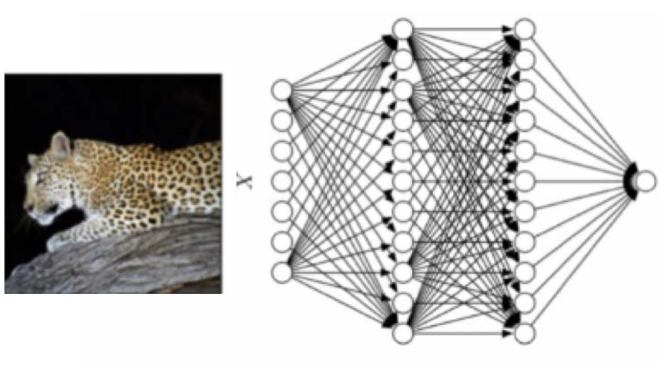
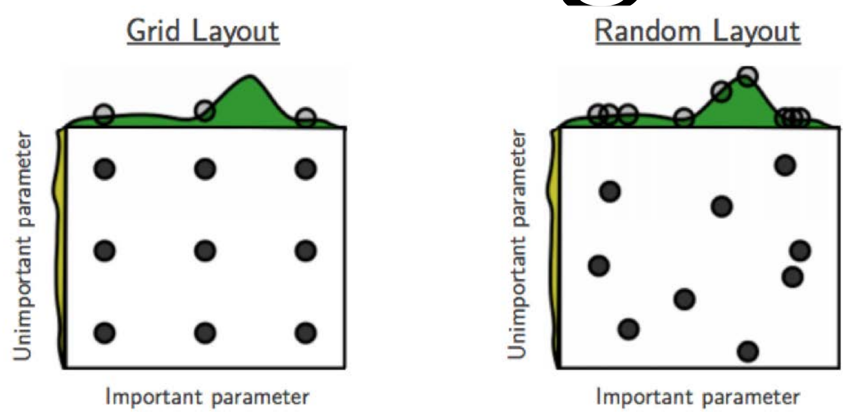
Tufts COMP 135: Introduction to Machine Learning

https://www.cs.tufts.edu/comp/135/2019s/

Hyperparameters and overfitting





Many slides attributable to:
Erik Sudderth (UCI), Emily Fox (UW),
Finale Doshi-Velez (Harvard)
James, Witten, Hastie, Tibshirani (ISL/ESL books)

Prof. Mike Hughes

Unit Objectives

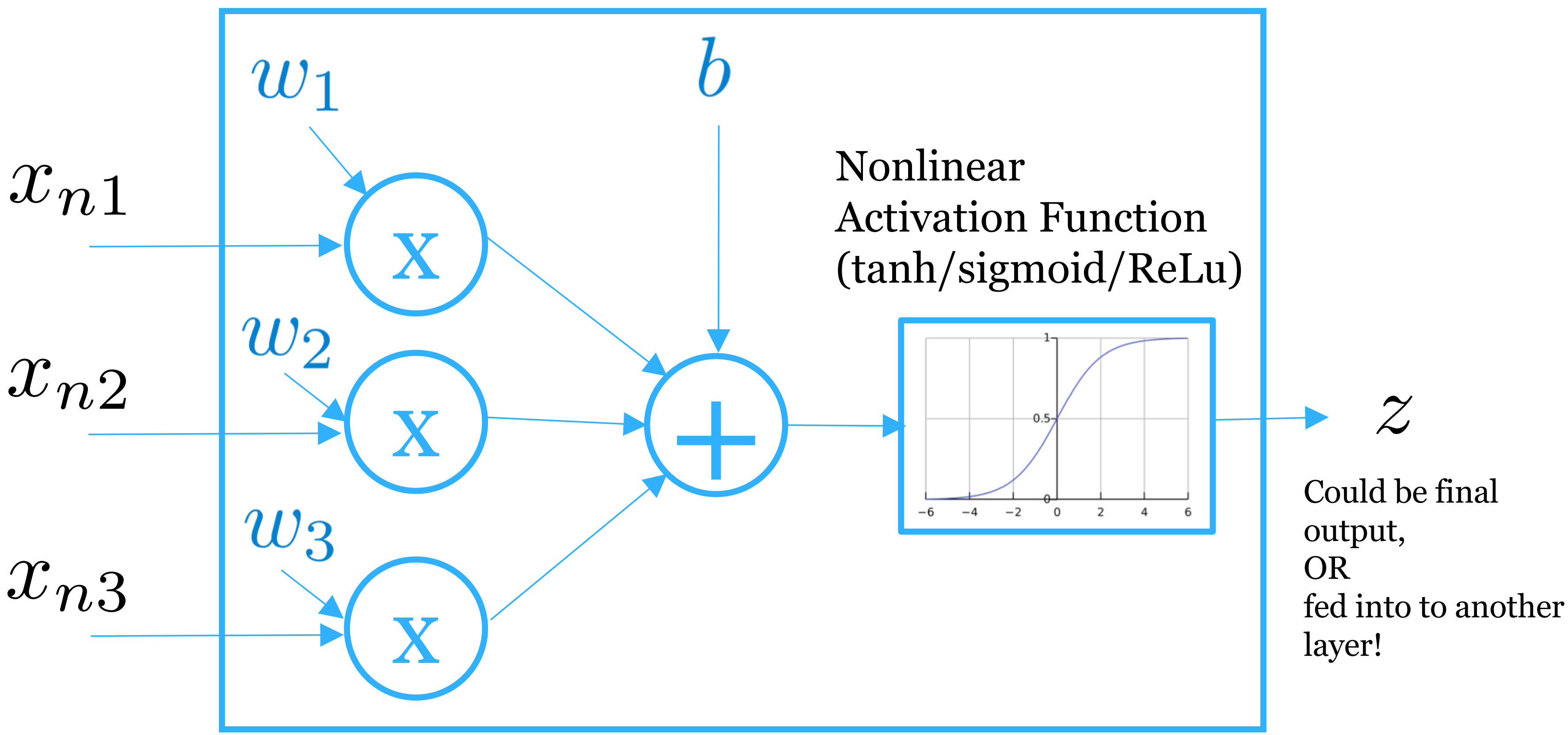
Summary of Deep Learning: pros and cons

Ways to improve heldout performance:
- Data Augmentation
- Early stopping
- Convolutions
- Dropout

Ways to select hyperparameters

Simple Many-to-one Neuron

The basic unit of neural networks for regression/classification



Deep Learning

Using neural networks to ***learn*** feature representations

Big ideas:

- Flexible Models from simple, easy-to-connect pieces
  - Simplest piece: linear weights + non-linear activation
  - Add layers! Add more units per layer!
- Focus on model, not algorithm
  - Use the same “universal” algorithm: back-propagation
  - Use automatic differentiation to compute gradients
- Scalability
  - Stochastic gradient descent for large datasets
  - GPUs make linear weights (matrix multiply) very fast

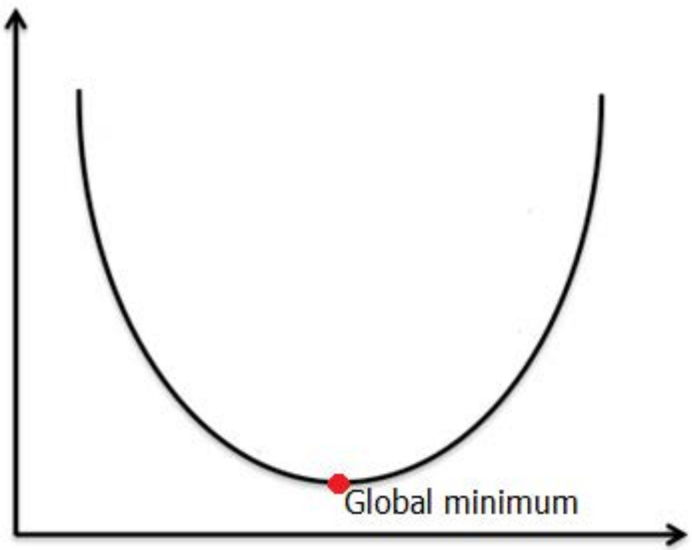
Deep Neural Nets

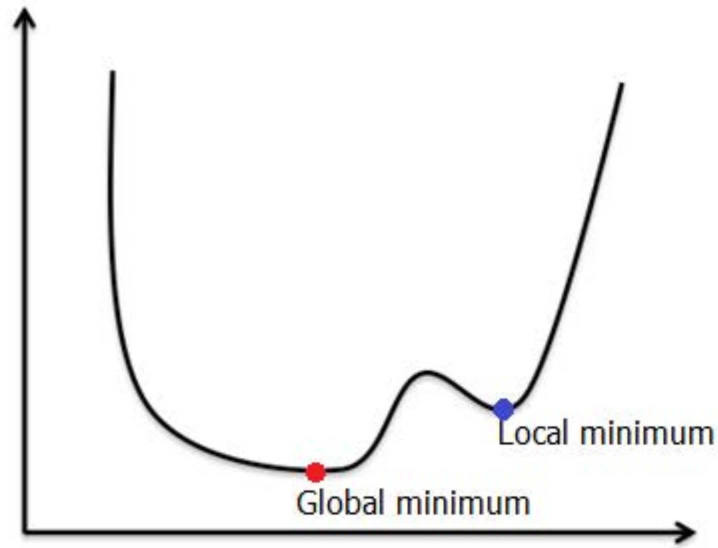
PROs

- Flexible models
- State-of-the-art success in many applications
  - Object recognition
  - Speech recognition
  - Language models
- Open-source software

CONs?

Two kinds of optimization problem





Convex

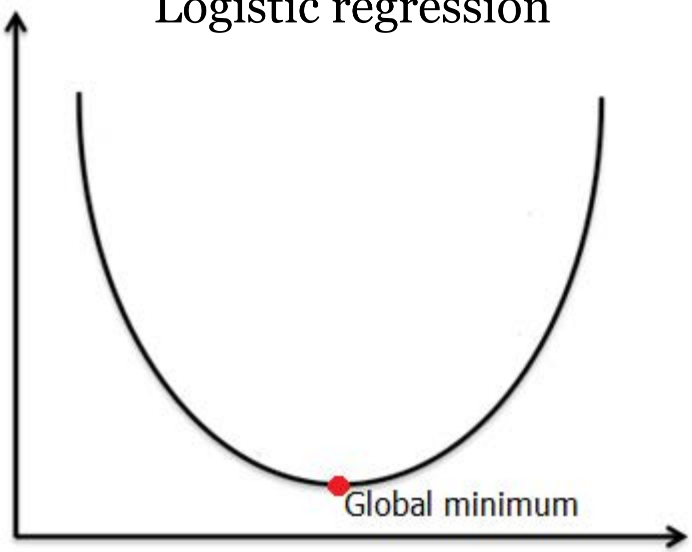
Only one global minimum
If GD converges, solution is best possible

Non-Convex

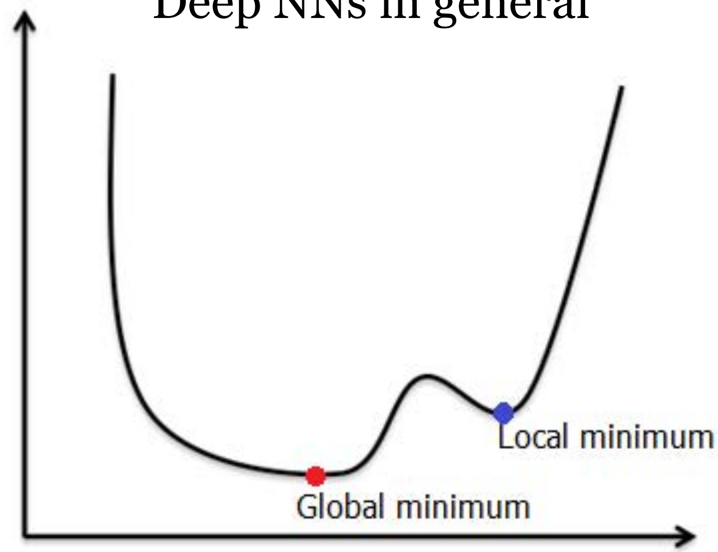
One or more local minimum
GD solution might be much worse than global minimum

Deep Neural Nets: Optimization is **not** convex

Linear regression
Logistic regression



MLPs with 1+ hidden layers
Deep NNs in general



Convex

Only one global minimum
If GD converges, solution is best possible

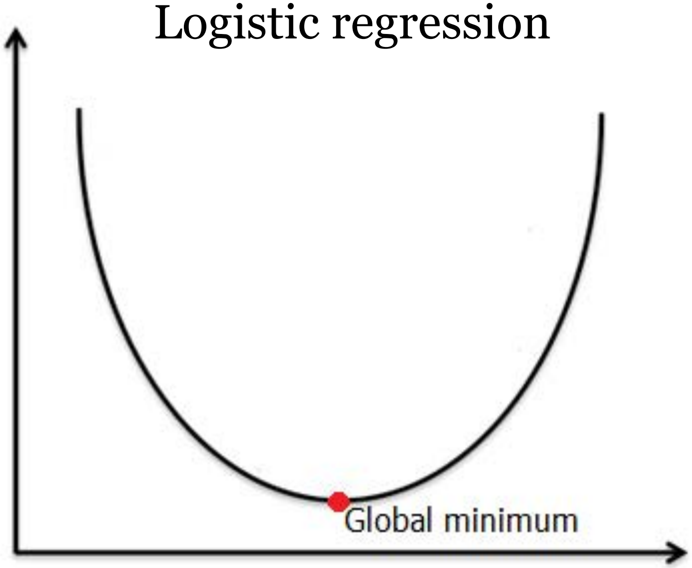
Non-Convex

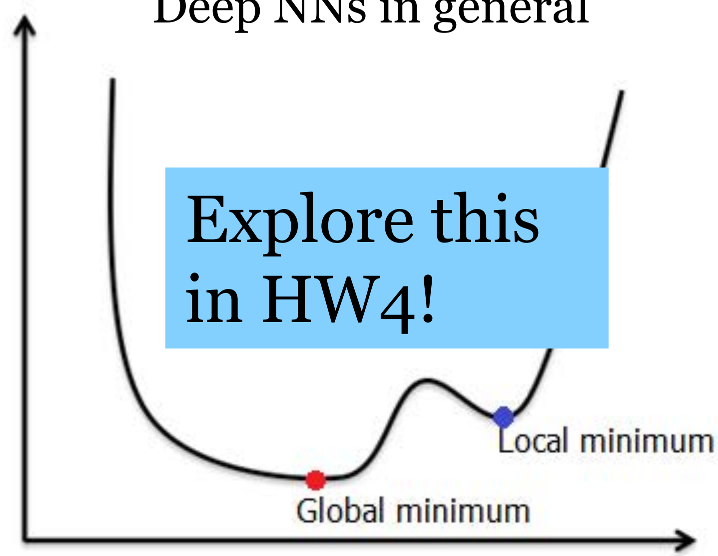
One or more local minimum
GD solution might be much worse than global minimum

Deep Neural Nets: Optimization is **not** convex

Linear regression
Logistic regression

MLPs with 1+ hidden layers
Deep NNs in general





Convex
Only one global minimum
If GD converges, solution is best possible

Non-Convex
One or more local minimum
GD solution might be much worse than global minimum

How many hyperparameters?

Many hyperparameters for a Deep Neural Network (MLP)

- Num. layers
- Num. units / layer
- Activation function
- L2 penalty strength

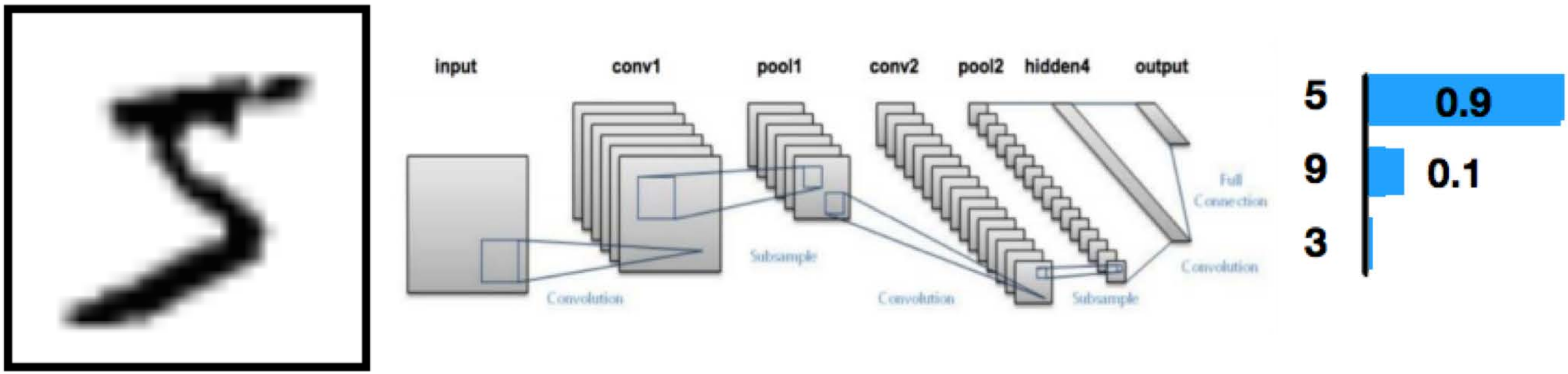
Control complexity

- Learning rate
- Batch size

Optimization quality/speed

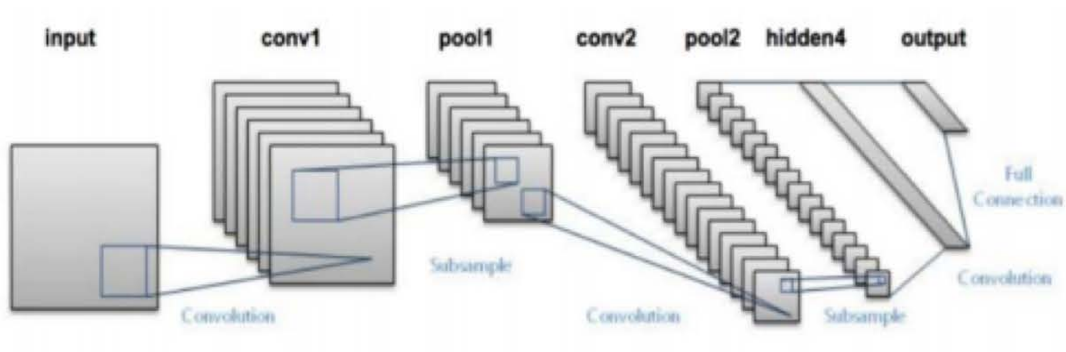
Will it generalize?

Familiar input: what we want



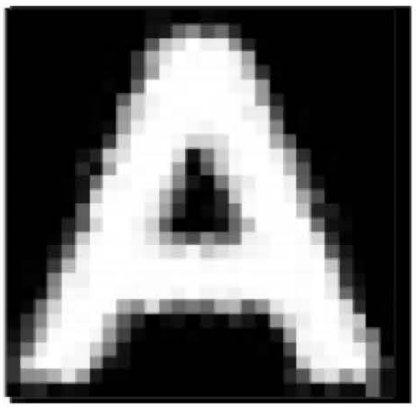
Network trained to recognize digits 0-9

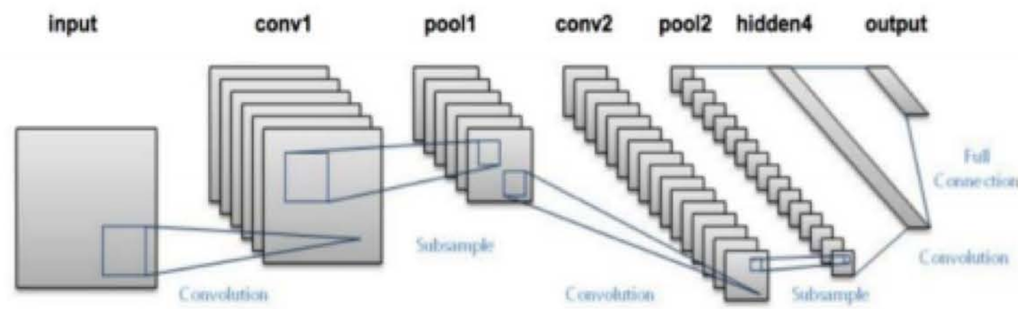
Unfamiliar input: what we want



Network trained to recognize digits 0-9

Unfamiliar input: typical result





Network trained to recognize digits 0-9

Deep Neural Nets

PROs

- Flexible models
- State-of-the-art success in many applications
  - Object recognition
  - Speech recognition
  - Language models
- Open-source software

CONs

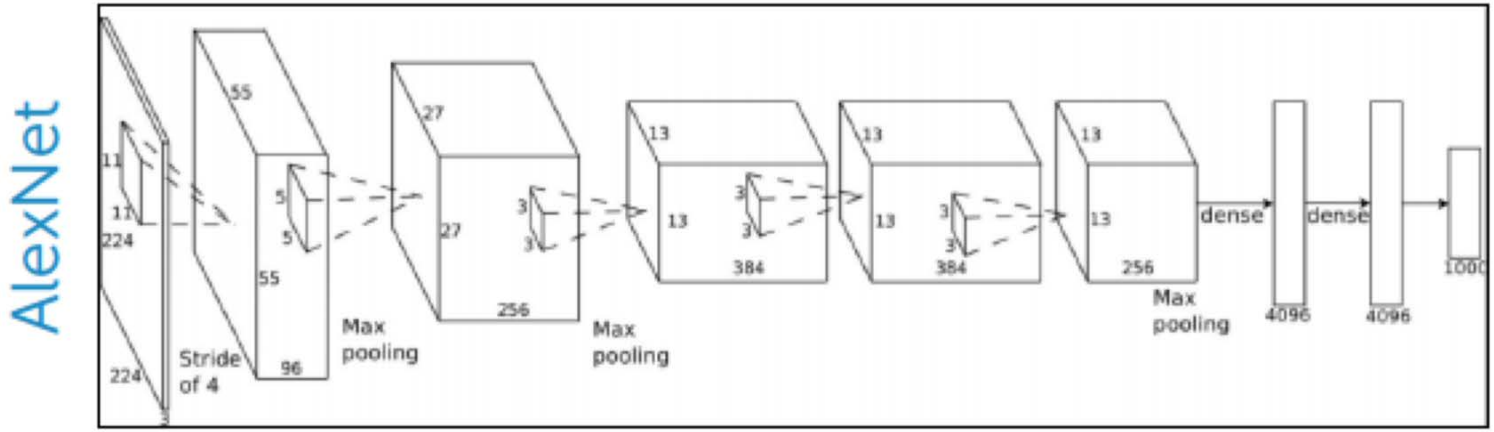
- Require lots of data
- Many tuning params
- Each run of SGD can take hours/days
- Optimization not easy
  - Will it converge?
  - Is local minimum enough?
- Hard to extrapolate
- Will it overfit?

Overfitting?

2012 ImageNet Challenge Winner

ImageNet challenge
1000 categories, 1.2 million images in training set

8 layers, 60M parameters [Krizhevsky et al. '12]

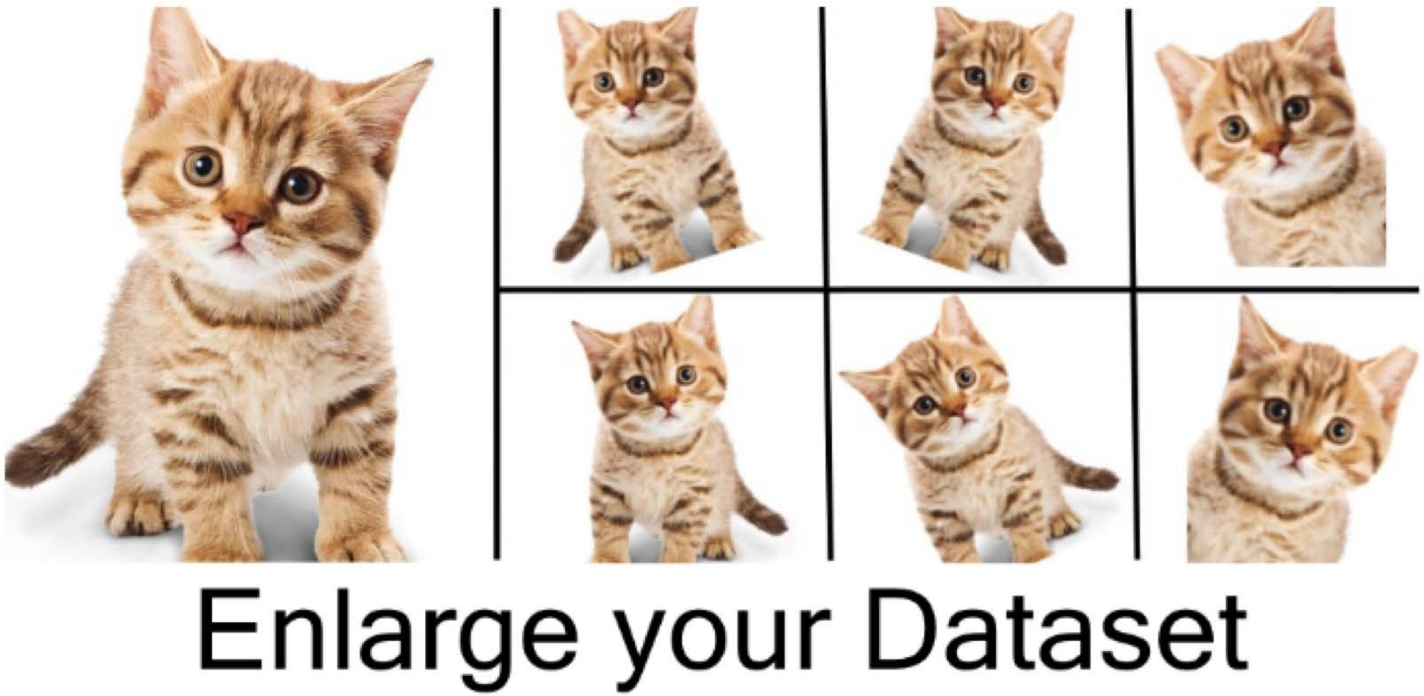


How to learn 60 million parameters from 1 million examples?

NN Tricks to avoid overfitting

- Gather more data
  - *Data augmentation*
- Modify optimization
  - *Early stopping*
- Reduce model complexity
  - *Convolutions*
  - *Dropout*

Data Augmentation: Gather more (artificial) data



Credit: Bharath Raj (medium.com post)

Data Augmentation

Data Augmentation: Increase effective size of training dataset by applying perturbations to existing features x to create new (x’, y) pairs

Choose perturbations which do not change label.

Images
- Flip left-to-right
- Slight rotations or crops
- Recolor or brighten

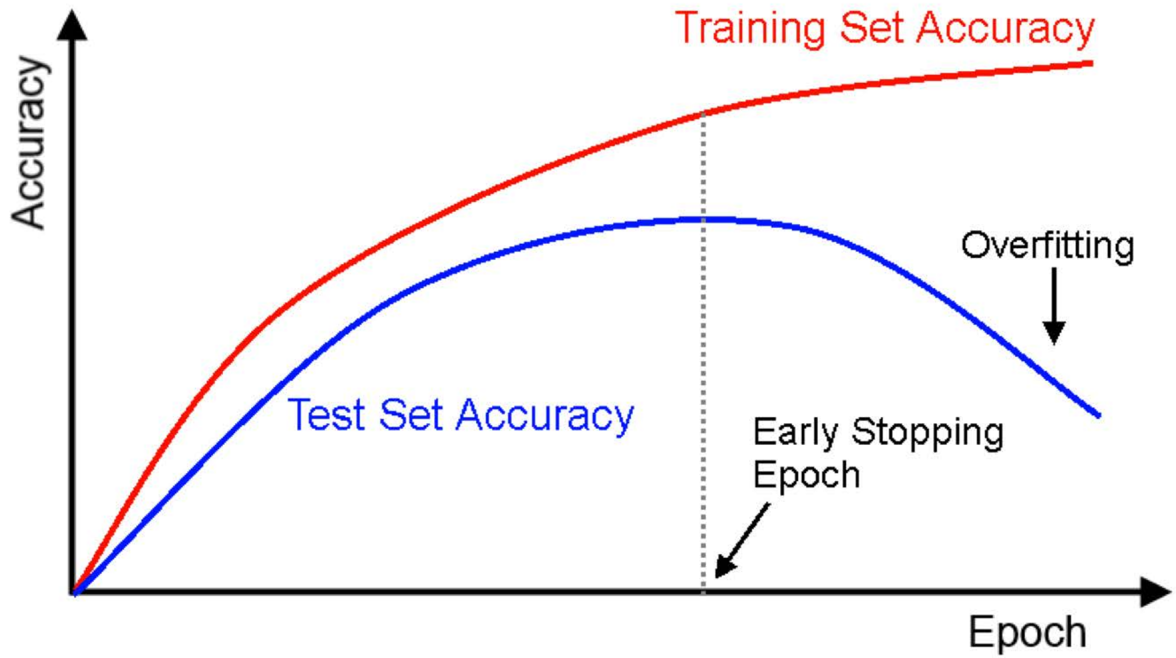
Text
- Add slight misspellings
- Replace word with similar word

This scheme approximately captures an important property of natural images, namely, that object identity is invariant to changes in the intensity and color of the illumination. This scheme reduces the top-1 error rate by over 1%.

from AlexNet paper (Krizhevsky et al. NIPS 2012)

Reduce overfitting by modifying optimization

Early Stopping



could be accuracy,
area under ROC,
Recall, precision,
whatever you care about

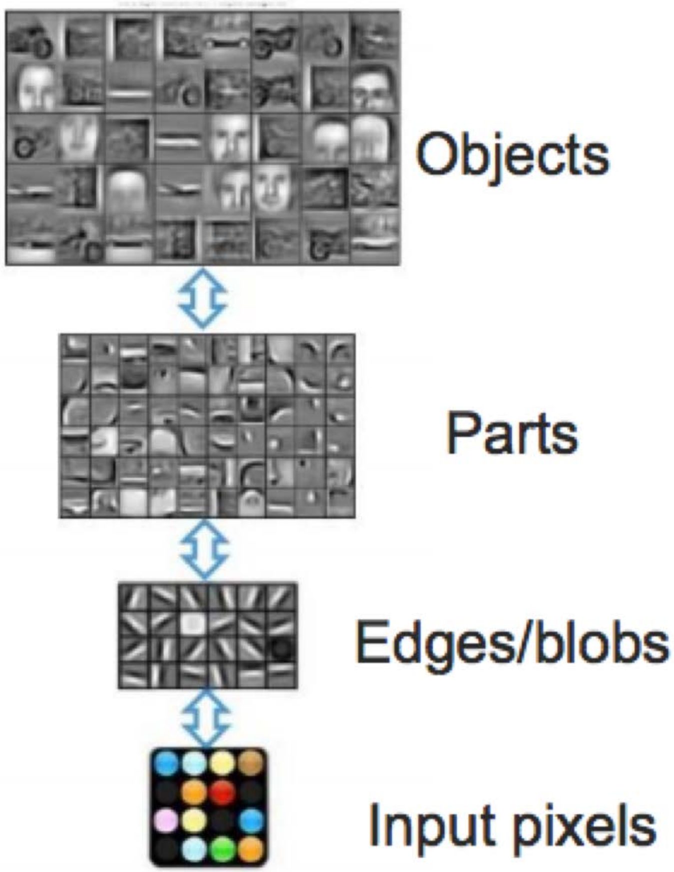
Big idea: stop training after your heldout set stops improving

- Avoid overfitting
- Save time / compute resources

Credit: https://deeplearning4j.org/docs/latest/deeplearning4j-nn-early-stopping

Reduce overfitting by reducing complexity (via domain-relevant architectures)

Convolutional Neural Networks (CNNs) for images

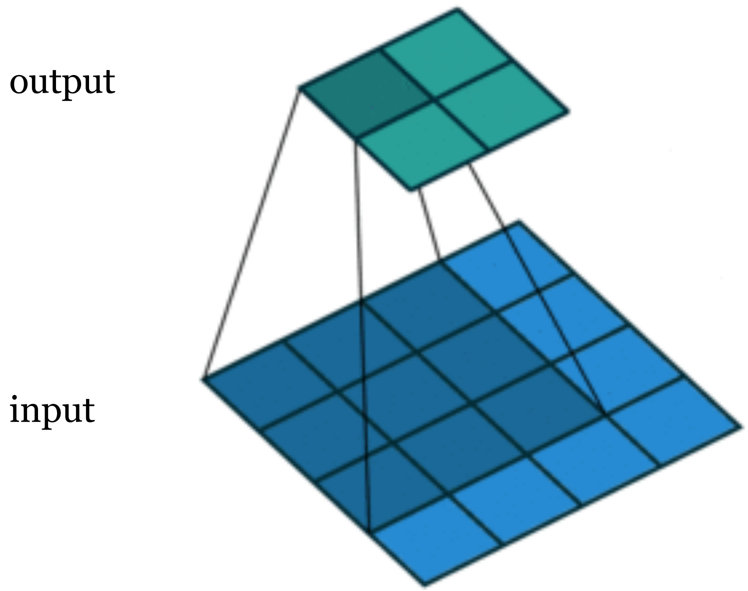


Goal: learn feature representations that:

- Represent high-level information
  - "objects" and "parts"
- Invariant to translation
  - object could appear anywhere

Credit: L.P. Morency & T. Baltrusaitis, ACL 2017 Tutorial
https://www.cs.cmu.edu/~morency/MMML-Tutorial-ACL2017.pdf

Basic 2D Convolution Operation

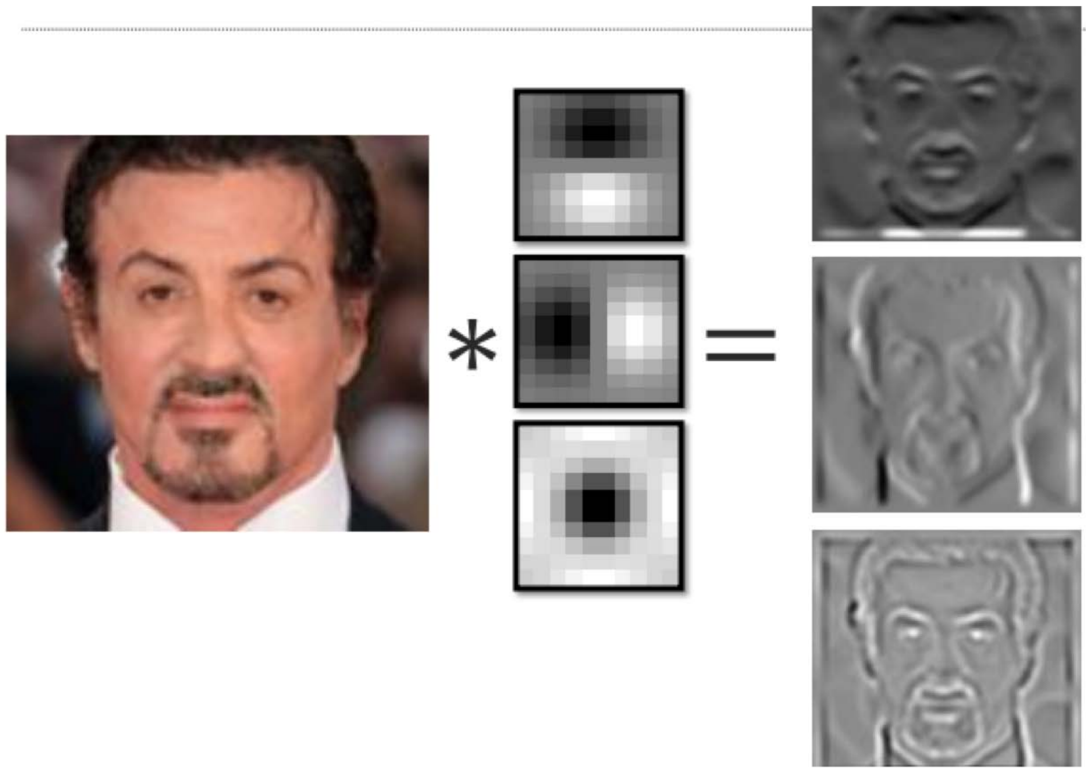


Slide same “small window” with fixed weights
across entire image

Each output value depends on **small subset** of input

Advantages
- Fewer parameters to learn
- Can detect same pattern in any position in the image

Example Convolution in 2D



Credit: L.P. Morency & T. Baltrusaitis, ACL 2017 Tutorial
https://www.cs.cmu.edu/~morency/MMML-Tutorial-ACL2017.pdf

Reduce overfitting by reducing complexity (via parameter dropout)

Existing complexity penalties

- L2 penalty
- L1 penalty

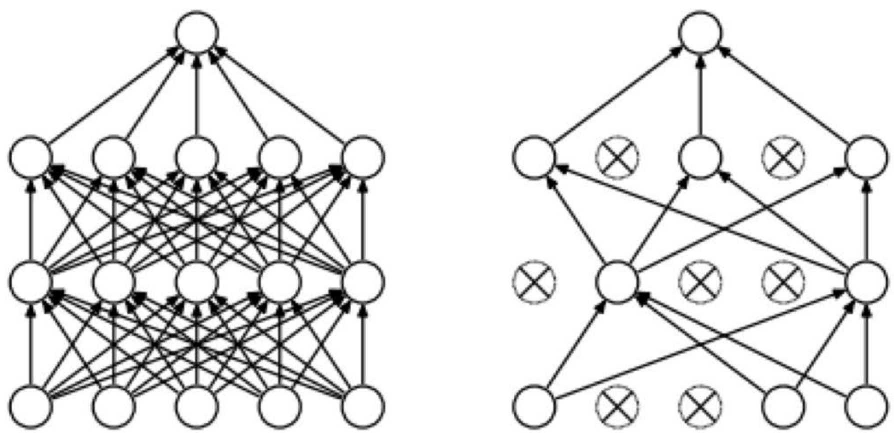
- Max norm

  After each update, enforce: $\sum_f w_f^2 < c$

Dropout

Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Nitish Srivastava NITISH@CS.TORONTO.EDU
Geoffrey Hinton HINTON@CS.TORONTO.EDU
Alex Krizhevsky KRIZ@CS.TORONTO.EDU
Ilya Sutskever ILYA@CS.TORONTO.EDU
Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU



(a) Standard Neural Net (b) After applying dropout.

Credit: Srivastava et al. JMLR 2014

Sample present/absent at train, downweight at test

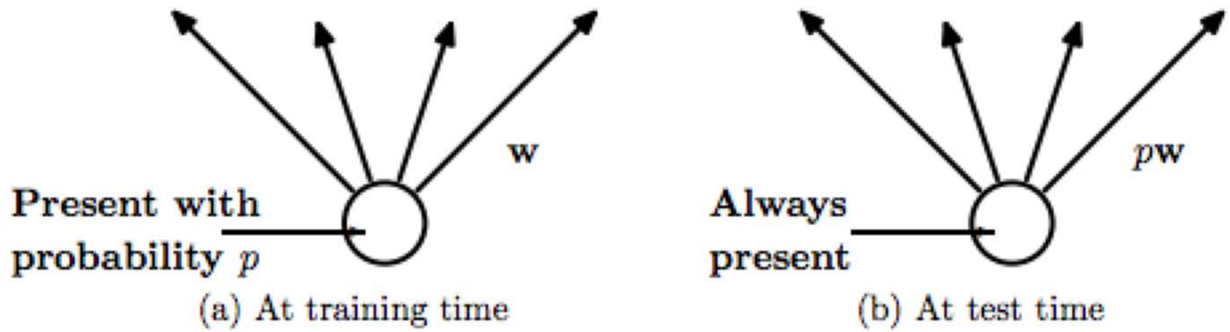


Figure 2: **Left**: A unit at training time that is present with probability p and is connected to units in the next layer with weights $\mathbf{w}$. **Right**: At test time, the unit is always present and the weights are multiplied by p. The output at test time is same as the expected output at training time.

In practice, often set dropout probabilities:

- 50% for hidden units
- 20% for input units

Credit: Srivastava et al. JMLR 2014

Dropout on MNIST

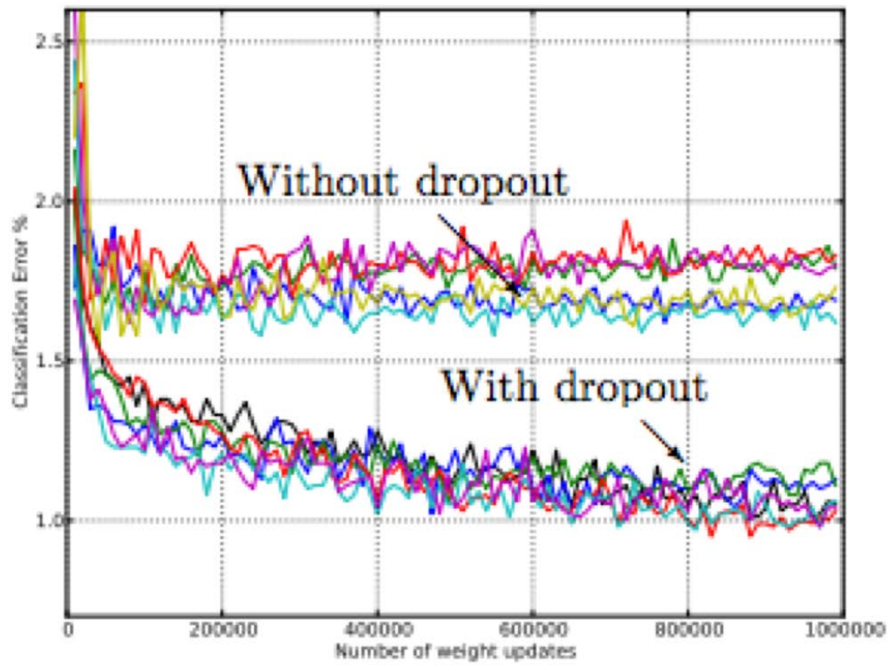


Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

Credit: Srivastava et al. JMLR 2014

Dropout Benefits

Decent gains on many tasks (images, genes, sequences)

- *over other regularization (L1/L2) and other models*

- MNIST images

Method	Test Classification error %
L2	1.62
L2 + L1 applied towards the end of training	1.60
L2 + KL-sparsity	1.55
Max-norm	1.35
Dropout + L2	1.25
Dropout + Max-norm	**1.05**

Table 9: Comparison of different regularization methods on MNIST.

lower
is better

Credit: Srivastava et al. JMLR 2014

Unit Objectives

Summary of Deep Learning: pros and cons

Ways to improve heldout performance:

- Data Augmentation
- Early stopping
- Convolutions
- Dropout

Ways to select hyperparameters

Hyperparameters for a Deep Neural Network (MLP)

- Num. layers
- Num. units / layer
- Activation function
- L2 penalty strength
- *Dropout probability*

Control complexity

- Learning rate
- Batch size

Optimization quality/speed

Guidelines: complexity params

- Num. units / layer
  - Start with similar to num. features
  - Add more (logspace) until serious overfitting
- Num. layers
  - Start with 1
  - Add more (+1 at a time) until serious overfitting
- L2 penalty strength scalar
  - Try range of logspace values
- Activation function
  - ReLU for most problems is reasonable

Grid Search

- List possible values of each hyperparameter

Step size/learning rate $\{0.1, 0.01, 10^{-3}, 10^{-4}, 10^{-5}\}$

Number of hidden units $\{50, 100, 200, 500, 1000, 2000\}$

- Try out all H1 x H2 x … x H5 combinations

Can yield impossibly large number of combinations
but, testing combinations can be parallelized

Random Search

- Define probability distribution over all hyperparameter configurations
  - Often, assume each parameter is independent
- Draw samples from joint configuration space
- Choose best of T samples

number of hidden units was drawn geometrically[3] from 18 to 1024.

sigmoidal or tanh nonlinearity with equal probability

learning rate ε_0 drawn geometrically from 0.001 to 10.0

Each trial can be parallelized

Random Search covers more of the important "lower dim. space"

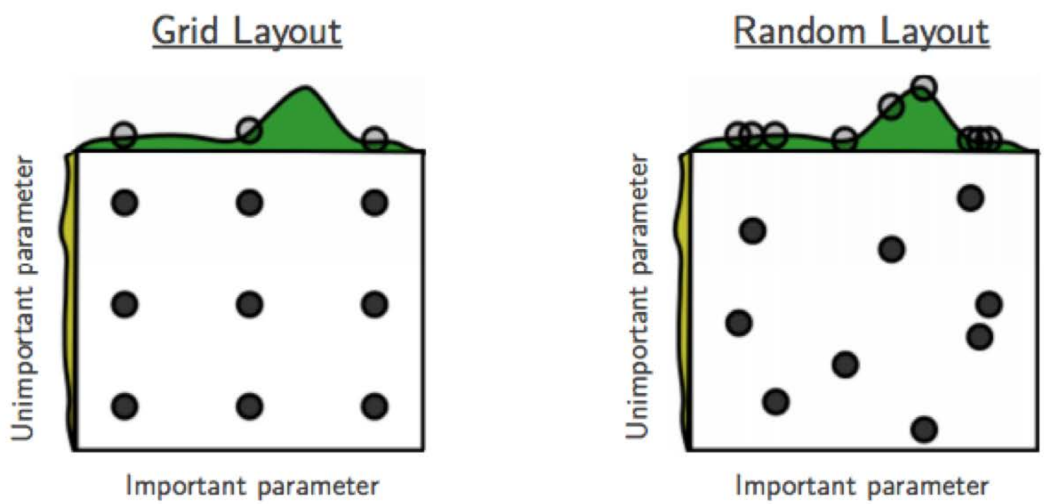
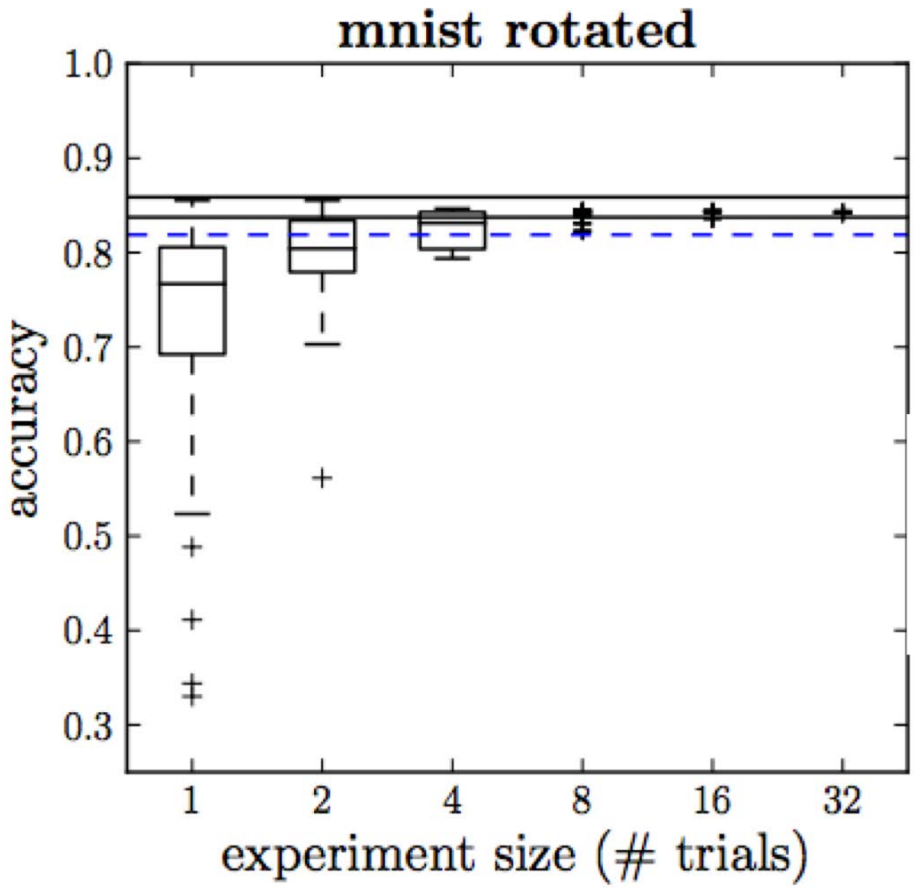


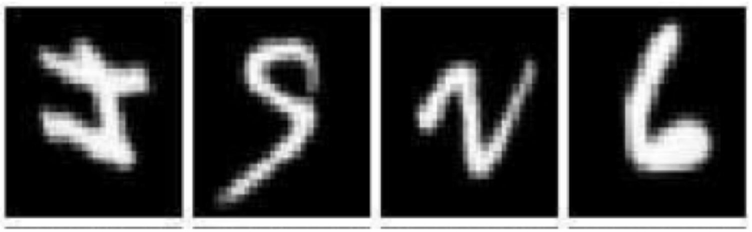
Figure 1: Grid and random search of nine trials for optimizing a function $f(x,y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct places. With random search, all nine trials explore distinct values of g. This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization.

Credit: Bergstra & Bengio JMLR 2012

8 random trials beats 100 grid search trials on MNIST digits



Grid search over 100 configs



Credit: Bergstra & Bengio JMLR 2012

Sequential Optimization

Fixed budget of T trials, search for best hyperparameters x based on a predictive model

SMBO(f, M_0, T, S)

$\mathcal{H} \leftarrow \emptyset$,
For $t \leftarrow 1$ **to** T,
    $x^* \leftarrow \operatorname{argmin}_x S(x, M_{t-1})$,
    Evaluate $f(x^*)$, ▷ *Expensive step*
    $\mathcal{H} \leftarrow \mathcal{H} \cup (x^*, f(x^*))$,
    Fit a new model M_t to $\mathcal{H}$.
return $\mathcal{H}$

Figure 1: The pseudo-code of generic Sequential Model-Based Optimization.

Credit: Bergstra et al. NeurIPS 2011
https://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf

Hyperopt Toolbox

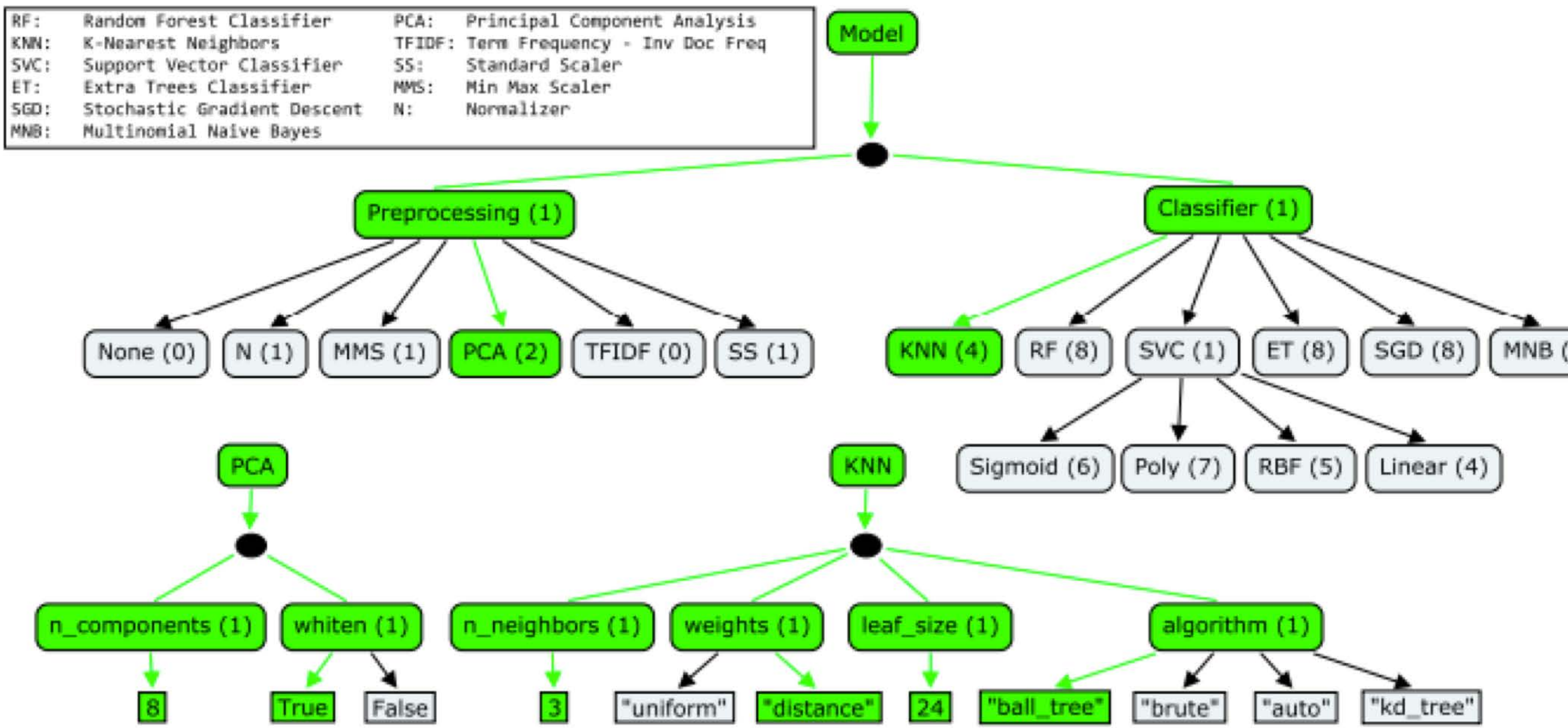
https://www.youtube.com/watch?v=Mp1xnPfE4PY

https://github.com/hyperopt/hyperopt/wiki/FMin

```
from hyperopt import fmin, tpe, rand, hp

def loss(x):
    return x**2

best_rand_search = fmin(fn=loss,
    space=hp.uniform('x', -10, 10),
    algo=rand.suggest,
    max_evals=100)

best_tpe_search = fmin(fn=loss,
    space=hp.uniform('x', -10, 10),
    algo=tpe.suggest,
    max_evals=100)
```

Hyperopt-Sklearn: Automatic Hyperparameter Configuration for Scikit-Learn

Brent Komer[‡*], James Bergstra[‡], Chris Eliasmith[‡]



PROJECT 2:
Text Sentiment Classification

Sentiment Analysis

- Question: How to represent text reviews?

Friendly staff, **good tacos**, and **fast service**. What more can you **look** for at **taco bell**?

$$\phi(x_n)?$$

Bag-of-words representation

original data

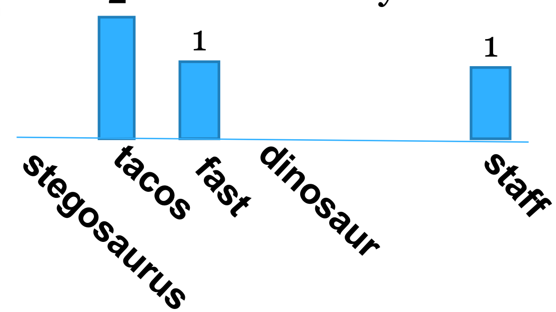
unordered "bag" of vocab symbols

count vector

over large (fixed-size) vocabulary

Text

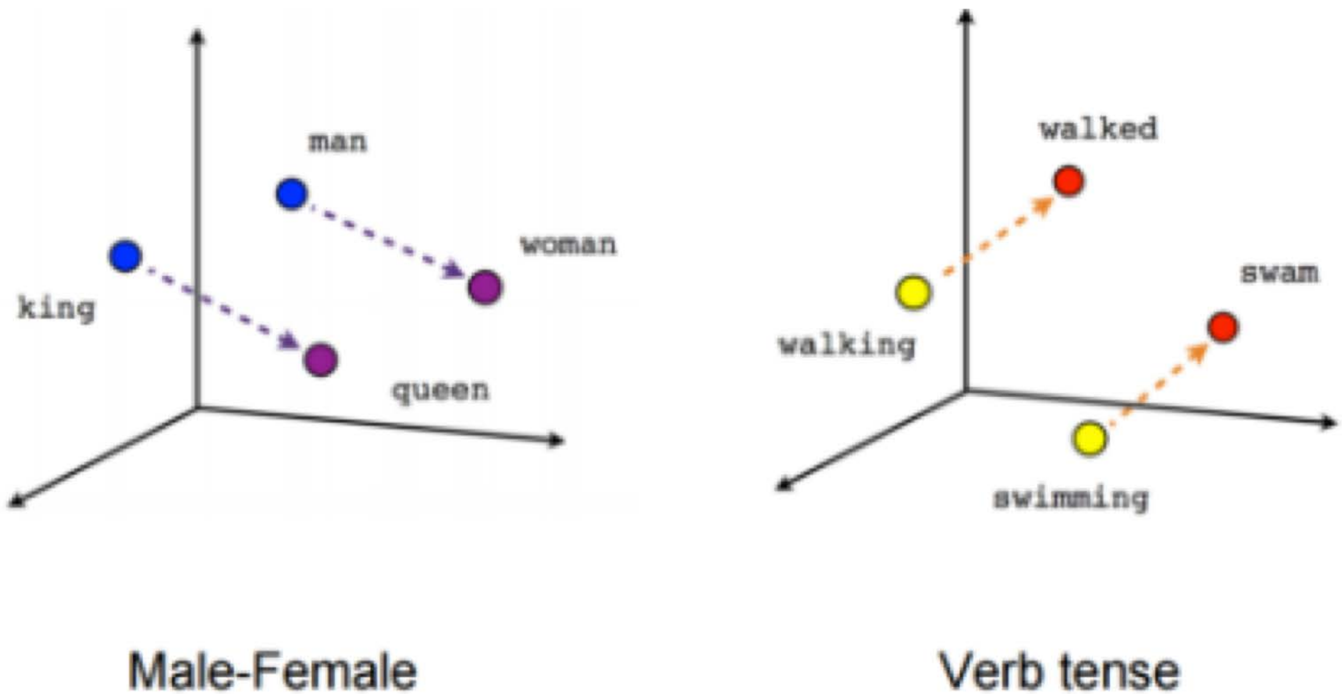
Friendly staff, **good tacos**, and **fast service**. What more can you **look** for at **taco bell**?



Word Embeddings (word2vec)

Goal: map each word in vocabulary to high-dimensional vector

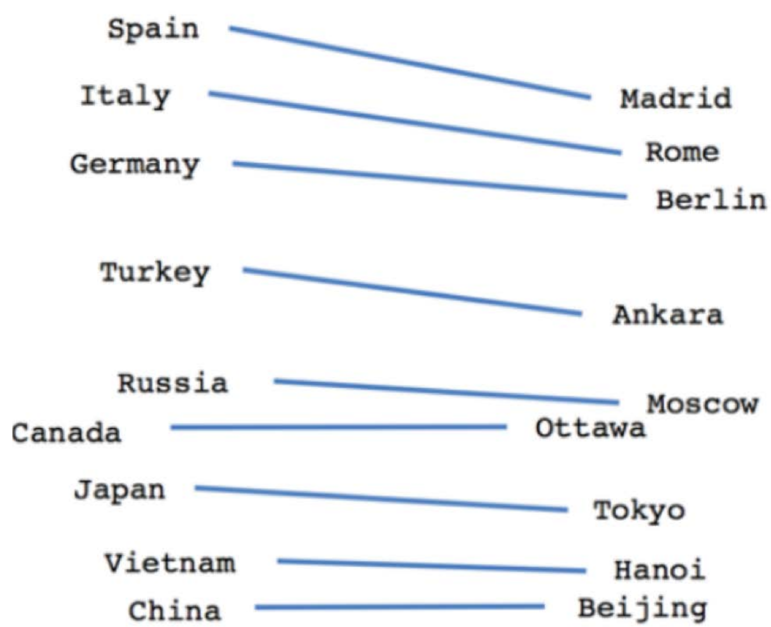
- Preserve semantic meaning in this new vector space



vec(swimming) – vec(swim) + vec(walk) = vec(walking)

Word Embeddings (word2vec)

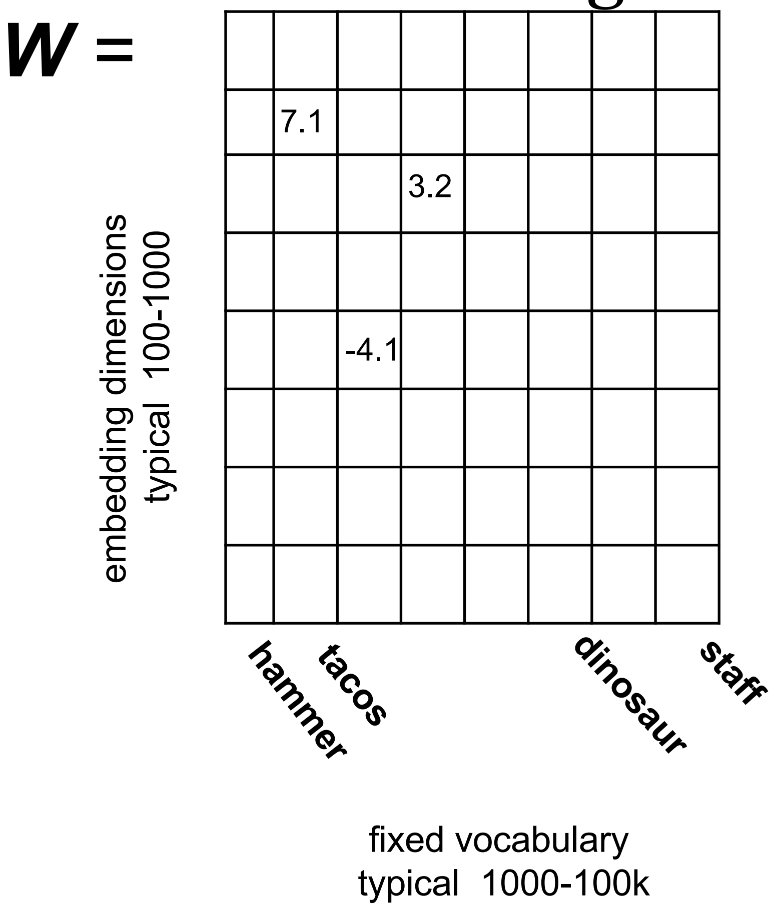
Goal: map each word in vocabulary to high-dimensional vector
- Preserve semantic meaning in this new vector space



Country-Capital

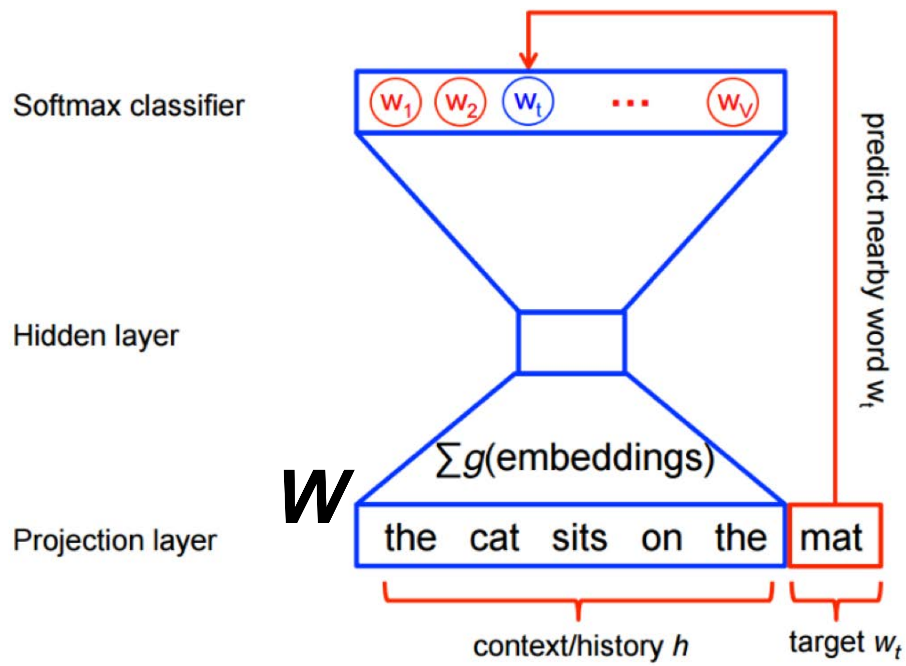
How to embed?

Goal: learn weights



Training

Reward embeddings that predict nearby words in the sentence.



Credit:
https://www.tensorflow.org/tutorials/representation/word2vec

PROJECT 2:
Text Sentiment Classification

What features are best?
What classifier is best?
What hyperparameters are best?