What drives success in ML?

Arcane knowledge of dozens of obscure algorithms?

Mountains of data?

Knowing how to apply 3-4 standard techniques?

Neural nets 101

Activation function: $h_i = f(z_i)$

"Weight vector" "Filter"
Example: Street View Address Number Transcription

(Goodfellow et al, 2014)
Three Step Process

• Use needs to define metric-based goals
• Build an end-to-end system
• Data-driven refinement
Identify Needs

• High accuracy or low accuracy?

• Surgery robot: high accuracy

• Celebrity look-a-like app: low accuracy
Choose Metrics

- Accuracy? (% of examples correct)
- Coverage? (% of examples processed)
- Precision? (% of detections that are right)
- Recall? (% of objects detected)
- Amount of error? (For regression problems)
End-to-end System

- Get up and running ASAP
- Build the simplest viable system first
- What baseline to start with though?
  - Copy state-of-the-art from related publication
Deep or Not?

- Lots of noise, little structure -> not deep
- Little noise, complex structure -> deep
- Good shallow baseline:
  - *Use what you know*
  - Logistic regression, SVM, boosted tree are all good
Choosing Architecture Family

- No structure $\rightarrow$ fully connected
- Spatial structure $\rightarrow$ convolutional
- Sequential structure $\rightarrow$ recurrent
Fully Connected Baseline

- 2-3 hidden layer feed-forward neural network
  - AKA “multilayer perceptron”
- Rectified linear units
- Batch normalization
- Adam
- Maybe dropout
Convolutional Network Baseline

- Download a pretrained network
- Or copy-paste an architecture from a related task
  - Or:
    - Deep residual network
    - Batch normalization
    - Adam
Recurrent Network Baseline

- LSTM
- SGD
- Gradient clipping
- High forget gate bias
Data-driven Adaptation

- Choose what to do based on data
- Don’t believe hype
- Measure train and test error
  - “Overfitting” versus “underfitting”
High Train Error

• Inspect data for defects

• Inspect software for bugs
  • Don’t roll your own unless you know what you’re doing

• Tune learning rate (and other optimization settings)

• Make model bigger
Checking Data for Defects

- Can a human process it?

26624
Increasing Depth

Effect of Depth

Test accuracy (%)

Number of hidden layers

(Goodfellow 2016)
High Test Error

• Add dataset augmentation
• Add dropout
• Collect more data
Increasing Training Set Size

![Graph showing error (MSE) and optimal capacity vs. number of train examples.](Goodfellow2016)
Tuning the Learning Rate

Figure 11.1: Typical relationship between the learning rate and the training error. Notice the sharp rise in error when the learning is above an optimal value. This is for a fixed training time, as a smaller learning rate may sometimes only slow down training by a factor proportional to the learning rate reduction. Generalization error can follow this curve or be complicated by regularization effects arising out of having a too large or too small learning rates, since poor optimization can, to some degree, reduce or prevent overfitting, and even points with equivalent training error can have different generalization error.
## Reasoning about Hyperparameters

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Increases capacity when. . .</th>
<th>Reason</th>
<th>Caveats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of hidden units</td>
<td>increased</td>
<td>Increasing the number of hidden units increases the representational capacity of the model.</td>
<td>Increasing the number of hidden units increases both the time and memory cost of essentially every operation on the model.</td>
</tr>
</tbody>
</table>
Hyperparameter Search

To perform grid search, we provide a set of values for each hyperparameter. The search algorithm runs training for every joint hyperparameter setting in the cross product of these sets.

To perform random search, we provide a probability distribution over joint hyperparameter configurations. Usually most of these hyperparameters are independent from each other. Common choices for the distribution over a single hyperparameter include uniform and log-uniform (to sample from a log-uniform distribution, take the exp of a sample from a uniform distribution). The search algorithm then randomly samples joint hyperparameter configurations and runs training with each of them. Both grid search and random search evaluate the validation set error and return the best configuration.

The figure illustrates the typical case where only some hyperparameters have a significant influence on the result. In this illustration, only the hyperparameter on the horizontal axis has a significant effect. Grid search wastes an amount of computation that is exponential in the number of non-influential hyperparameters, while random search tests a unique value of every influential hyperparameter on nearly every trial.

Figure 11.2

(Reproduced with permission from Bergstra and Bengio (2012)).