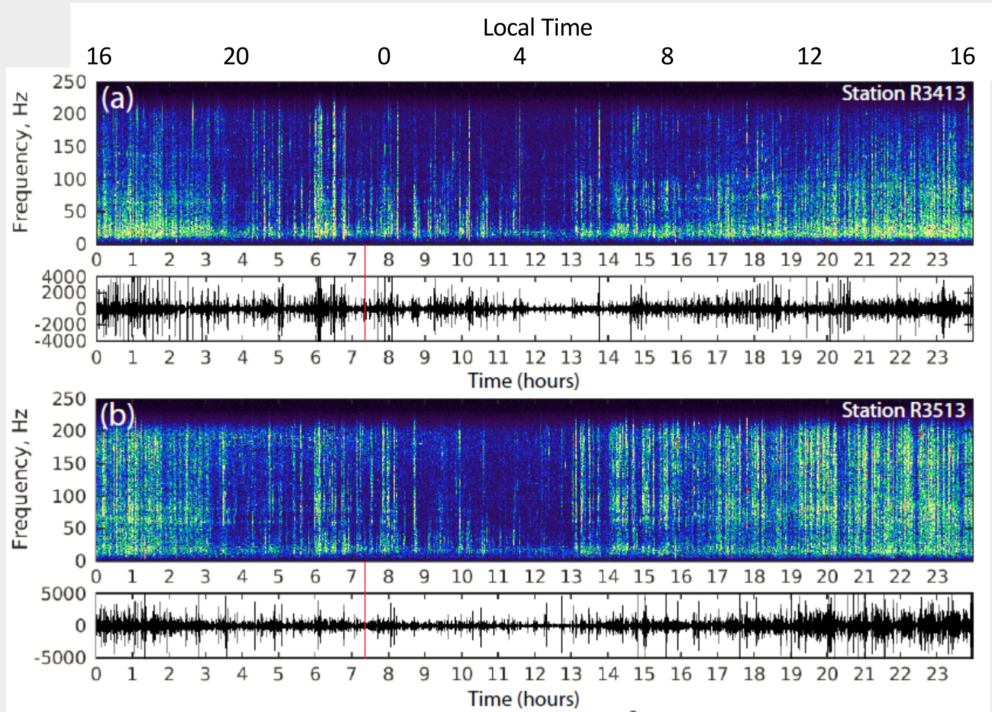
Christopher W Johnson

NSF Postdoctoral Fellow

UCSD / Scripps Institution of Oceanography

The problem

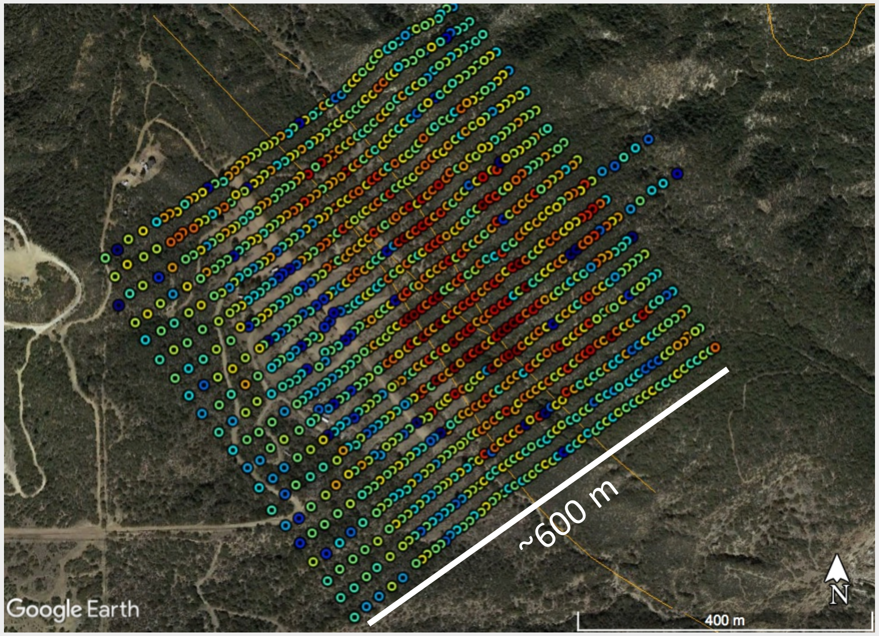
- Identify material failures in the upper 1 km of the crust
- Separate microseismicity (M<1)
- 59-74% of daily record is not random noise
  - Earthquake <1%
  - Air-traffic ~7%
  - Wind ~6%
- Develop new waveform classes
  - air-traffic, vehicle-traffic, wind, human, instrument, etc.



Ben-Zion et al., GJI 2015

The data

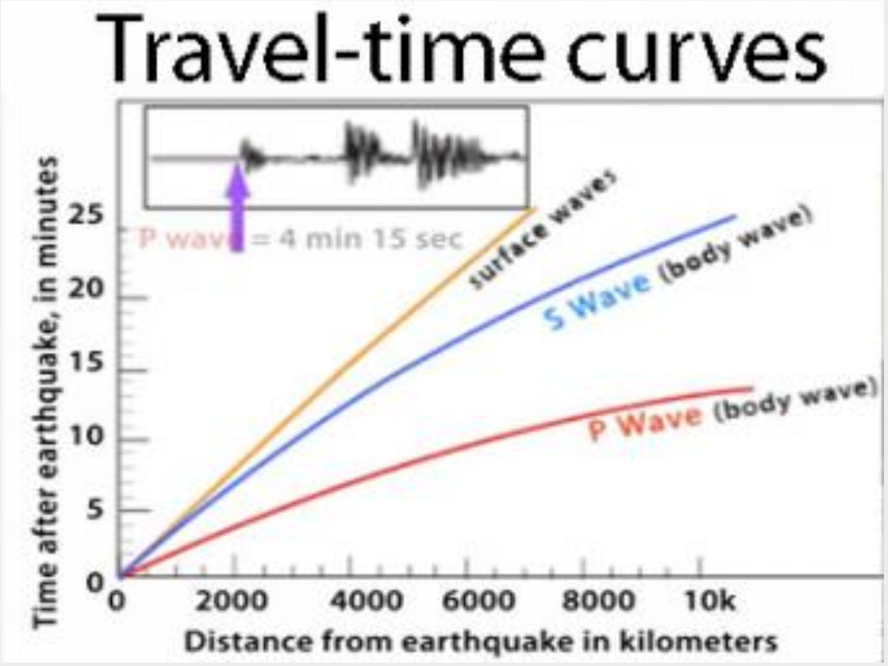
- 2014 deployment for ~30 days
  - 1100 vertical 10Hz geophones
  - 10-30 m spacing
  - 500 samples per second
  - 1.6 Tb of waveform data
- Experiment design optimized to explore properties and deformation in the shallow crust; upper 1km
  - High res. velocity structure
  - Imaging the damage zone
  - Microseismic detection



Ben-Zion et al., GJI 2015

Earthquake detection

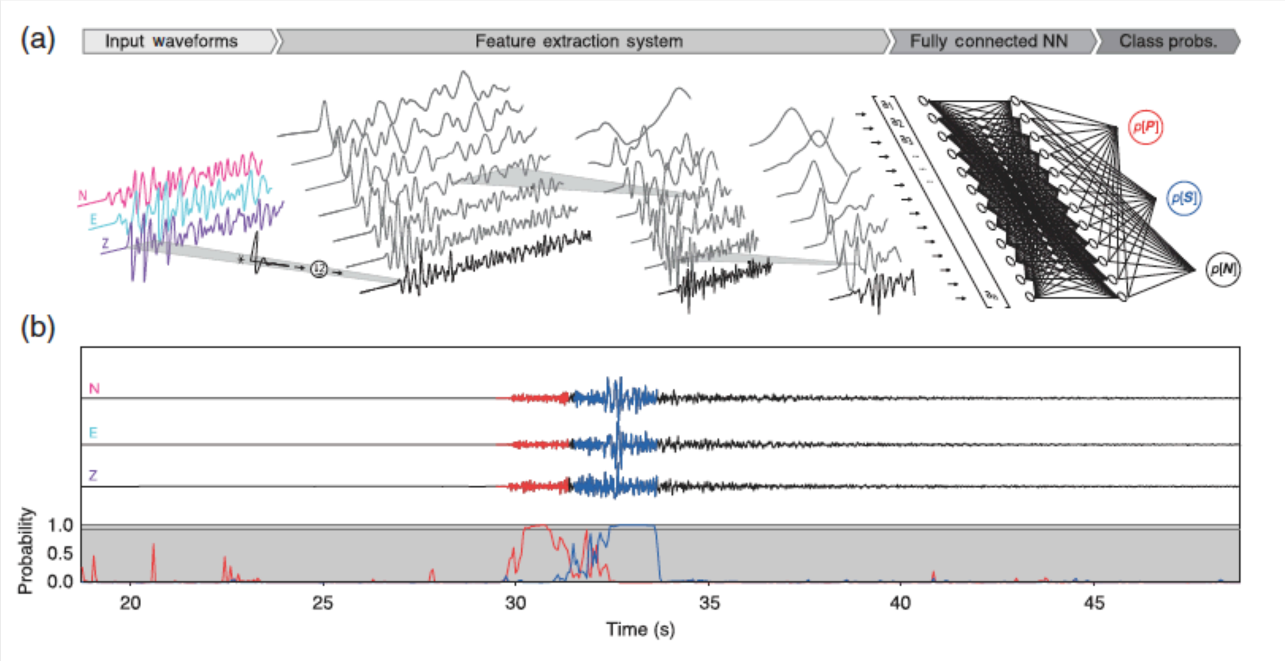
- **Distributed region sensor network**
- **Source location random, but expected along major fault lines**
- **P-wave (compression) & S-wave (shear) travel times**
- **Grid search / regression to obtain location**
- **Requires robust detections for small events**



from IRIS website

Recent advances in seismic detection

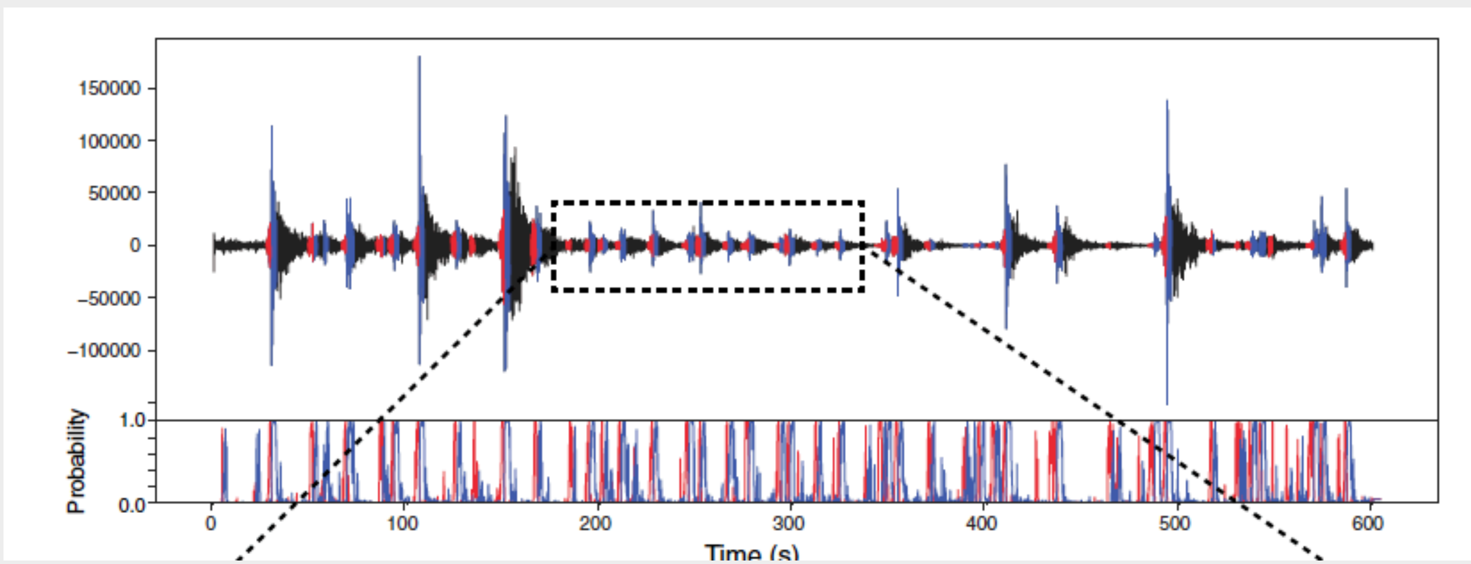
- **3-component seismic data (east, north, vert)**
- **CNN**
  - **Each component is channel**
  - **Softmax probability**



Ross et al., BSSA 2018

Recent advances in seismic detection

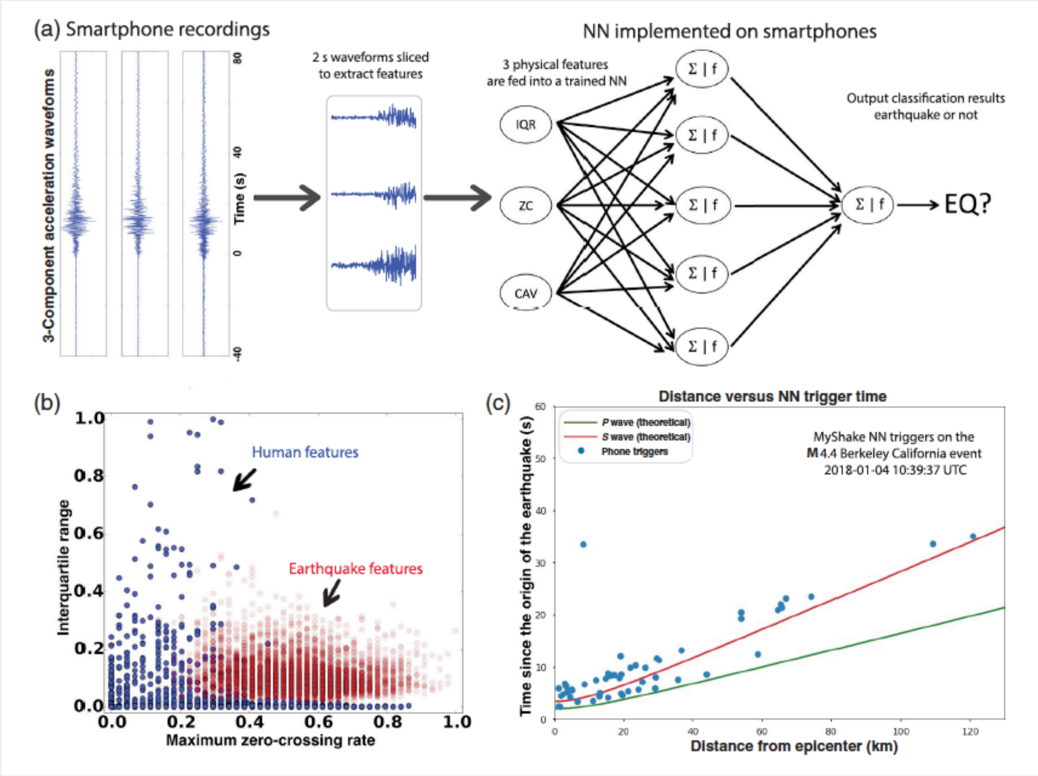
- Example of continuous waveform
- Every sample is classified as noise, P-wave, or S-wave
  - Outperforms traditional methods utilizing STA/LTA



Ross et al., BSSA 2018

Future direction is seismology

- **Utilize accelerometer in everyone's smart phone**



Kong et al., SRL, 2018

Research Approach and Objectives

- Need labeled data. This is >80% of the work!
  - Earthquakes
    - Arrival time obtained from borehole seismometer within array
  - Define noise
    - Develop new algorithm to produce 2 noise labels
  - Signal processing / spectral analysis
    - Calculate earthquake SNR
      - Discard events with SNR ~1
    - Waveforms to spectrogram
      - Matrix of complex values
      - Retain amplitude and phase
    - Each input has 2 channels
      - This is not a rule, just a choice

Deep learning model – Noise Labeling

- Labeling is expensive
  - 1 day with 1100 geophones
    - ~1800 CPU hrs on 3.4GHz Xeon Gold (1.7hr/per daily record)
    - ~9000 CPU hrs on 2.6 GHz Xeon E5 on COMET (5x decrease)
- Noise training data
  - 1s labels
  - 1100 stations for 3 days
  - Use consecutive 4 s intervals
  - Calculate spectrogram

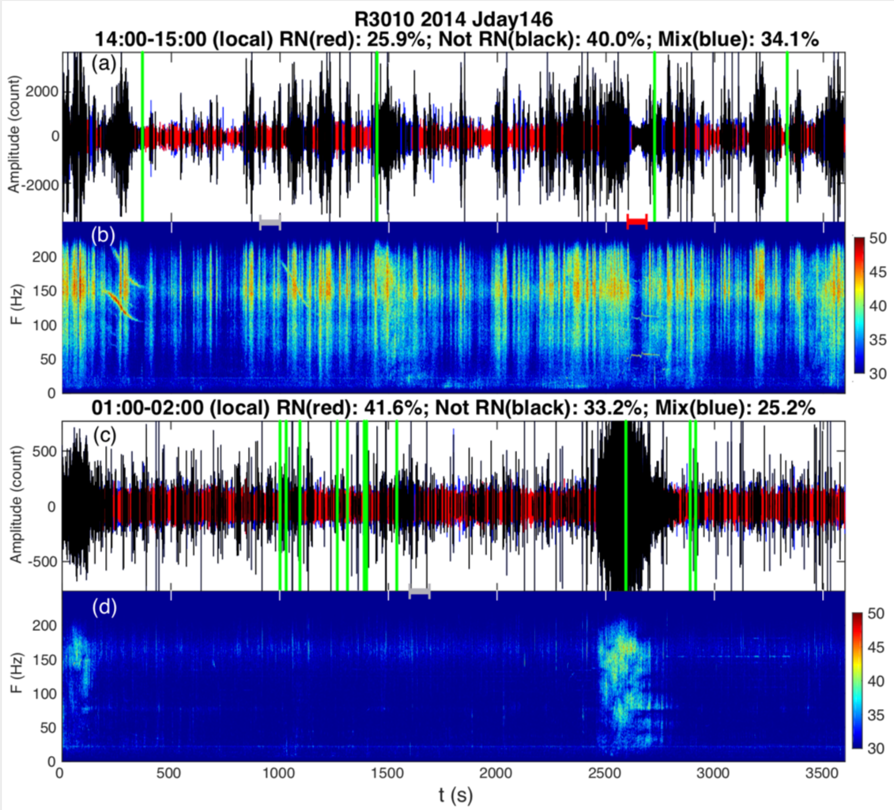
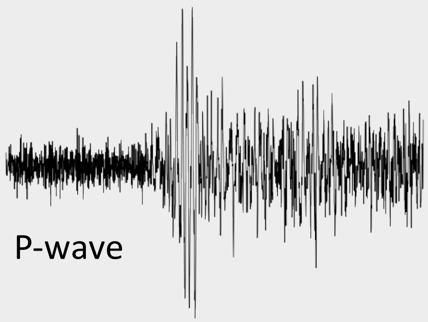


Image from Meng, Ben-Zion, and Johnson, in GJI revisions

Deep learning model – Assemble data

- Obtain earthquake arrival times
  - Extract 4s waveforms 1s before p-wave arrival
  - Vary start time within ±0.75s before p-wave
  - Use each event 5x to retain equal weight with noise
  - Filter 5-30 Hz, require SNR > 1.5
  - Obtain ~480,000 p-wave examples
  - Incorporates spatial variability across array
- Precalculate 2 noise labels
  - Use 4s of continuous labels
- Data set contains ~1.2 million labeled wavelets
  - Each API has input format
  - Shuffle data – Data must contain variability in subsets

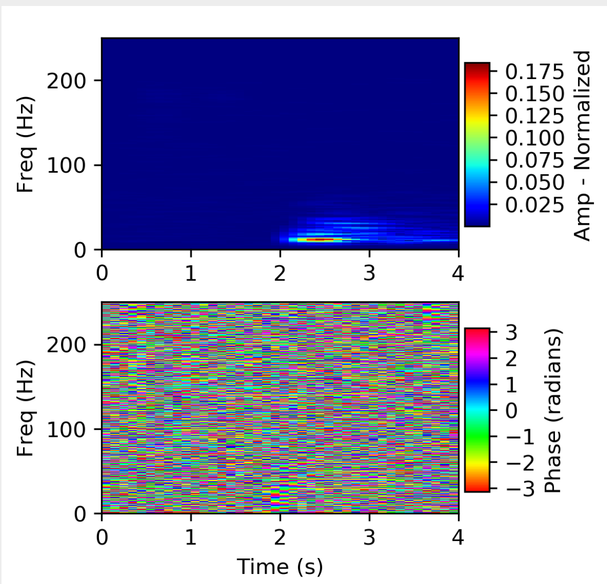


P-wave

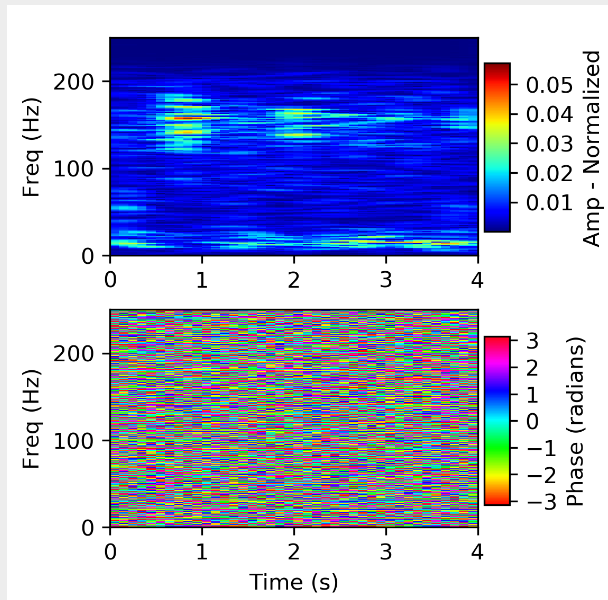
Noise

Deep learning model - Labels

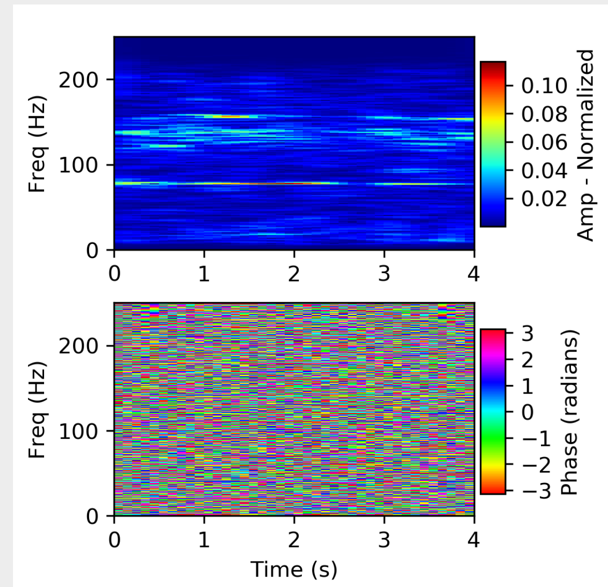
- Earthquake



- Random noise



- Not random noise



- Start with 3 labels
  - Equal number in each class
  - It is possible that non-random noise contains earthquakes
- STFT
  - Normalize waveform
  - Retain amp & phase
  - 2 layer input matrix

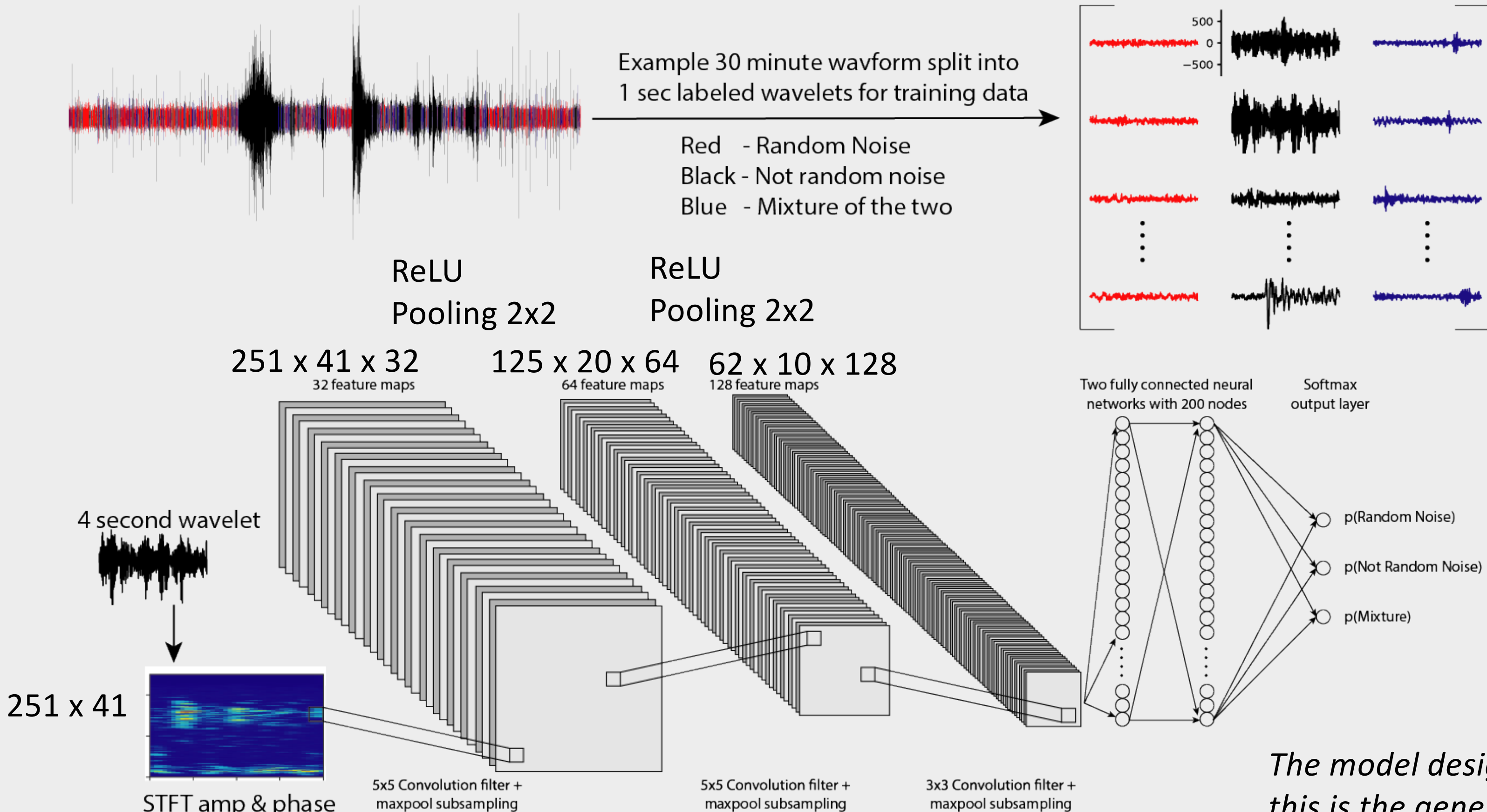
Research Approach and Objectives

- Build Convolutional Neural Network
  - Filter size, # layers, activation func (ReLU),
  - Pooling, batch normalization
  - FCN, softmax
- Get the model working before fine tuning
  - Hyperparameters
    - Learning rate
      - Good start is 0.01; Adjust up/down by an order of magnitude
      - Test decay
        - Slow the learning rate with each epoch
  - Test model design
    - Improve model by systematically adjusting
      - If too many things change at once, which one helps / hurts
    - Batch size
      - 32-256 is a good start

Software

- SKlearn
  - Data preprocessing
    - Train, Validate, Test
    - Shuffle
  - Model performance
    - Classification report
- Keras / Tensorflow
  - Keras uses Tensorflow backend
    - Great place to start learning
- Pytorch
  - Use if familiar with Python and CNN
  - Model is a class
    - Many examples exist

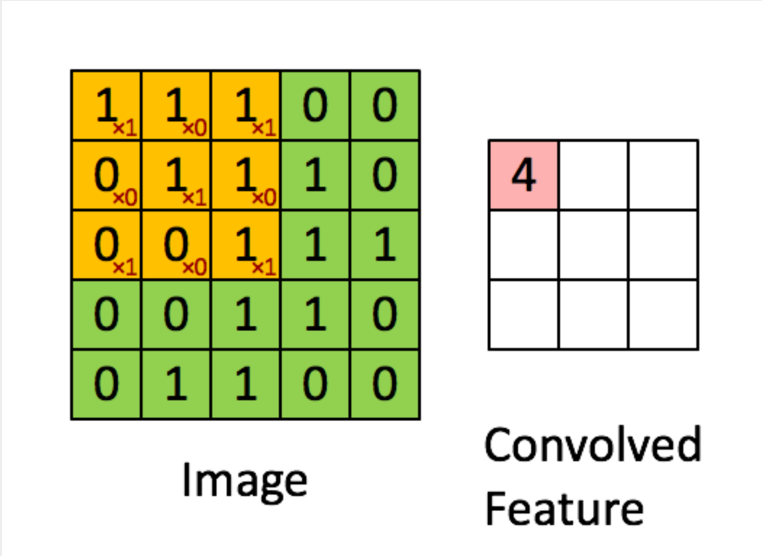
Convolutional Neural Network



The model design varies but this is the general setup

Convolutional Neural Network

- Convolutional
  - Scan matrix by translating a mask or template and taking inner product
  - Each mask contains filter weights
  - Add bias to convolution output
  - Repeat for set number of output layers all using different weights
- Weights and biases are the only parameters
  - Number of parameters increases to the millions if using multiple hidden layers

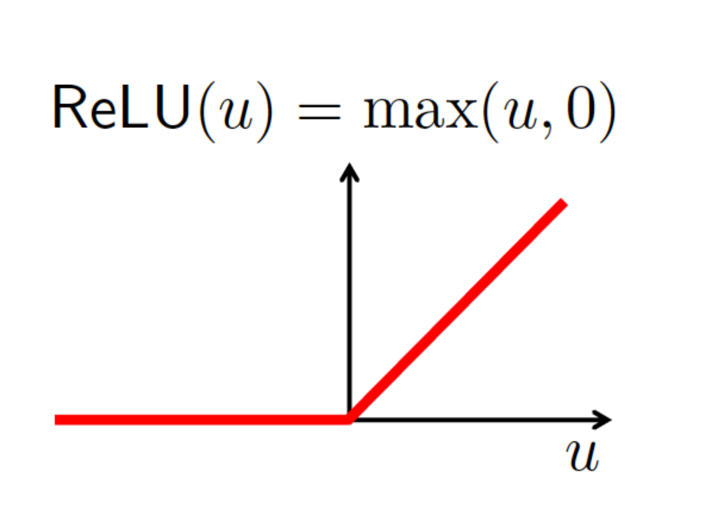


from http://deeplearning.stanford.edu/

Convolutional Neural Network

- Rectifier
  - Rectified linear unit (ReLU)
  - Remove negative values
  - Otherwise the problem is linear

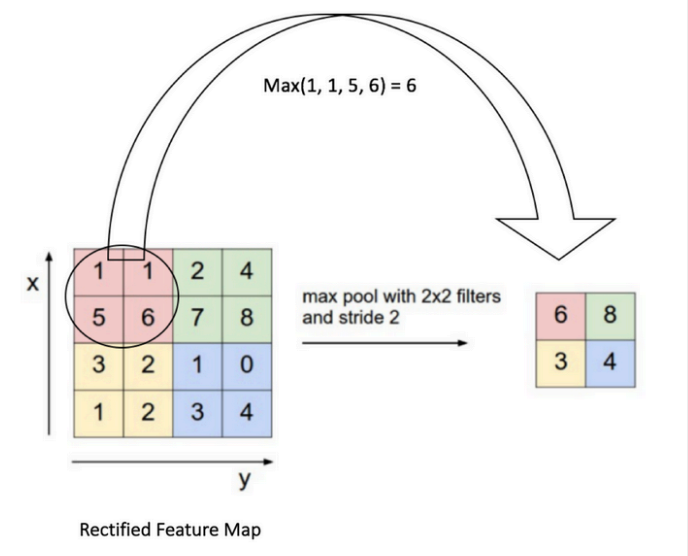
  - Can also try
    - tanh, Leaky ReLU, etc



from algorithmia.com

Convolutional Neural Network

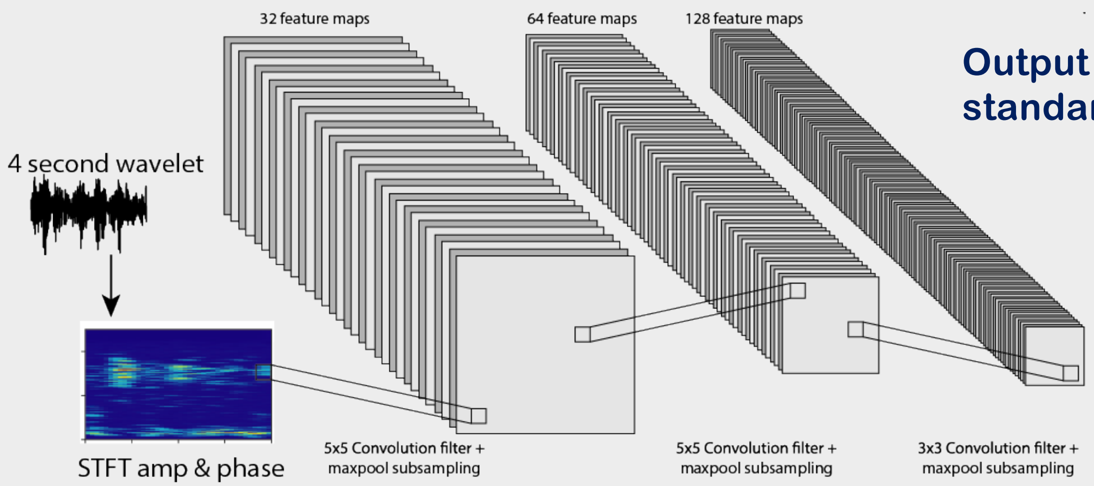
- Pooling
  - Down sample
  - Reduce dimensionality of subsequent layers
  - Common techniques
    - Max pooling (non-linear)
    - Avg. pooling (linear)
- After each pooling the filter kernel is ‘zoomed out’ from the input matrix



from algorithmia.com

Convolutional Neural Network

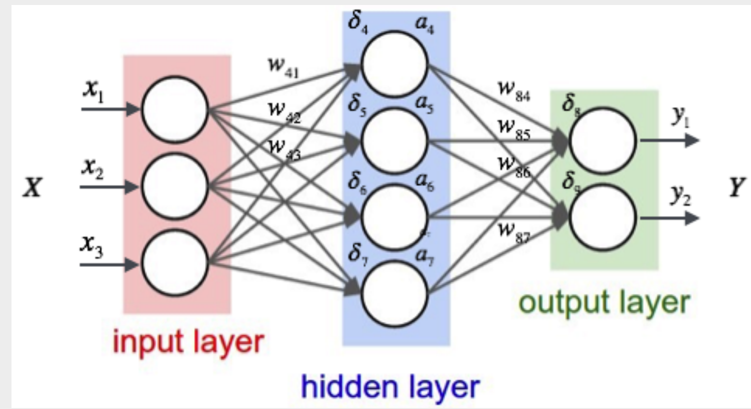
- **Advanced feature extraction technique**
- **Each layer has many filters detecting various features**



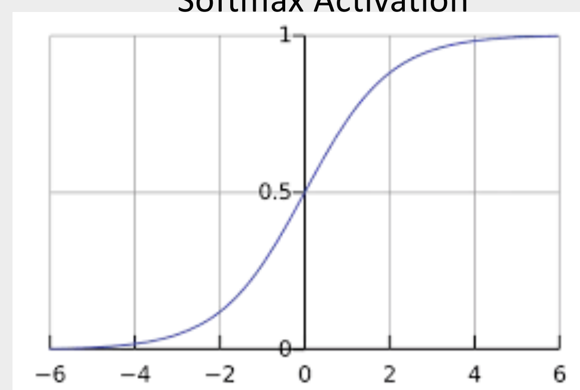
Output ConvNet features to a standard neural network

Convolutional Neural Network

- Designed to learn complex neural decision path
  - Hidden layers with ReLU activation
    - Weights are trainable parameters
- Output final layer to softmax activation function
  - sum(output layer) = 1
  - Probability estimate for final layer
- Stochastic gradient descent
  - Adam optimization
    - Variable learning rate
- ConvNet models require >50k LABELED training examples; even more for very complex problems



Softmax Activation



How is that actually done?

```
# Very simple Keras with Tensorflow backend example
model = Sequential()
# First filter
model.add(Conv2D(64, (5, 5), activation='relu', padding='same', input_shape=(n, o, p)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
# Second filter
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
# Convolution operators are multi-dimension matrix. Flatten to array
model.add(Flatten())
# Send extracted features from convolutions to fully connected Neural Network
model.add(Dense(1024, activation='relu'))
model.add(BatchNormalization())
# Hidden layer
model.add(Dense(1024, activation='relu'))
model.add(BatchNormalization())
# Output layer with softmax activation
model.add(Dense(3, activation='softmax'))
```

Model performance (on test data!!)

- Type I Error (precision)
  - Quantify false positive
  - Prediction correct
  - $\frac{True\ Positive}{True\ Positive + False\ Positive}$
- Type II Error (recall)
  - Quantify false negative
  - Prediction misclassifies
  - $\frac{True\ Positive}{True\ Positive + False\ Negative}$
- F1-score
  - Good = low FP and low FN
  - Bad = high FP and high FN
  - Perfect == 1
  - Failure == 0

TP = True Positives
TN = True Negatives
FP = False Positives
FN = False Negatives

	p' (Predicted)	n' (Predicted)
p (Actual)	True Positive	False Negative
n (Actual)	False Positive	True Negative

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

$$\textbf{F1-score} = \frac{2 * precision * recall}{precision + recall}$$

Deep learning model - Training

- Model training w/ ~930,000 2-layer spectral amp and phase
  - ~1 hour training time
- Validation and test
  - Good precision on earthquakes
  - Mislabeled noise data is expected
  - Random noise and non-random noise shows 80-88% precision
  - Non-random will contain some earthquakes producing

Training metrics

Validation Set # 168587

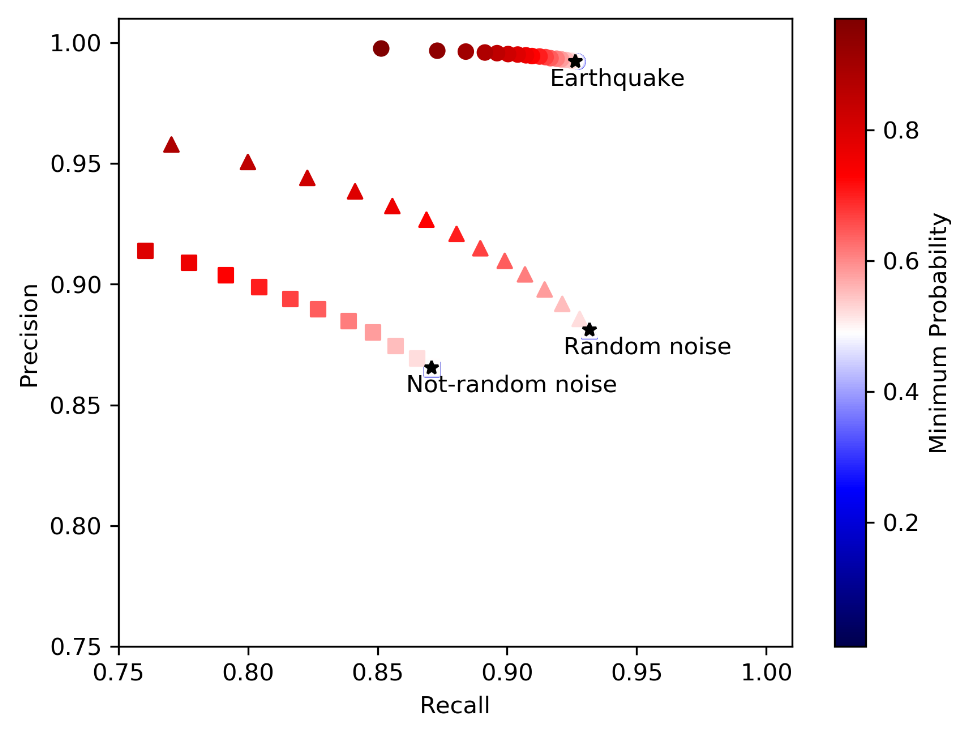
	precision	recall	f1-score	support
EQ	0.99	0.93	0.96	56107
RN	0.88	0.93	0.91	56298
NRN	0.86	0.87	0.87	56182
weighted avg	0.91	0.91	0.91	168587

Test Set # 50000

	precision	recall	f1-score	support
EQ	0.98	0.85	0.91	16799
RN	0.87	0.93	0.90	16677
NRN	0.80	0.86	0.83	16524
weighted avg	0.89	0.88	0.88	50000

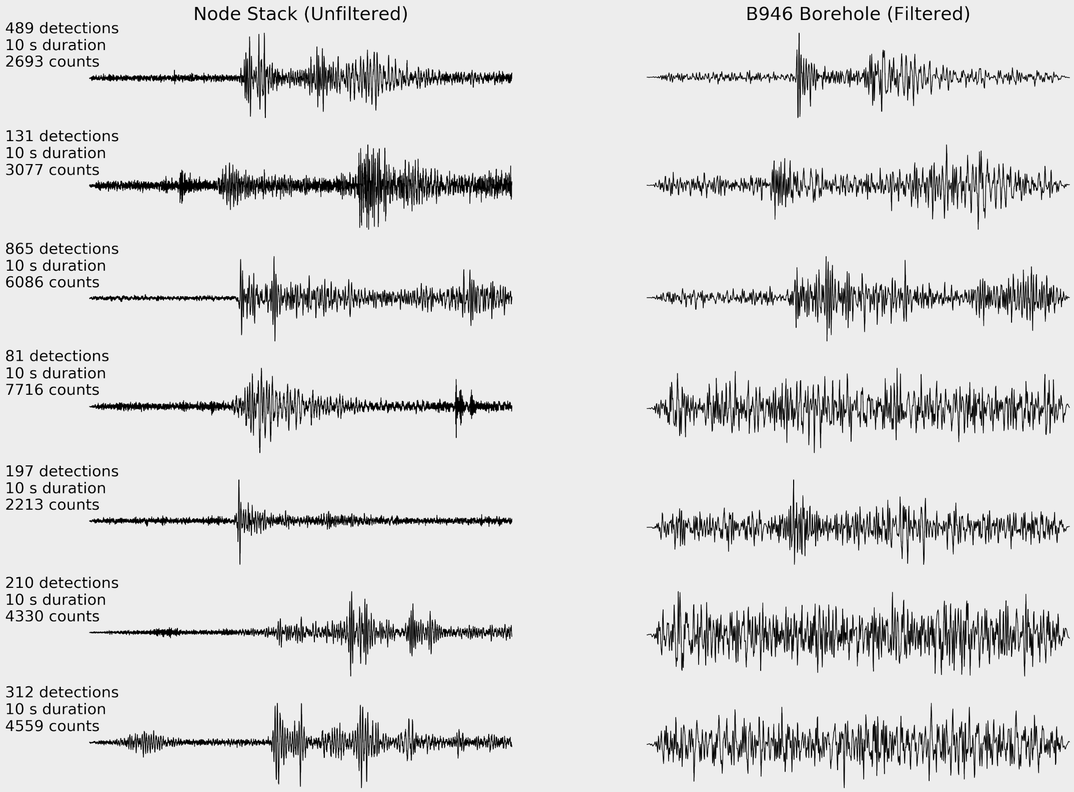
Deep learning model - Training

- Earthquakes
  - High precision ~99%
  - Recall ~93%
    - Not-random noise expected to have mislabeled input
- Random noise
  - Precision ~88%
  - Recall ~93%
- Non-random noise
  - Precision ~86%
  - Recall ~87%



Deep learning model – Eq Detections

- 1.5 minutes to classify 1 s interval for entire daily record
- Results for J-day 149
  - 19 catalog events
  - 64 CNN detections
  - 10 node minimum for detection
  - Node stack average
    - Time shifted to max cc
  - Borehole seismometer comparison
    - Filtered 5-30 Hz
- Similar results for all days processed
- Comparable to RF model but faster



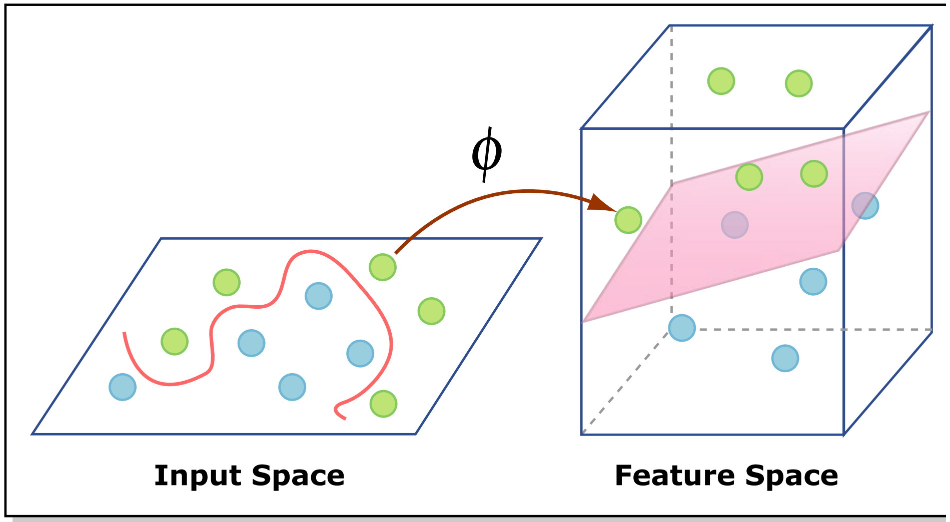
Remarks

- CNN can classify subtle variations in waveforms
  - Used spectrogram here
  - Time domain waveforms also will perform well if trained correctly
- Advantages
  - Trained model can classify waveforms more efficiently
  - Potential to discover new observations
- Other possible directions
  - Recurrent Neural Networks
    - Incorporate time information
  - Denoise with autoencoders

Kernels

We might want to consider something more complicated than a linear model:

Example 1: $[x^{(1)}, x^{(2)}] \rightarrow \mathbf{\Phi}\left([x^{(1)}, x^{(2)}]\right) = \left[x^{(1)2}, x^{(2)2}, x^{(1)}x^{(2)}\right]$



Information unchanged, but now we have a **linear** classifier on the transformed points.

With the kernel trick, we just need kernel $k(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{\Phi}(\boldsymbol{a})^T\, \boldsymbol{\Phi}(\boldsymbol{b})$

Lecture 10
Support Vector Machines

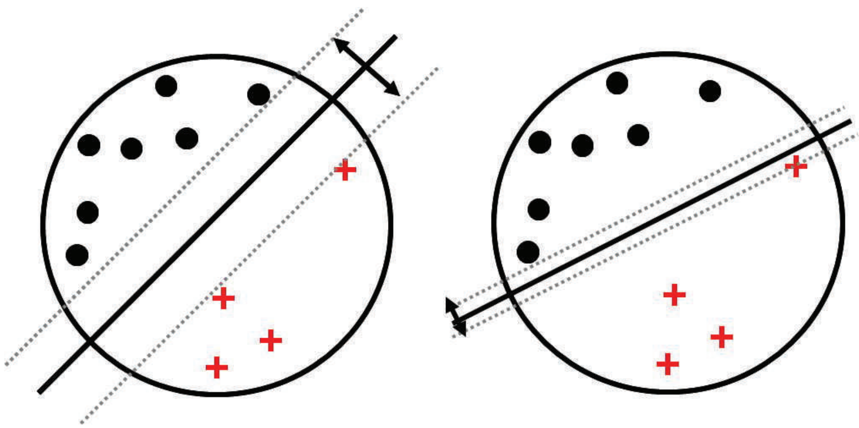
Non Bayesian!

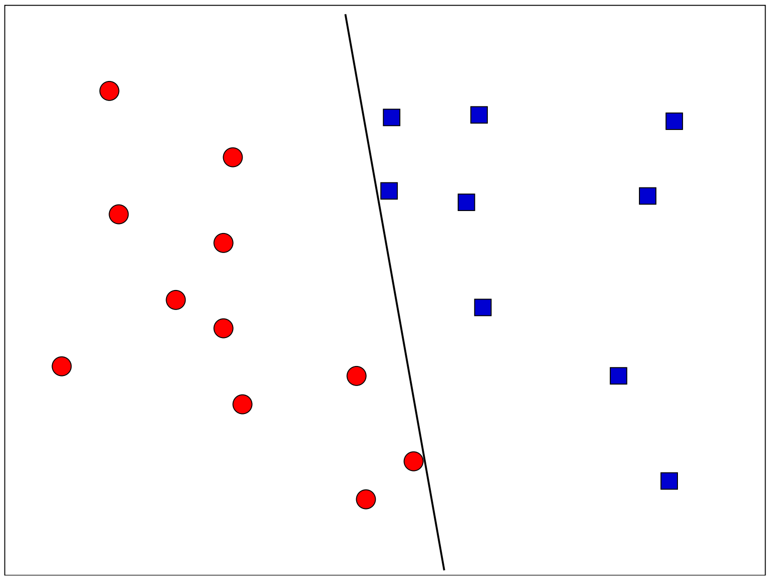
Features:

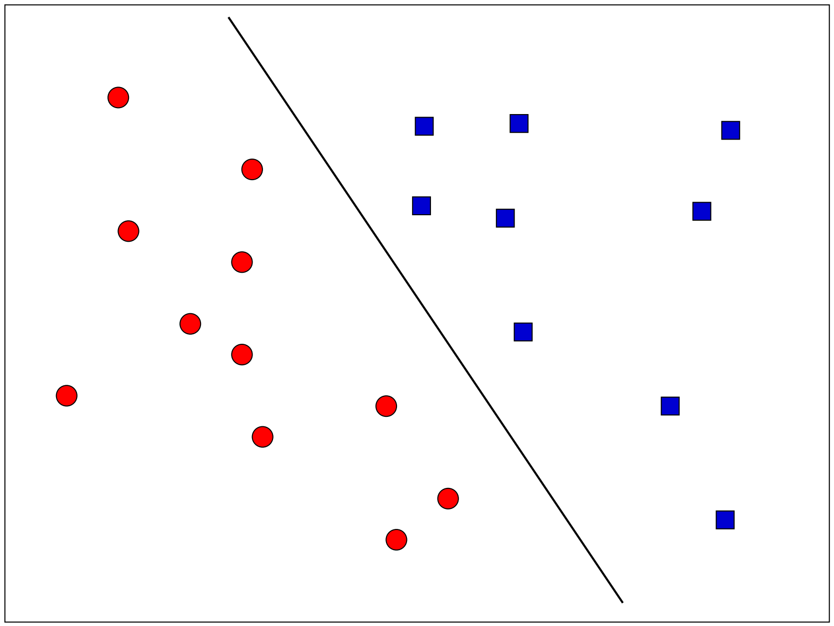
- Kernel
- Sparse representations
- Large margins

Regularize for plausibility

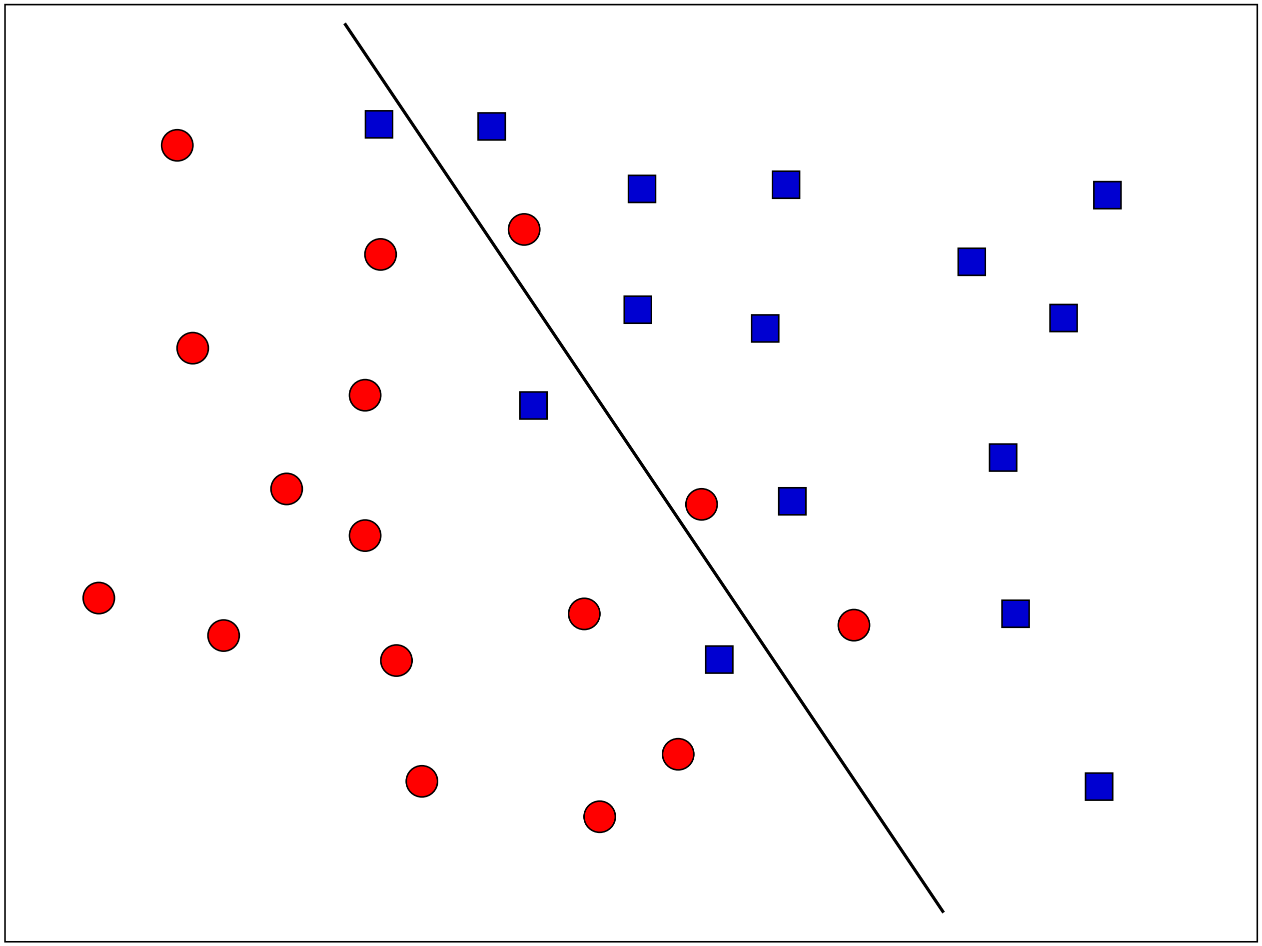
- Which one is best?
- We maximize the margin





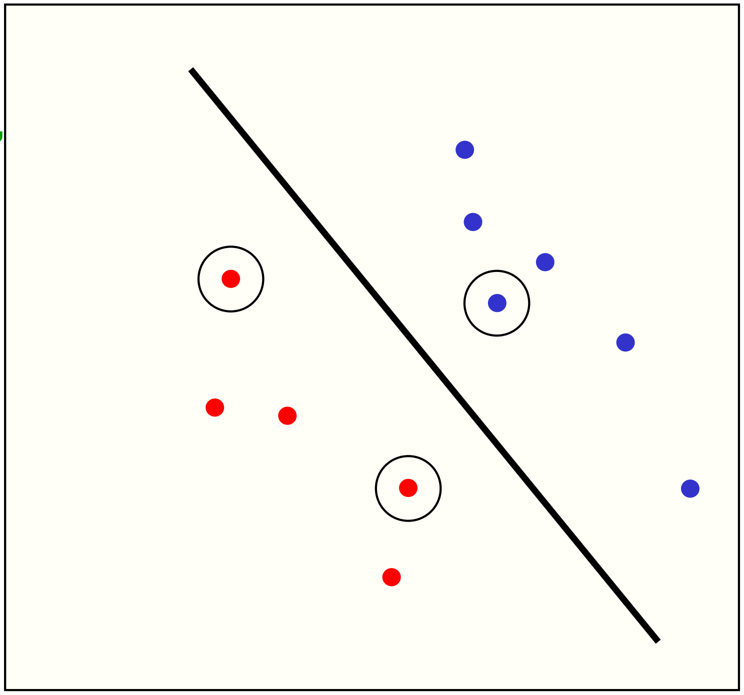


Regularize for plausibility



Support Vector Machines

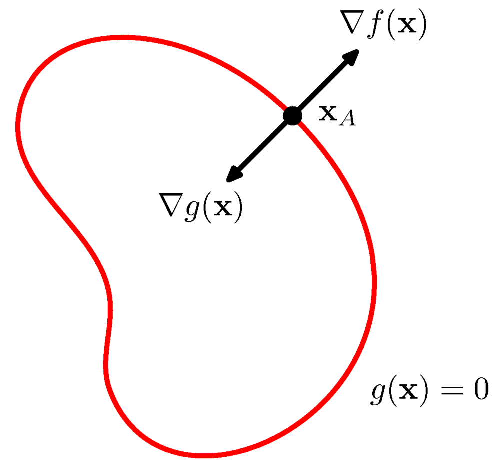
- The line that maximizes the minimum margin is a good bet.
  – The model class of "hyper-planes with a margin *m*" has a low VC dimension if *m* is big.
- This maximum-margin separator is determined by a subset of the datapoints.
  – Datapoints in this subset are called "support vectors".
  – It is useful computationally if only few datapoints are support vectors, because the support vectors decide which side of the separator a test case is on.



The support vectors are indicated by the circles around them.

Lagrange multiplier (Bishop App E)

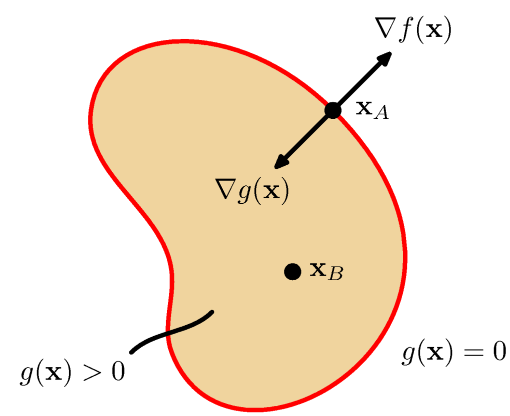
$$\max\big(f(x)\big) \text{ subject to } g(x) = 0$$



Taylor expansion

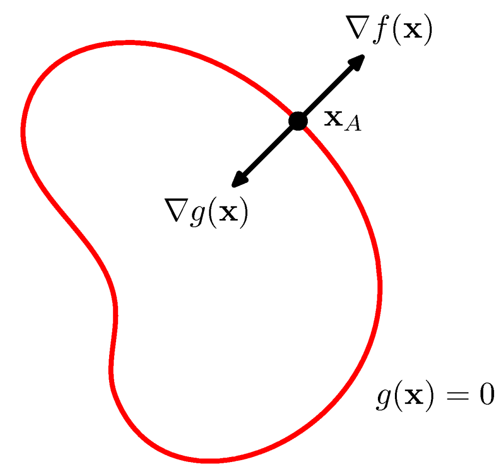
$$g(\boldsymbol{x} + \boldsymbol{\varepsilon}) = g(\boldsymbol{x}) + \boldsymbol{\epsilon}^T \nabla\, g(\boldsymbol{x})$$

$$L(x, \lambda) = f(x) + \lambda g(x)$$



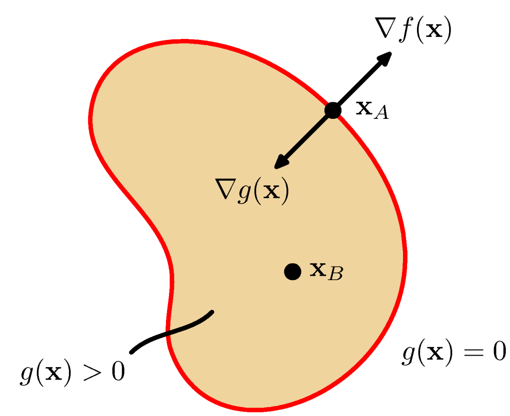
Lagrange multiplier (Bishop App E)

$$\max(f(\boldsymbol{x})) \text{ subject to } g(\boldsymbol{x}) > 0$$
$$L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda g(\boldsymbol{x})$$



Either ∇ f$(\boldsymbol{x}) = \mathbf{0}$
Then $g(\boldsymbol{x})$ is inactive, λ=0

$Or\ g(\boldsymbol{x}) = 0\ but\ \lambda > 0$



Thus optimizing $L(\boldsymbol{x}, \lambda)$ with the Karesh-Kuhn-Trucker (KKT) equations

$$g(\boldsymbol{x}) \geq 0$$
$$\lambda \geq 0$$
$$\lambda g(\boldsymbol{x}) = 0$$

Testing a linear SVM

- The separator is defined as the set of points for which:

$$\mathbf{w}.\mathbf{x} + b = 0$$

$$so\ if\ \ \mathbf{w}.\mathbf{x}^c + b > 0\ \ say\ its\ a\ positive\ case$$

$$and\ if\ \ \mathbf{w}.\mathbf{x}^c + b < 0\ \ say\ its\ a\ negative\ case$$