Announcements

- HW4 requires installing software.
- Poster session December 7

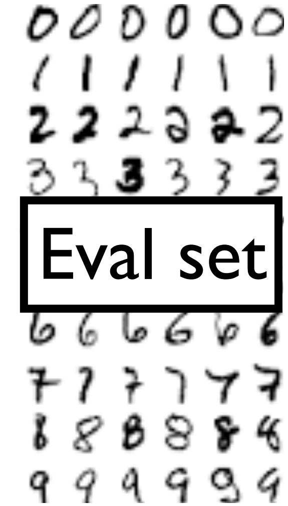
Hyperparameter Optimization

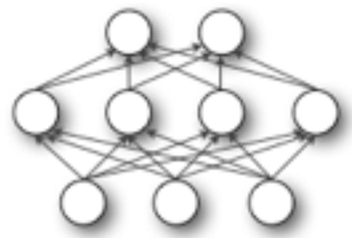
Machine Learning – CSE546
Kevin Jamieson
University of Washington

November 28, 2017

©Kevin Jamieson 2017


Training set

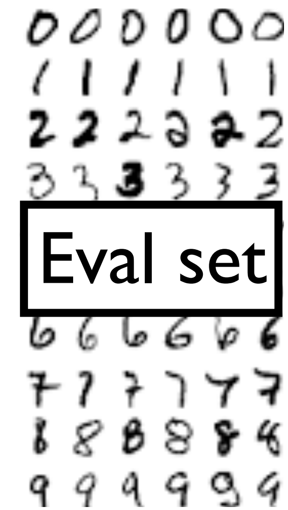

Eval set



$N_{out} = 10$

N_{hid}

$N_{in} = 784$

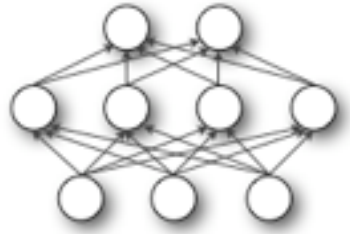


hyperparameters

learning rate $\eta \in [10^{-3}, 10^{-1}]$

ℓ_2-penalty $\lambda \in [10^{-6}, 10^{-1}]$

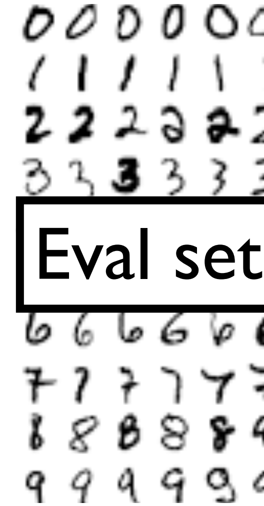
\# hidden nodes $N_{hid} \in [10^1, 10^3]$



$N_{out} = 10$

N_{hid}

$N_{in} = 784$



$\widehat{f}$

Hyperparameters

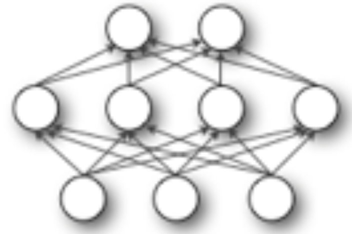
$(10^{-1.6}, 10^{-2.4}, 10^{1.7})$

hyperparameters

learning rate $\eta \in [10^{-3}, 10^{-1}]$

ℓ_2-penalty $\lambda \in [10^{-6}, 10^{-1}]$

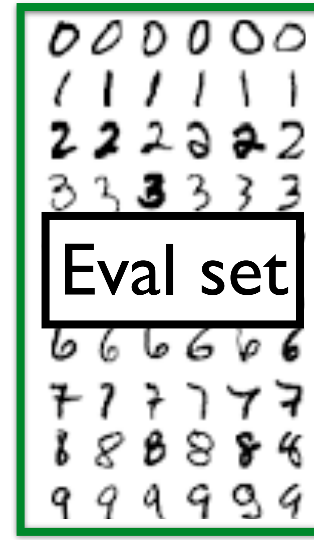
hidden nodes $N_{hid} \in [10^1, 10^3]$



$N_{out} = 10$

N_{hid}

$N_{in} = 784$



$\widehat{f}$

Hyperparameters

$(10^{-1.6}, 10^{-2.4}, 10^{1.7})$

Eval-loss

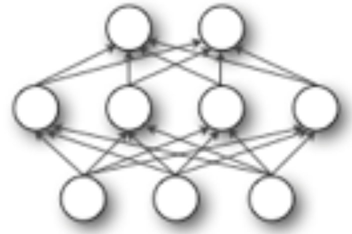
$\boxed{0.0577}$

hyperparameters

learning rate $\eta \in [10^{-3}, 10^{-1}]$

ℓ_2-penalty $\lambda \in [10^{-6}, 10^{-1}]$

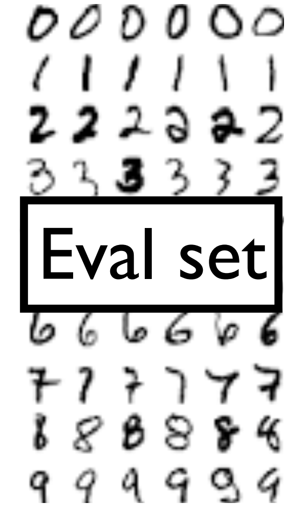
hidden nodes $N_{hid} \in [10^1, 10^3]$



$N_{out} = 10$

N_{hid}

$N_{in} = 784$



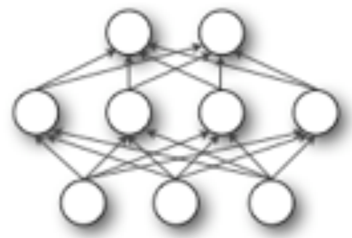
Hyperparameters	Eval-loss
$(10^{-1.6}, 10^{-2.4}, 10^{1.7})$	0.0577
$(10^{-1.0}, 10^{-1.2}, 10^{2.6})$	0.182

hyperparameters

learning rate $\eta \in [10^{-3}, 10^{-1}]$

ℓ_2-penalty $\lambda \in [10^{-6}, 10^{-1}]$

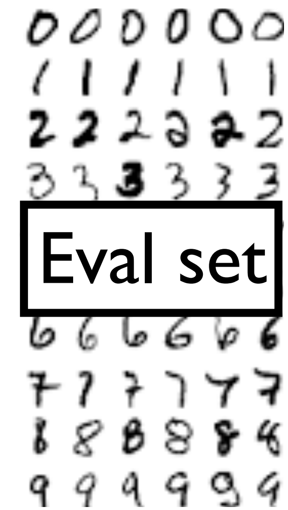
hidden nodes $N_{hid} \in [10^1, 10^3]$



$N_{out} = 10$

N_{hid}

$N_{in} = 784$



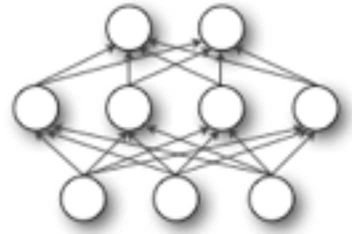
Hyperparameters	Eval-loss
$(10^{-1.6}, 10^{-2.4}, 10^{1.7})$	0.0577
$(10^{-1.0}, 10^{-1.2}, 10^{2.6})$	0.182
$(10^{-1.2}, 10^{-5.7}, 10^{1.4})$	0.0436

hyperparameters

learning rate $\eta \in [10^{-3}, 10^{-1}]$

ℓ_2-penalty $\lambda \in [10^{-6}, 10^{-1}]$

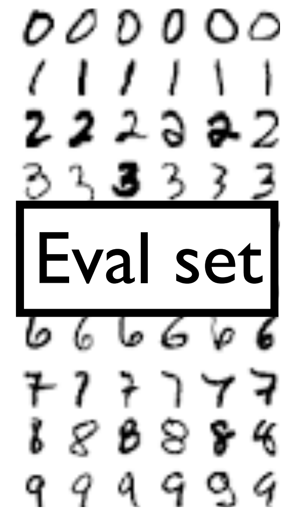
hidden nodes $N_{hid} \in [10^1, 10^3]$



$N_{out} = 10$

N_{hid}

$N_{in} = 784$



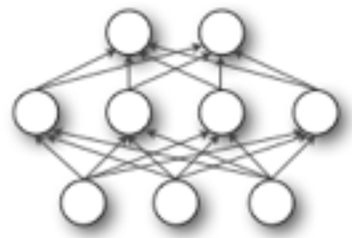
Hyperparameters	Eval-loss
$(10^{-1.6}, 10^{-2.4}, 10^{1.7})$	0.0577
$(10^{-1.0}, 10^{-1.2}, 10^{2.6})$	0.182
$(10^{-1.2}, 10^{-5.7}, 10^{1.4})$	0.0436
$(10^{-2.4}, 10^{-2.0}, 10^{2.9})$	0.0919
$(10^{-2.6}, 10^{-2.9}, 10^{1.9})$	0.0575
$(10^{-2.7}, 10^{-2.5}, 10^{2.4})$	0.0765
$(10^{-1.8}, 10^{-1.4}, 10^{2.6})$	0.1196
$(10^{-1.4}, 10^{-2.1}, 10^{1.5})$	0.0834
$(10^{-1.9}, 10^{-5.8}, 10^{2.1})$	0.0242
$(10^{-1.8}, 10^{-5.6}, 10^{1.7})$	0.029

hyperparameters

learning rate $\eta \in [10^{-3}, 10^{-1}]$

ℓ_2-penalty $\lambda \in [10^{-6}, 10^{-1}]$

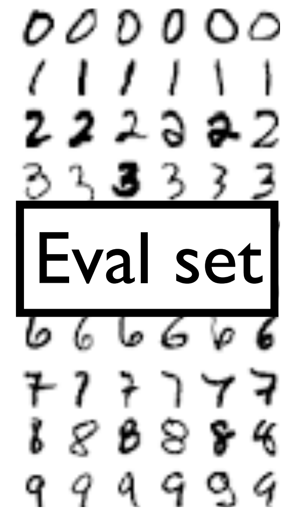
hidden nodes $N_{hid} \in [10^1, 10^3]$



$N_{out} = 10$

N_{hid}

$N_{in} = 784$



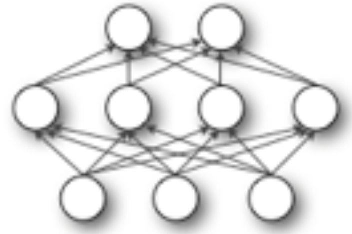
Hyperparameters	Eval-loss
$(10^{-1.6}, 10^{-2.4}, 10^{1.7})$	0.0577
$(10^{-1.0}, 10^{-1.2}, 10^{2.6})$	0.182
$(10^{-1.2}, 10^{-5.7}, 10^{1.4})$	0.0436
$(10^{-2.4}, 10^{-2.0}, 10^{2.9})$	0.0919
$(10^{-2.6}, 10^{-2.9}, 10^{1.9})$	0.0575
$(10^{-2.7}, 10^{-2.5}, 10^{2.4})$	0.0765
$(10^{-1.8}, 10^{-1.4}, 10^{2.6})$	0.1196
$(10^{-1.4}, 10^{-2.1}, 10^{1.5})$	0.0834
$(10^{-1.9}, 10^{-5.8}, 10^{2.1})$	0.0242
$(10^{-1.8}, 10^{-5.6}, 10^{1.7})$	0.029

hyperparameters

learning rate $\eta \in [10^{-3}, 10^{-1}]$

ℓ_2-penalty $\lambda \in [10^{-6}, 10^{-1}]$

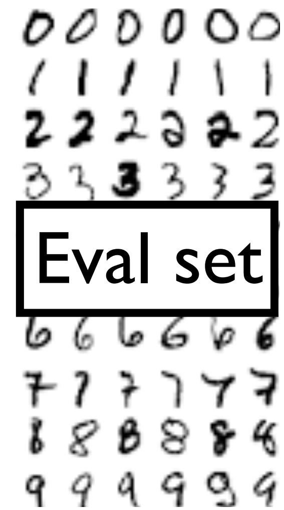
hidden nodes $N_{hid} \in [10^1, 10^3]$



$N_{out} = 10$

N_{hid}

$N_{in} = 784$

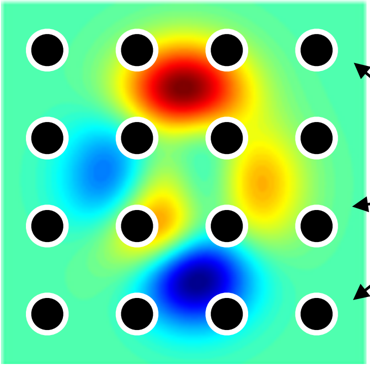


Hyperparameters	Eval-loss
$(10^{-1.6}, 10^{-2.4}, 10^{1.7})$	0.0577
$(10^{-1.0}, 10^{-1.2}, 10^{2.6})$	0.182
$(10^{-1.2}, 10^{-5.7}, 10^{1.4})$	0.0436
$(10^{-2.4}, 10^{-2.0}, 10^{2.9})$	0.0919
$(10^{-2.6}, 10^{-2.9}, 10^{1.9})$	0.0575
$(10^{-2.7}, 10^{-2.5}, 10^{2.4})$	0.0765
$(10^{-1.8}, 10^{-1.4}, 10^{2.6})$	0.1196
$(10^{-1.4}, 10^{-2.1}, 10^{1.5})$	0.0834
$(10^{-1.9}, 10^{-5.8}, 10^{2.1})$	0.0242
$(10^{-1.8}, 10^{-5.6}, 10^{1.7})$	0.029

How do we choose hyperparameters to train and evaluate?

How do we choose hyperparameters to train and evaluate?

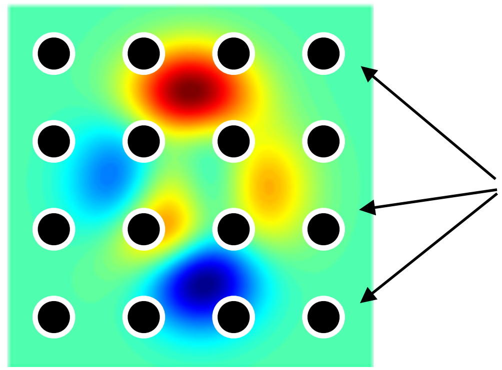
Grid search:



Hyperparameters on 2d uniform grid

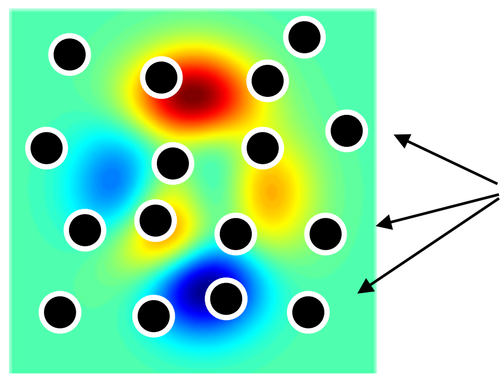
How do we choose hyperparameters to train and evaluate?

Grid search:



Hyperparameters on 2d uniform grid

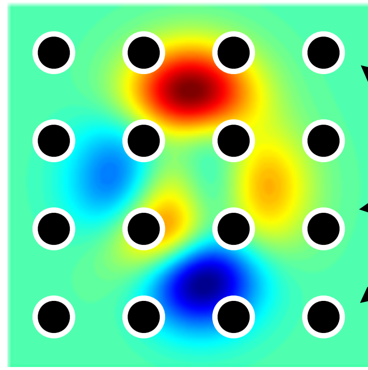
Random search:



Hyperparameters randomly chosen

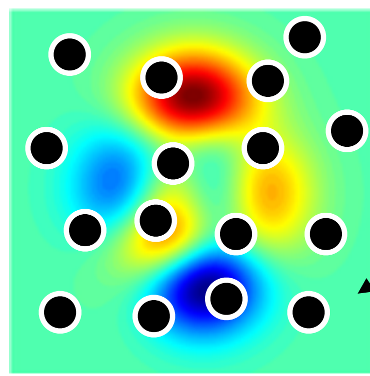
How do we choose hyperparameters to train and evaluate?

Grid search:



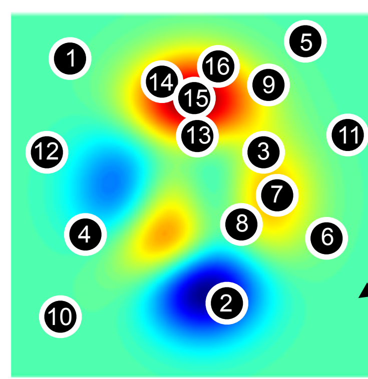
Hyperparameters on 2d uniform grid

Random search:



Hyperparameters randomly chosen

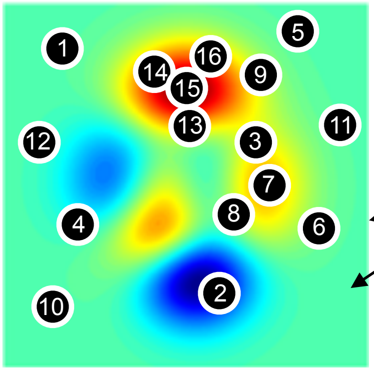
Bayesian Optimization:



Hyperparameters *adaptively* chosen

Bayesian Optimization:

How does it work?



Hyperparameters *adaptively* chosen

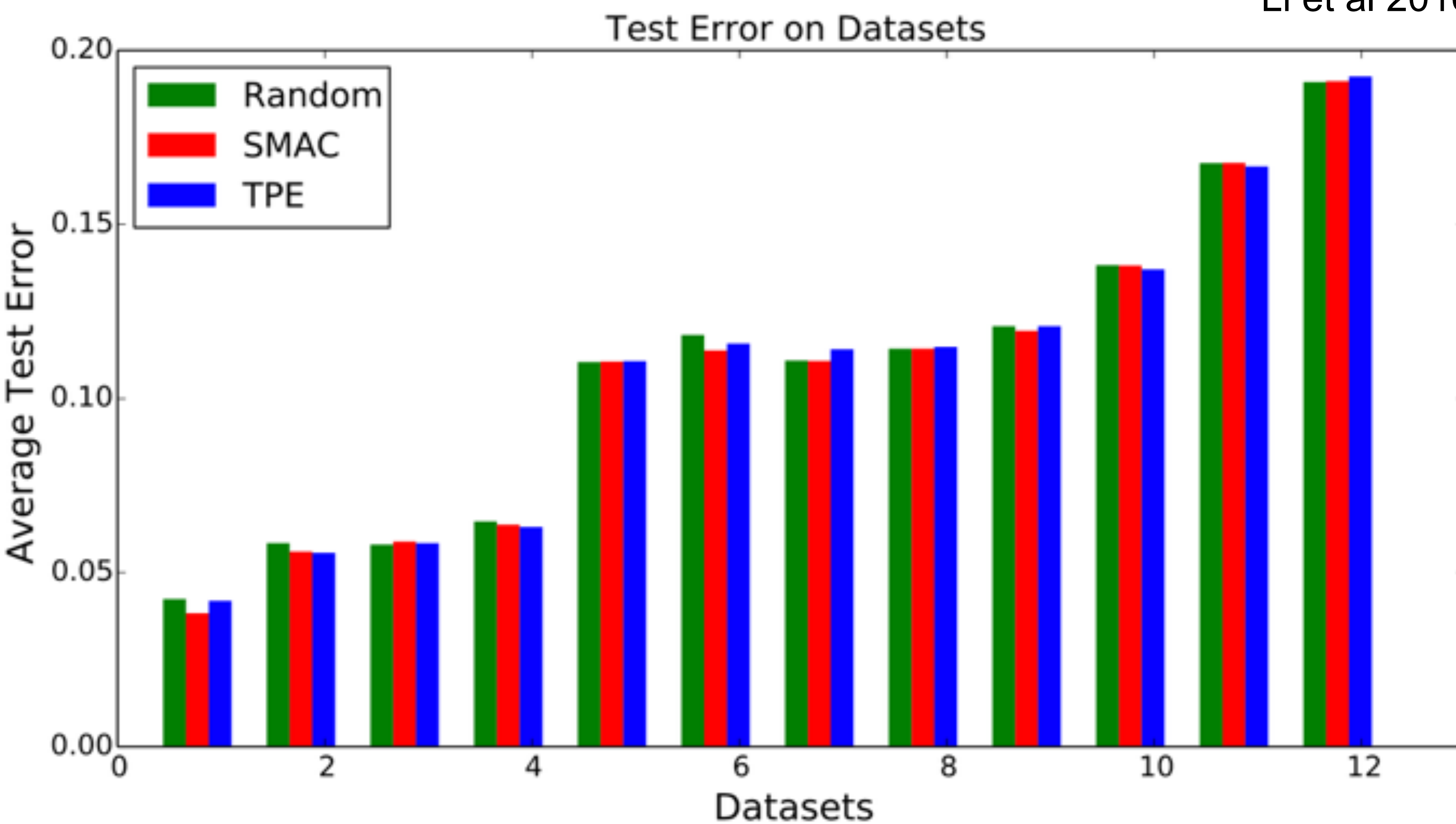
E. Sparks, A. Talwalkar, D. Haas, M. J. Franklin, M. I. Jordan, T. Kraska. "Automating Model Search for Large Scale Machine Learning," In Symposium on Cloud Computing, 2015.

E. Sparks, A. Talwalkar, D. Haas, M. J. Franklin, M. I. Jordan, T. Kraska. "Automating Model Search for Large Scale Machine Learning," In Symposium on Cloud Computing, 2015.

~15 dimensional hyperparameter space

Test error of output hyperparameter setting from each searcher after 1 hour per dataset

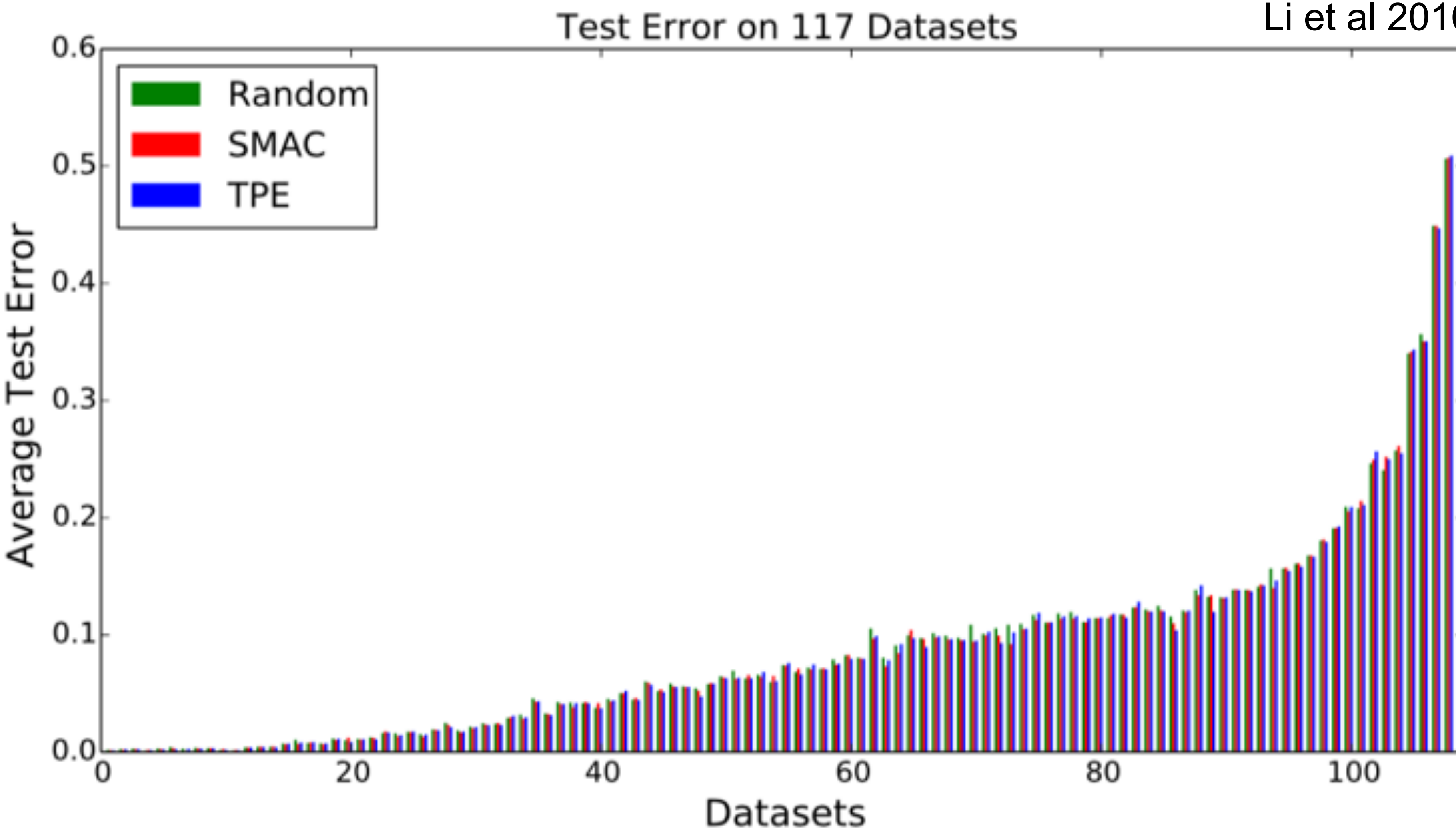
Li et al 2016



~15 dimensional hyperparameter space

Test error of output hyperparameter setting from each searcher after 1 hour per dataset

Li et al 2016



Recent work attempts to speed up hyperparameter evaluation by stopping poor performing settings before they are fully trained.

Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw bayesian optimization. arXiv:1406.3896, 2014.

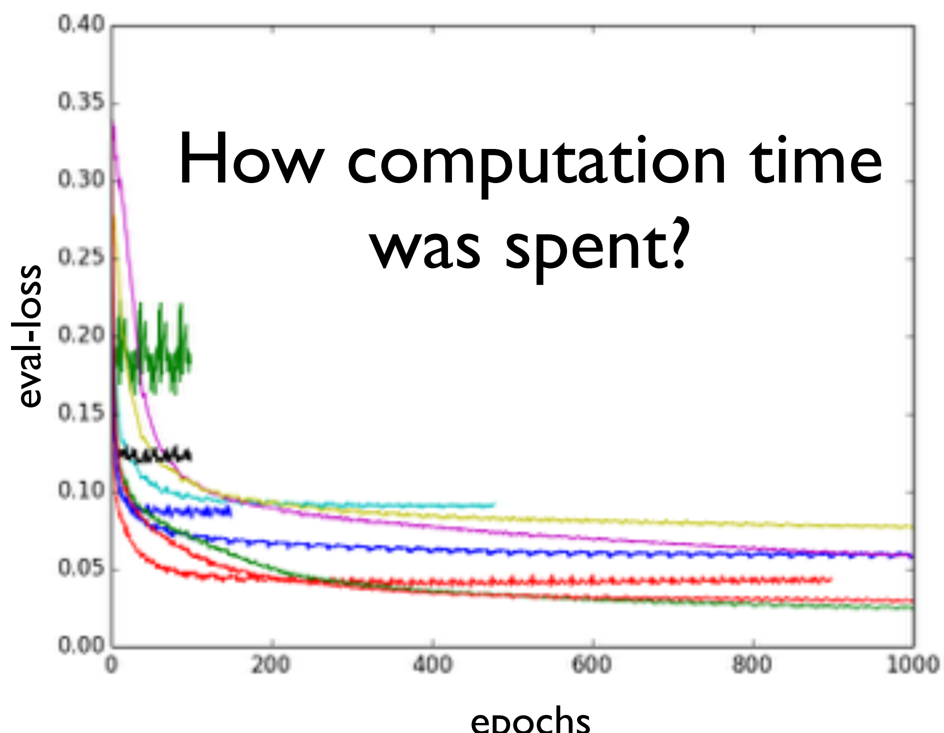
Alekh Agarwal, Peter Bartlett, and John Duchi. Oracle inequalities for computationally adaptive model selection. COLT, 2012.

Domhan, T., Springenberg, J. T., and Hutter, F. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *IJCAI*, 2015.

András György and Levente Kocsis. Efficient multi-start strategies for local search algorithms. JAIR, 41, 2011.

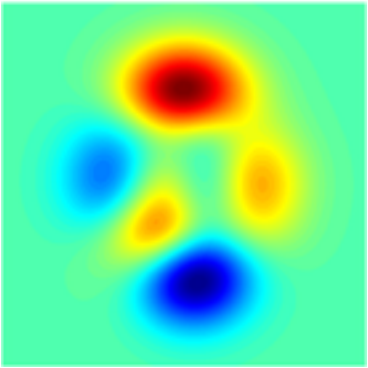
Li, Jamieson, DeSalvo, Rostamizadeh, Talwalkar. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. ICLR 2016.

Hyperparameters	Eval-loss
$(10^{-1.6}, 10^{-2.4}, 10^{1.7})$	0.0577
$(10^{-1.0}, 10^{-1.2}, 10^{2.6})$	0.182
$(10^{-1.2}, 10^{-5.7}, 10^{1.4})$	0.0436
$(10^{-2.4}, 10^{-2.0}, 10^{2.9})$	0.0919
$(10^{-2.6}, 10^{-2.9}, 10^{1.9})$	0.0575
$(10^{-2.7}, 10^{-2.5}, 10^{2.4})$	0.0765
$(10^{-1.8}, 10^{-1.4}, 10^{2.6})$	0.1196
$(10^{-1.4}, 10^{-2.1}, 10^{1.5})$	0.0834
$(10^{-1.9}, 10^{-5.8}, 10^{2.1})$	0.0242
$(10^{-1.8}, 10^{-5.6}, 10^{1.7})$	0.029



Hyperparameter Optimization

In general, hyperparameter optimization is non-convex optimization and little is known about the underlying function (only observe validation loss)



Your time is valuable, computers are cheap:
Do not employ "grad student descent" for hyper parameter search.
Write modular code that takes parameters as input and automate this embarrassingly parallel search. Use crowd resources (see **`pywren`**)

Tools for different purposes:
- Very few evaluations: use random search (and pray) or be *clever*
- Few evaluations and long-running computations: see refs on last slide
- Moderate number of evaluations (but still exp(#params)) and high accuracy needed: use Bayesian Optimization
- Many evaluations possible: use random search. Why overthink it?

Convolutional Neural Networks & Application to Computer Vision

Machine Learning – CSE4546
Kevin Jamieson
University of Washington

November 28, 2017

Contains slides from…

- LeCun & Ranzato
- Russ Salakhutdinov
- Honglak Lee
- Google images…

Convolution of images

(Note to EEs: deep learning uses the word “convolution” to mean what is usually known as “cross-correlation”, e.g., neither signal is flipped)

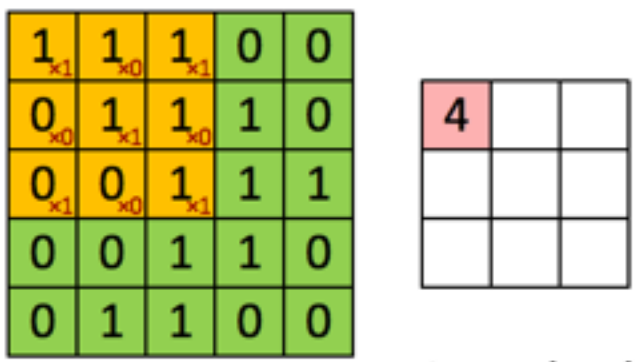
$$(I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image I

1	0	1
0	1	0
1	0	1

Filter K



Image

Convolved Feature

$I * K$

Slide credit: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Convolution of images

(Note to EEs: deep learning uses the word “convolution” to mean what is usually known as “cross-correlation”, e.g., neither signal is flipped)

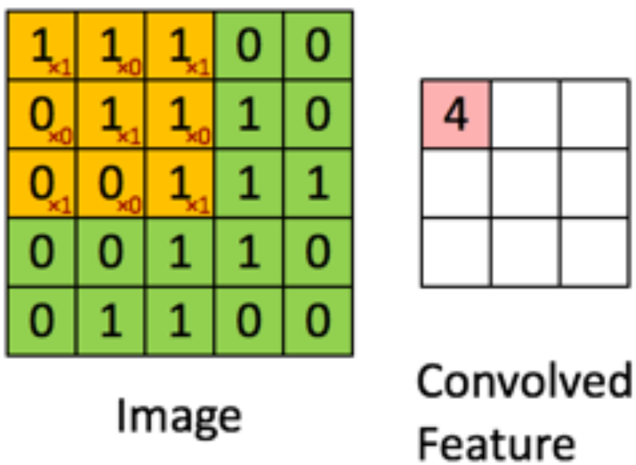
$$(I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image I

1	0	1
0	1	0
1	0	1

Filter K



$I * K$

Slide credit: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Convolution of images

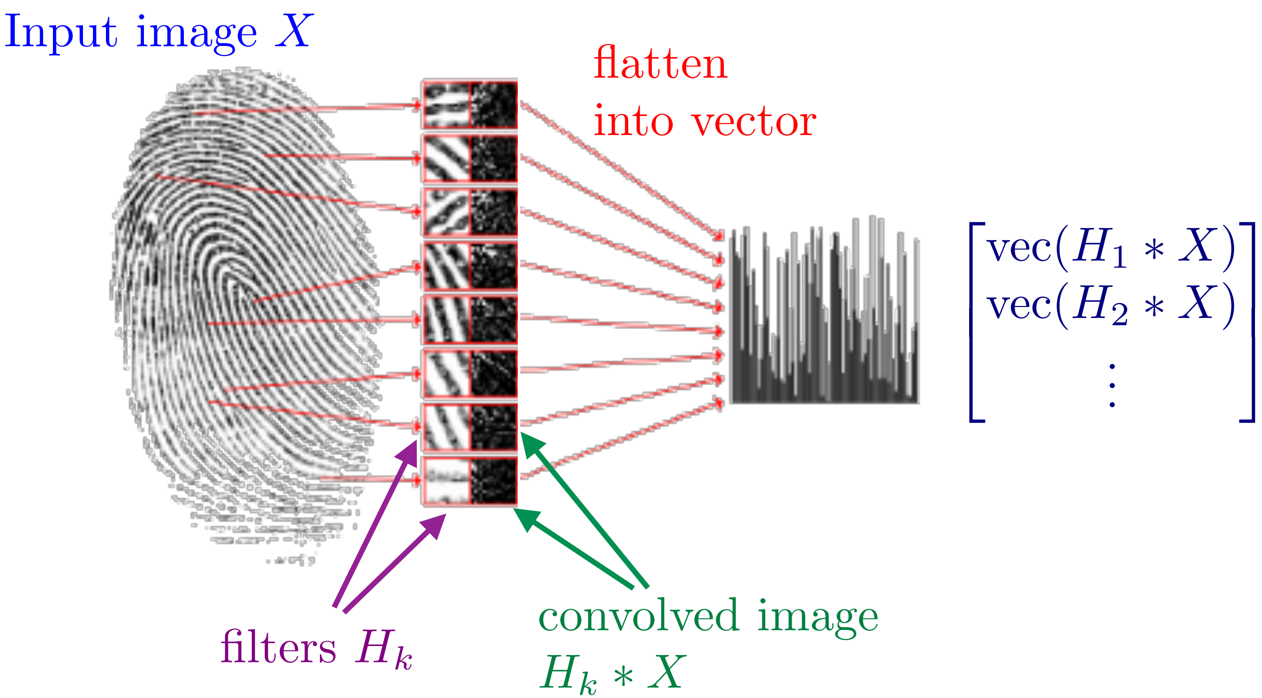
(Note to EEs: deep learning uses the word “convolution” to mean what is usually known as “cross-correlation”, e.g., neither signal is flipped)

$$(I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

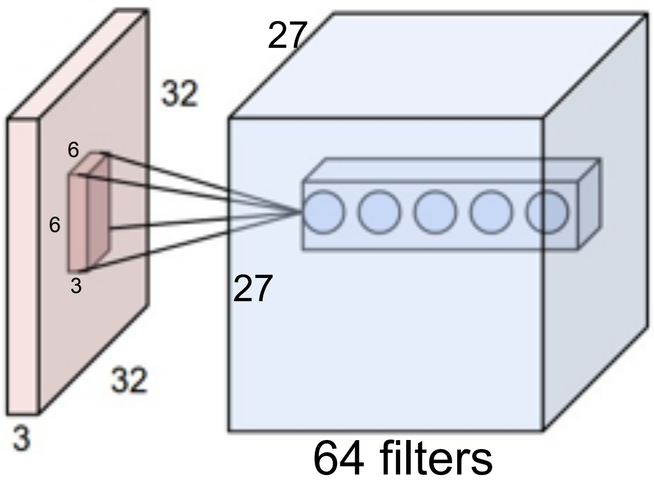
Image I

Operation	Filter K	Convolved Image $I * K$
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

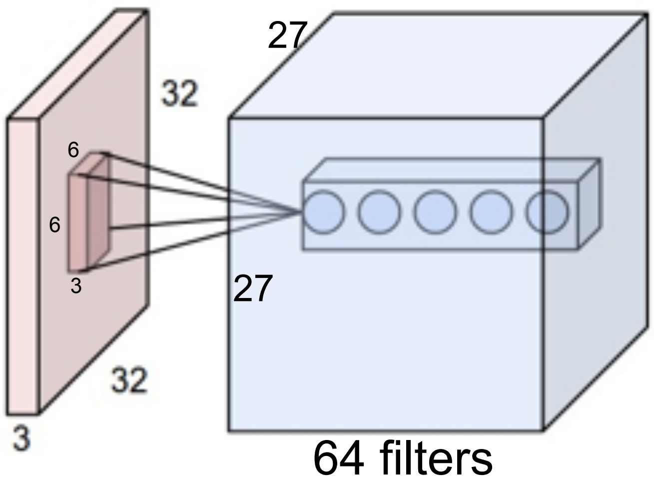
Convolution of images



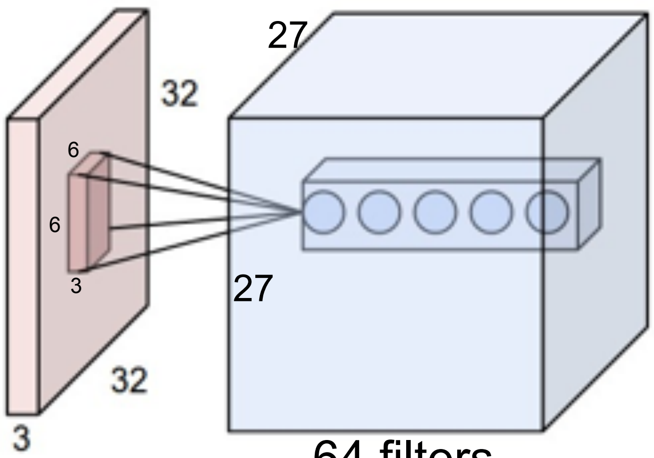
Stacking convolved images



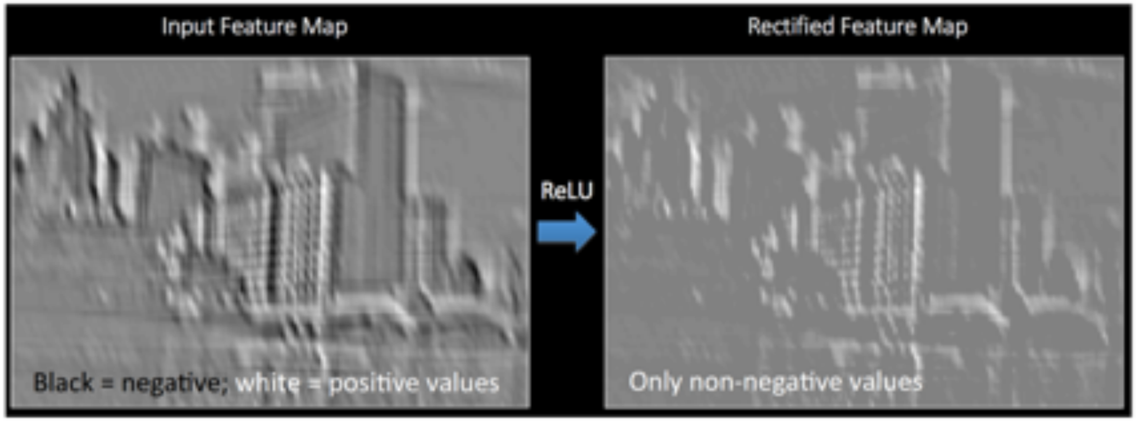
Stacking convolved images



Stacking convolved images

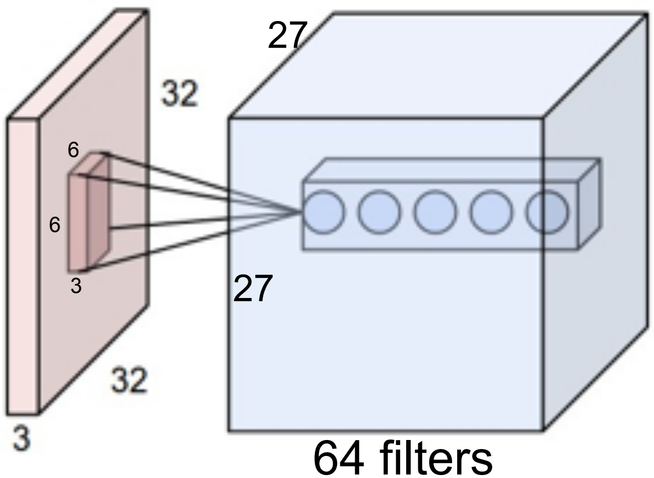


Apply Non-linearity to the output of each layer, Here: ReLu (rectified linear unit)

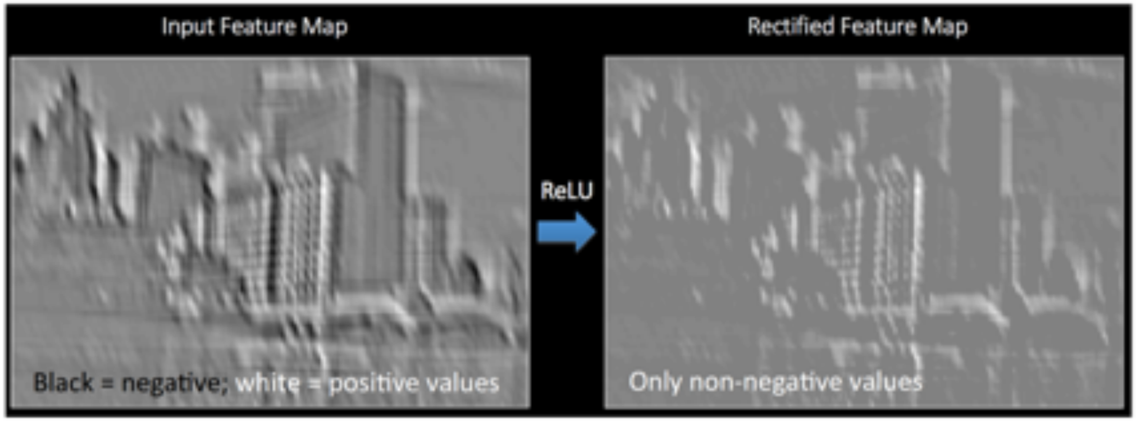


Other choices: sigmoid, arctan

Stacking convolved images



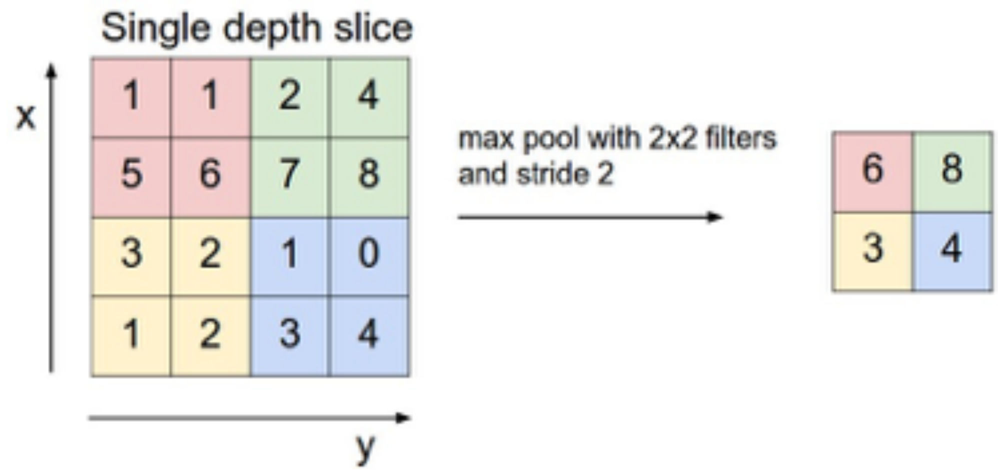
Apply Non-linearity to the output of each layer, Here: ReLu (rectified linear unit)

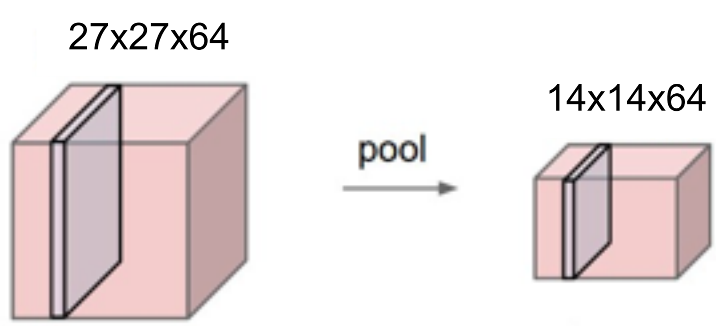


Other choices: sigmoid, arctan

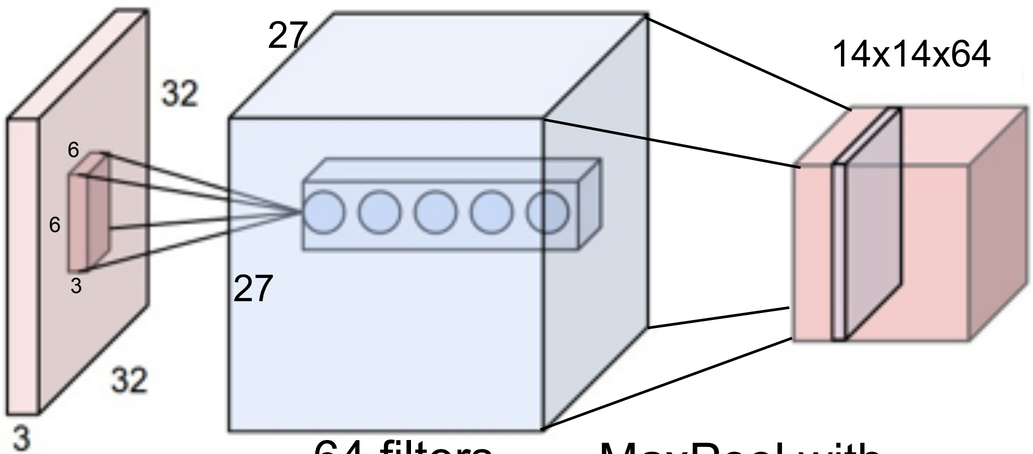
Pooling

Pooling reduces the dimension and can be interpreted as "This filter had a high response in this general region"





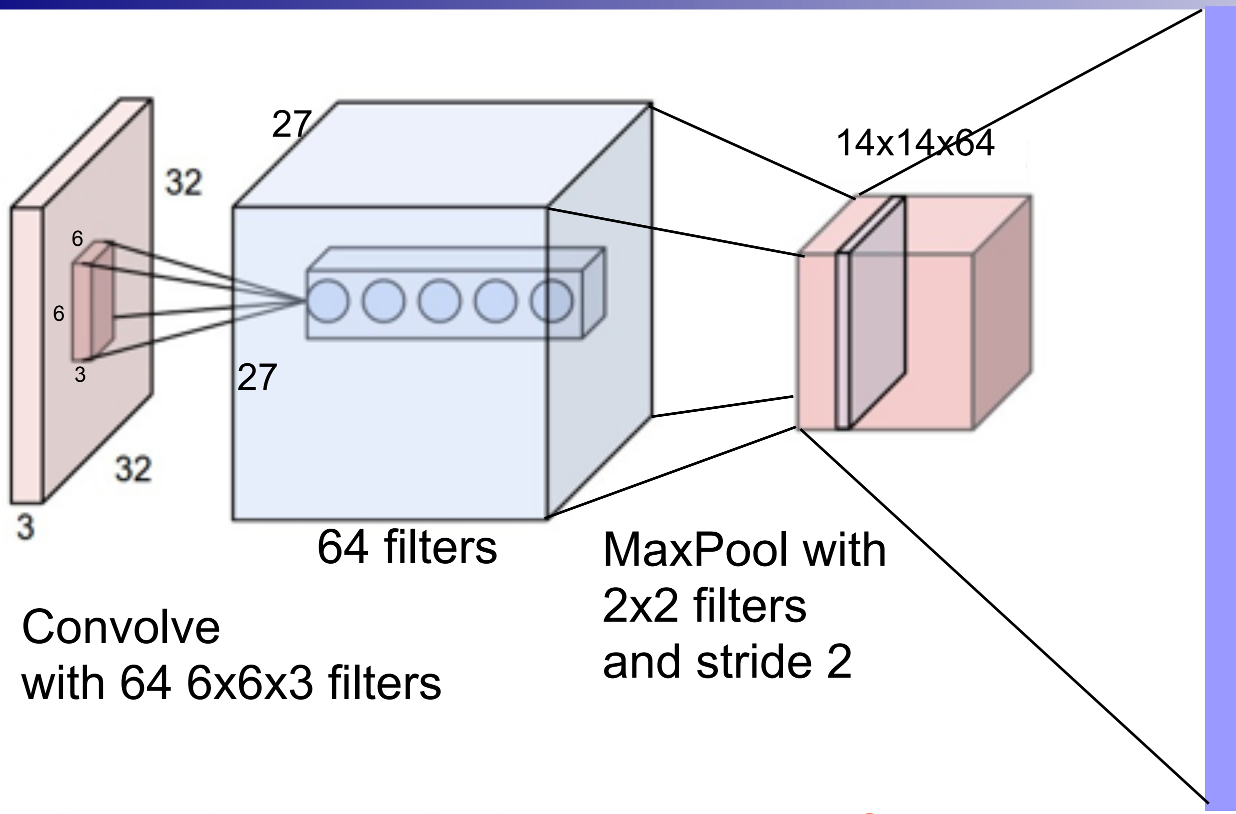
Pooling Convolution layer



Convolve
with 64 6x6x3 filters

MaxPool with
2x2 filters
and stride 2

Full feature pipeline



Convolve
with 64 6x6x3 filters

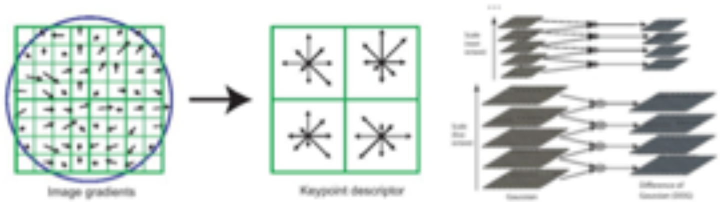
MaxPool with
2x2 filters
and stride 2

Flatten into a single
vector of size
14*14*64=12544

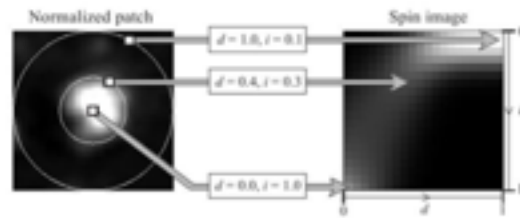
How do we choose all the hyperparameters?

How do we choose the filters?
- Hand design them (digital signal processing, c.f. wavelets)
- Learn them (deep learning)

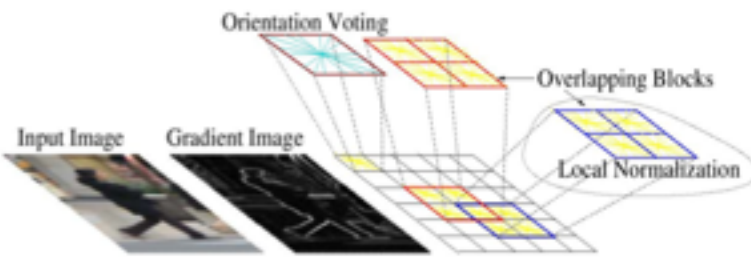
Some hand-created image features



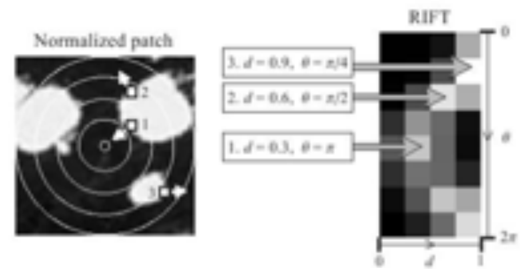
SIFT



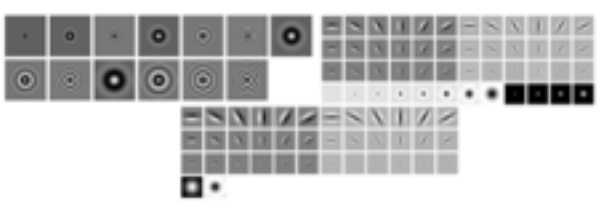
Spin Image



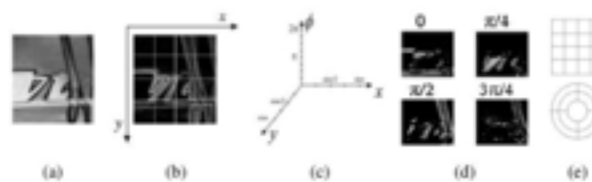
HoG



RIFT



Texton



GLOH

Slide from Honglak Lee

Mini case study 1/3

Inspired by Coates and Ng (2012)

Inspired by Coates and Ng (2012)

Input is CIFAR-10 dataset: 50000 examples of 32x32x3 images

1. Construct set of patches by random selection from images
2. Standardize patch set (de-mean, norm 1, whiten, etc.)
3. Run k-means on random patches
4. Convolve each image with all patches (plus an offset)
5. Push through ReLu
6. Solve least squares for multiclass classification
7. Classify with argmax

Mini case study 2/3

Inspired by Coates and Ng (2012)

Methods of standardization:

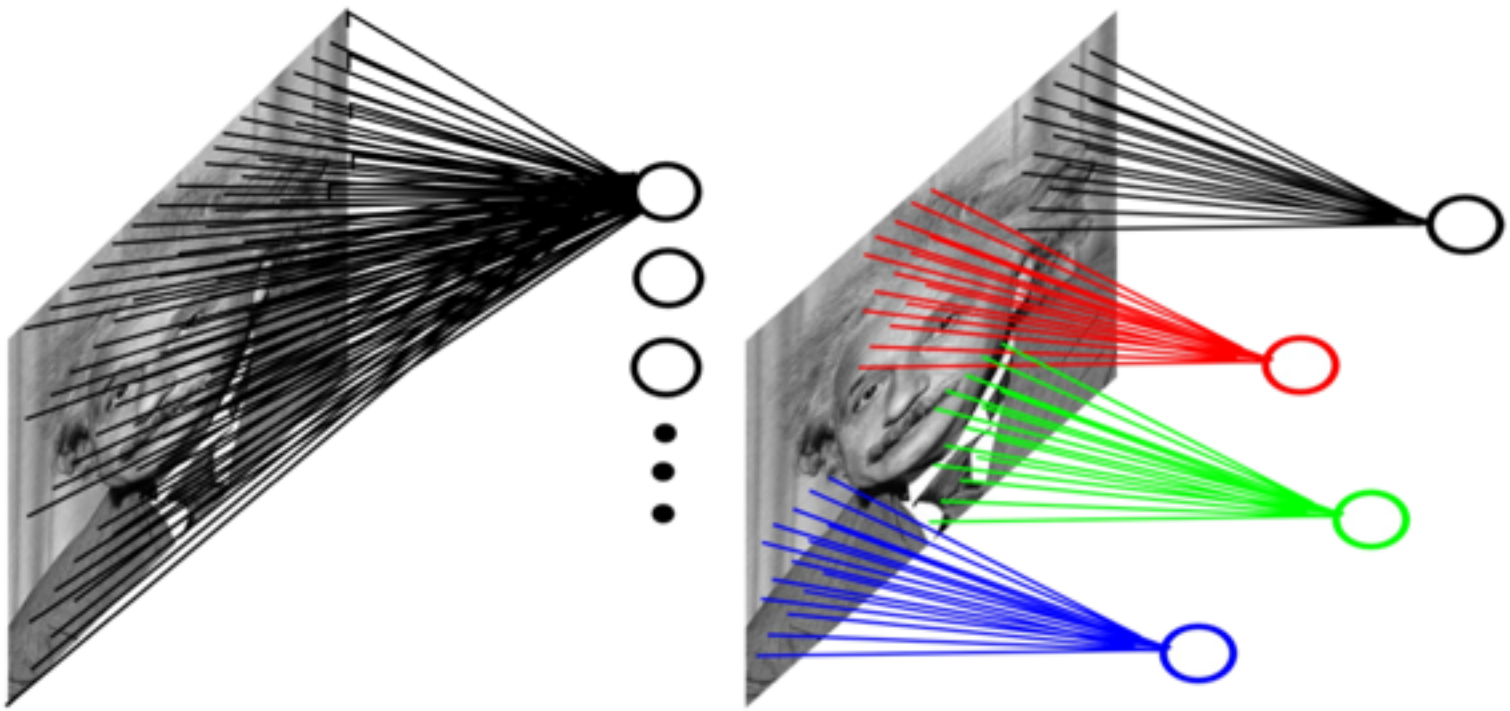
Mini case study 3/3

Inspired by Coates and Ng (2012)

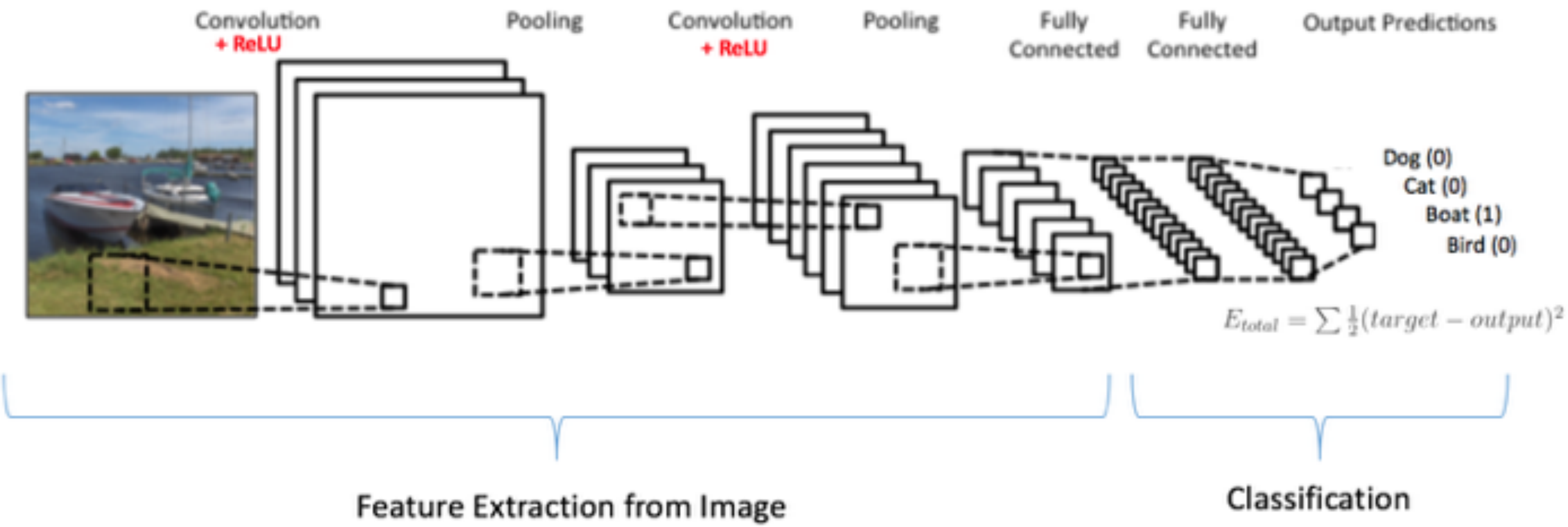
Dealing with class imbalance:

Convolution Layer

- **Example: 200x200 image**
  - Fully-connected, 400,000 hidden units = 16 billion parameters
  - Locally-connected, 400,000 hidden units 10x10 fields = 40 million params
  - Local connections capture local dependencies



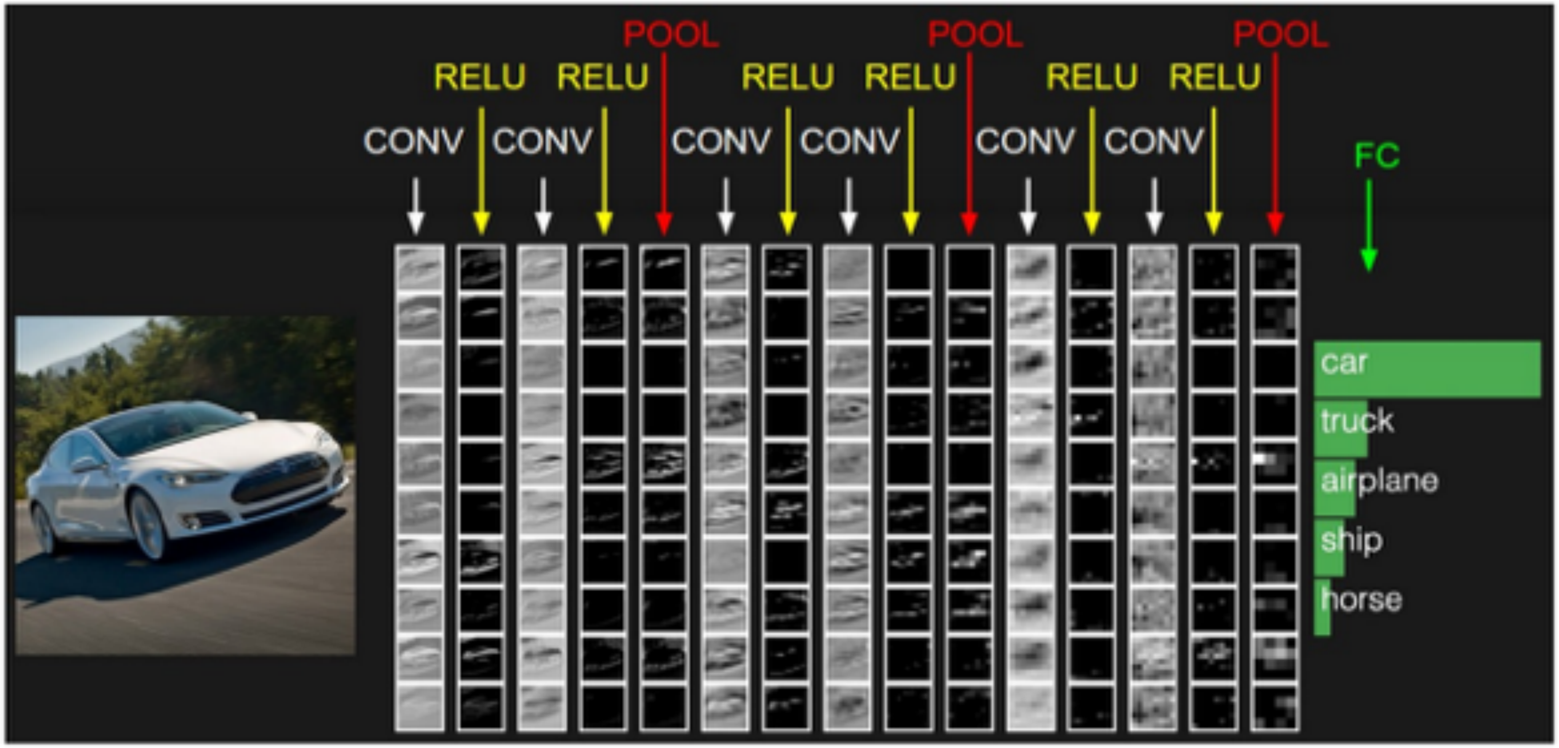
Could be very complicated…



Learn the convolutional filters using back propagation.

Once learned, you can fix and apply the learned features to other datasets, and only learn the last fully connected layers.

Could be very complicated…

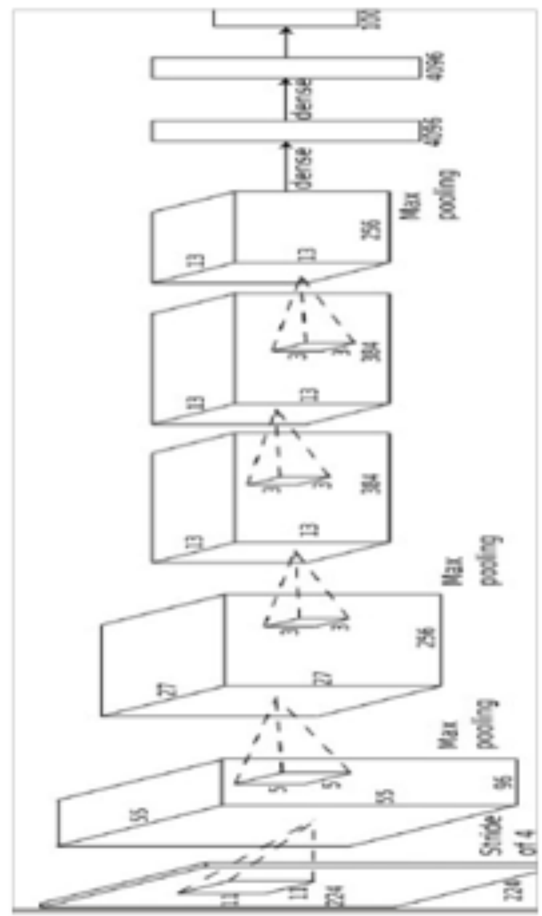


Different architectures have different effects (not well understood)

Example from Krizhevsky, Sutskever, Hinton 2012

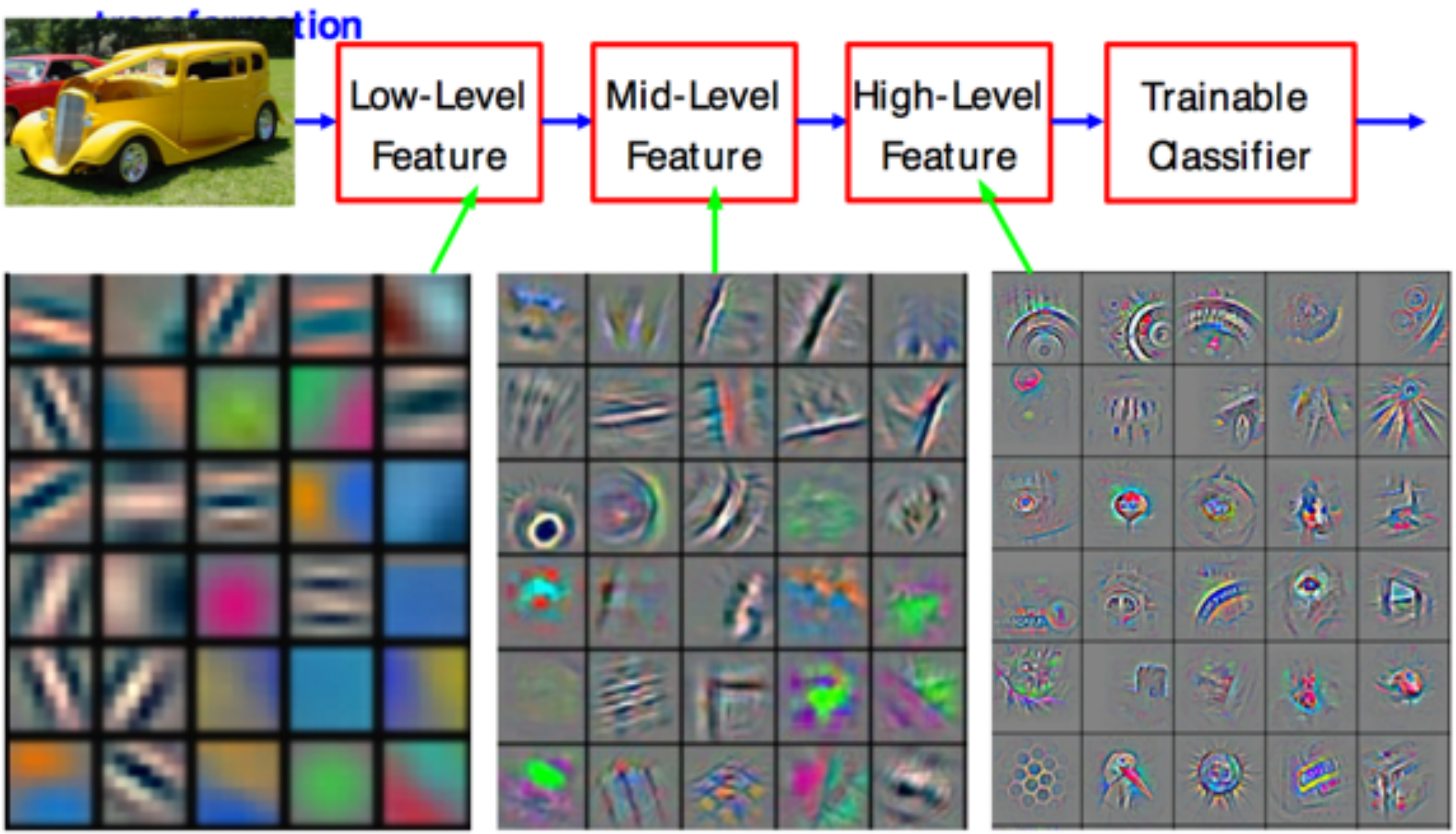
- Won the 2012 ImageNet LSVRC. 60 Million parameters, 832M MAC ops

4M	FULL CONNECT	4Mflop
16M	FULL 4096/ReLU	16M
37M	FULL 4096/ReLU	37M
	MAX POOLING	
442K	CONV 3x3/ReLU 256fm	74M
1.3M	CONV 3x3ReLU 384fm	224M
884K	CONV 3x3/ReLU 384fm	149M
	MAX POOLING 2x2sub	
	LOCAL CONTRAST NORM	
307K	CONV 11x11/ReLU 256fm	223M
	MAX POOL 2x2sub	
	LOCAL CONTRAST NORM	
35K	CONV 11x11/ReLU 96fm	105M




mite
container ship
motor scooter
leopard
mite
black widow
cockroach
tick
starfish
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
go-kart
moped
bumper car
golfcart
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
mushroom
cherry
Madagascar cat
convertible
grille
pickup
beach wagon
fire engine
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
dalmatian
grape
elderberry
ffordshire bullterrier
currant
squirrel monkey
spider monkey
titi
indri
howler monkey

Using neural nets to learn non-linear features



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Sequences and Recurrent Neural Networks

Machine Learning – CSE4546

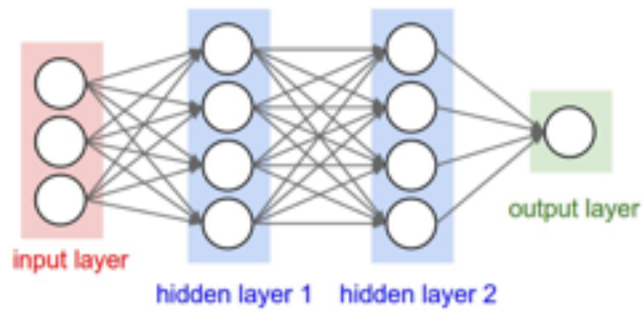
Kevin Jamieson
University of Washington

November 28, 2017

Variable length sequences

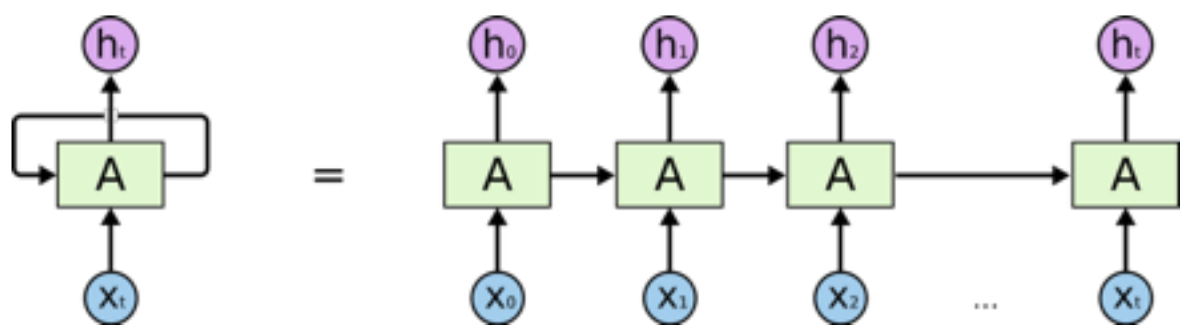
Images are usually standardized to be the same size (e.g., 256x256x3)

Neural Network



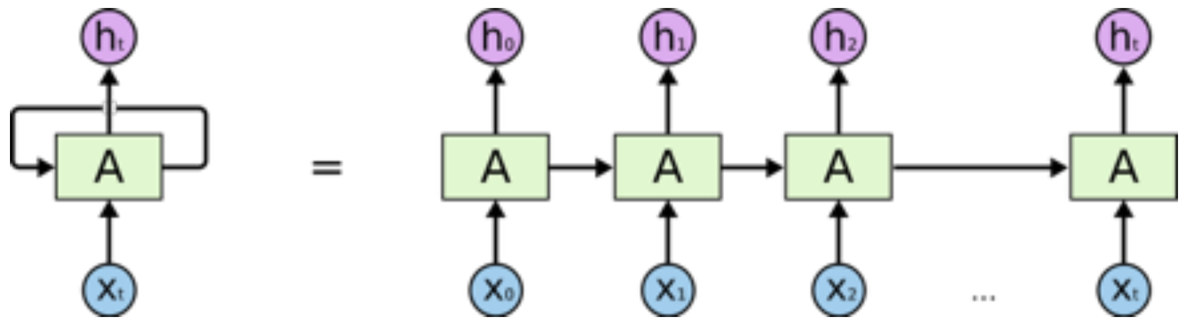
But what if we wanted to do classification on country-of-origin for names?

Recurrent Neural Network

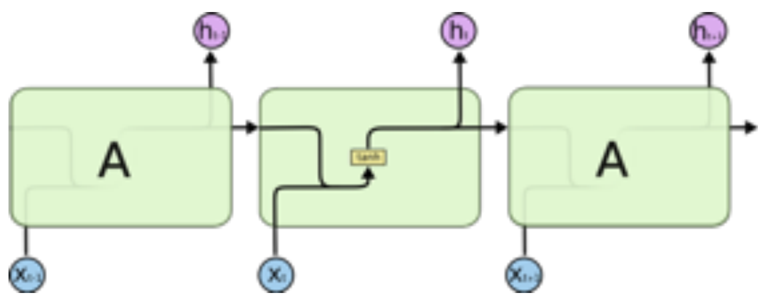


Variable length sequences

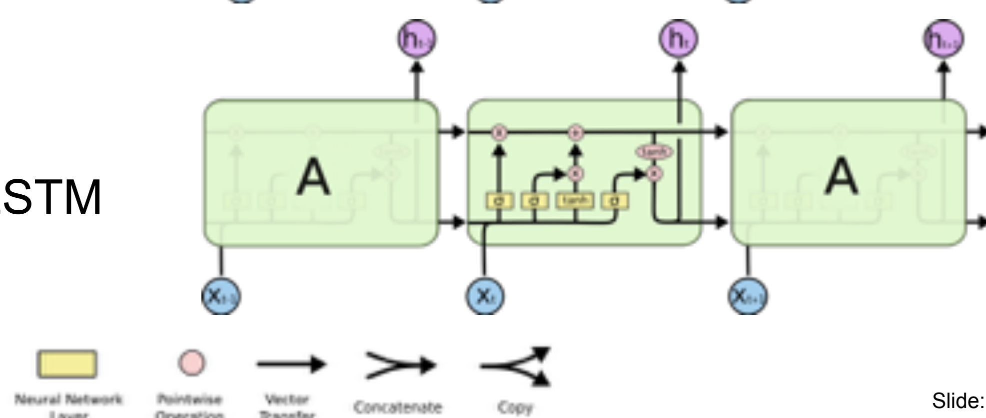
Recurrent Neural Network



Standard RNN



LSTM



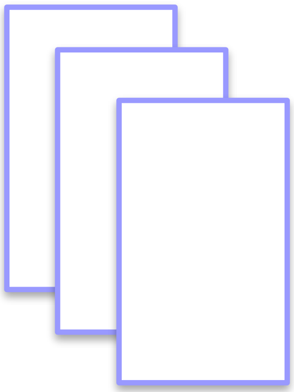
Slide: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Basic Text/Document Processing

Machine Learning – CSE4546
Kevin Jamieson
University of Washington

November 28, 2017

TF*IDF



n documents/articles with lots of text

How to get a feature representation of each article?

1. For each document *d* compute the proportion of times word *t* occurs out of all words in *d*, i.e. **term frequency**

$$TF_{d,t}$$

2. For each word *t* in your corpus, compute the proportion of documents out of *n* that the word *t* occurs, i.e., **document frequency**

$$DF_t$$

3. Compute score for word *t* in document *d* as $TF_{d,t} \log(\frac{1}{DF_t})$

BeerMapper - Under the Hood

Algorithm requires feature representations of the beers $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$

ratebeer

Two Hearted Ale - Input ~2500 natural language reviews

http://www.ratebeer.com/beer/two-hearted-ale/1502/2/1/

3.8 AROMA 8/10 APPEARANCE 4/5 TASTE 8/10 PALATE 3/5 OVERALL 15/20

fonefan (25678) - VestJylland, DENMARK - JAN 18, 2009

Bottle 355ml.

Clear light to medium yellow orange color with a average, frothy, good lacing, fully lasting, off-white head. Aroma is moderate to heavy malty, moderate to heavy hoppy, perfume, grapefruit, orange shell, soap. Flavor is moderate to heavy sweet and bitter with a average to long duration. Body is medium, texture is oily, carbonation is soft. [250908]

4 AROMA 8/10 APPEARANCE 4/5 TASTE 7/10 PALATE 4/5 OVERALL 17/20

Ungstrup (24358) - Oamaru, NEW ZEALAND - MAR 31, 2005

An orange beer with a huge off-white head. The aroma is sweet and very freshly hoppy with notes of hop oils - very powerful aroma. The flavor is sweet and quite hoppy, that gives flavors of oranges, flowers as well as hints of grapefruit. Very refreshing yet with a powerful body.

Reviews for each beer → Bag of Words weighted by TF*IDF → Get 100 nearest neighbors using cosine distance → Non-metric multidimensional scaling → Embedding in d dimensions

BeerMapper - Under the Hood

Algorithm requires feature representations of the beers $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$

<u>Two Hearted Ale - Weighted Bag of Words:</u>

Reviews for each beer

Bag of Words weighted by TF*IDF

Get 100 nearest neighbors using cosine distance

Non-metric multidimensional scaling

Embedding in d dimensions

BeerMapper - Under the Hood

Algorithm requires feature representations of the beers $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$

Weighted count vector for the ith beer:

$$z_i \in \mathbb{R}^{400,000}$$

Cosine distance:

$$d(z_i, z_j) = 1 - \frac{z_i^T z_j}{||z_i|| \, ||z_j||}$$

<u>Two Hearted Ale - Nearest Neighbors:</u>

Bear Republic Racer 5
Avery IPA
Stone India Pale Ale (IPA)
Founders Centennial IPA
Smuttynose IPA
Anderson Valley Hop Ottin IPA
AleSmith IPA
BridgePort IPA
Boulder Beer Mojo IPA
Goose Island India Pale Ale
Great Divide Titan IPA
New Holland Mad Hatter Ale
Lagunitas India Pale Ale
Heavy Seas Loose Cannon Hop3
Sweetwater IPA

Reviews for each beer

Bag of Words weighted by TF*IDF

Get 100 nearest neighbors using cosine distance

Non-metric multidimensional scaling

Embedding in d dimensions

BeerMapper - Under the Hood

Algorithm requires feature representations of the beers $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$

Find an embedding $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ such that

$$||x_k - x_i|| < ||x_k - x_j|| \text{ whenever } \underline{d(z_k, z_i)} < \underline{d(z_k, z_j)}$$

for all 100-nearest neighbors.

(10^7 constraints, 10^5 variables)

distance in 400,000 dimensional "word space"

Solve with hinge loss and stochastic gradient descent.

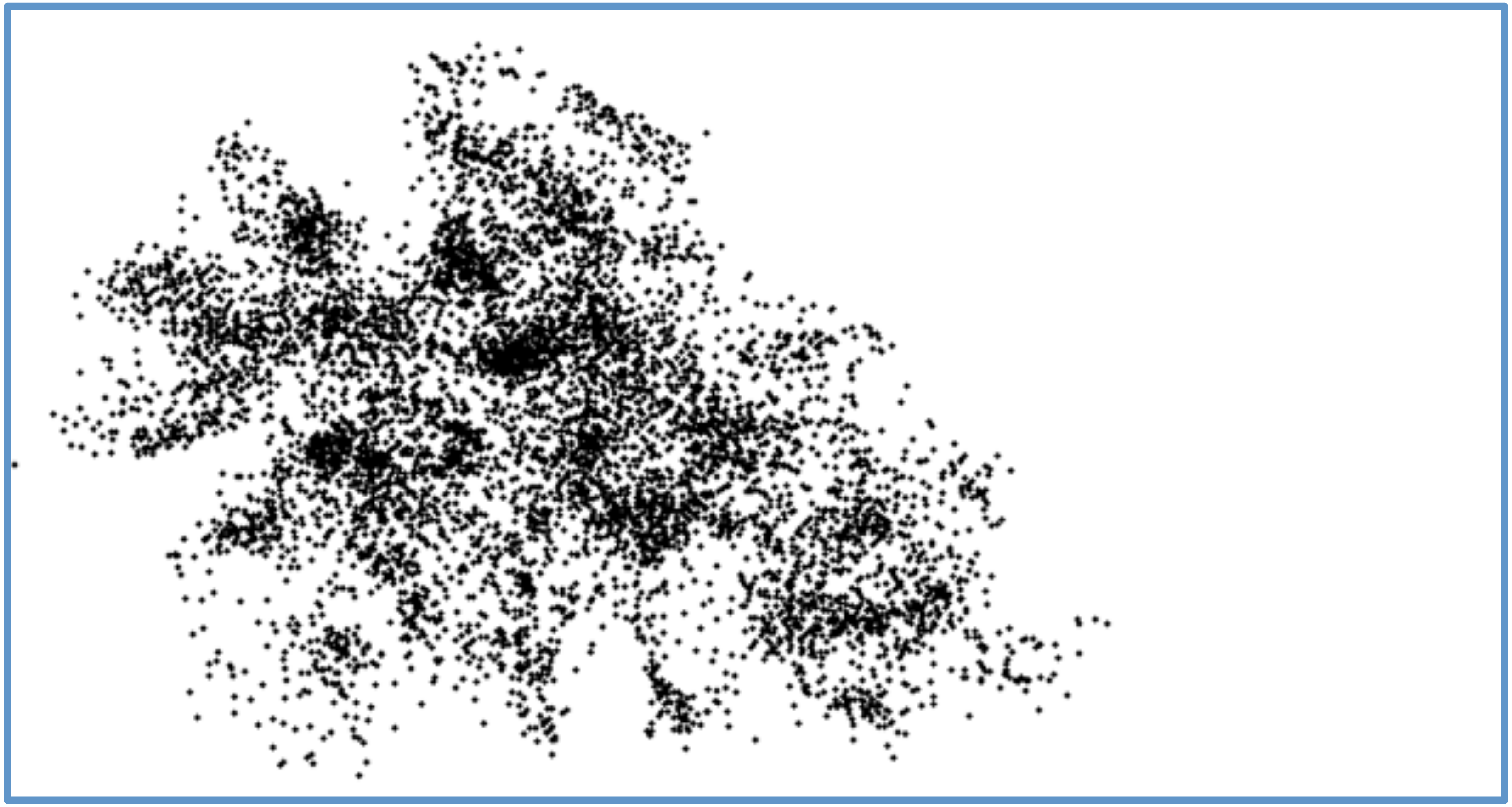
(20 minutes on my laptop) (d=2,err=6%) (d=3,err=4%)

Could have also used local-linear-embedding, max-volume-unfolding, kernel-PCA, etc.

Reviews for each beer → Bag of Words weighted by TF*IDF → Get 100 nearest neighbors using cosine distance → **Non-metric multidimensional scaling** → Embedding in d dimensions

BeerMapper - Under the Hood

Algorithm requires feature representations of the beers $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$



Reviews for each beer

Bag of Words weighted by TF*IDF

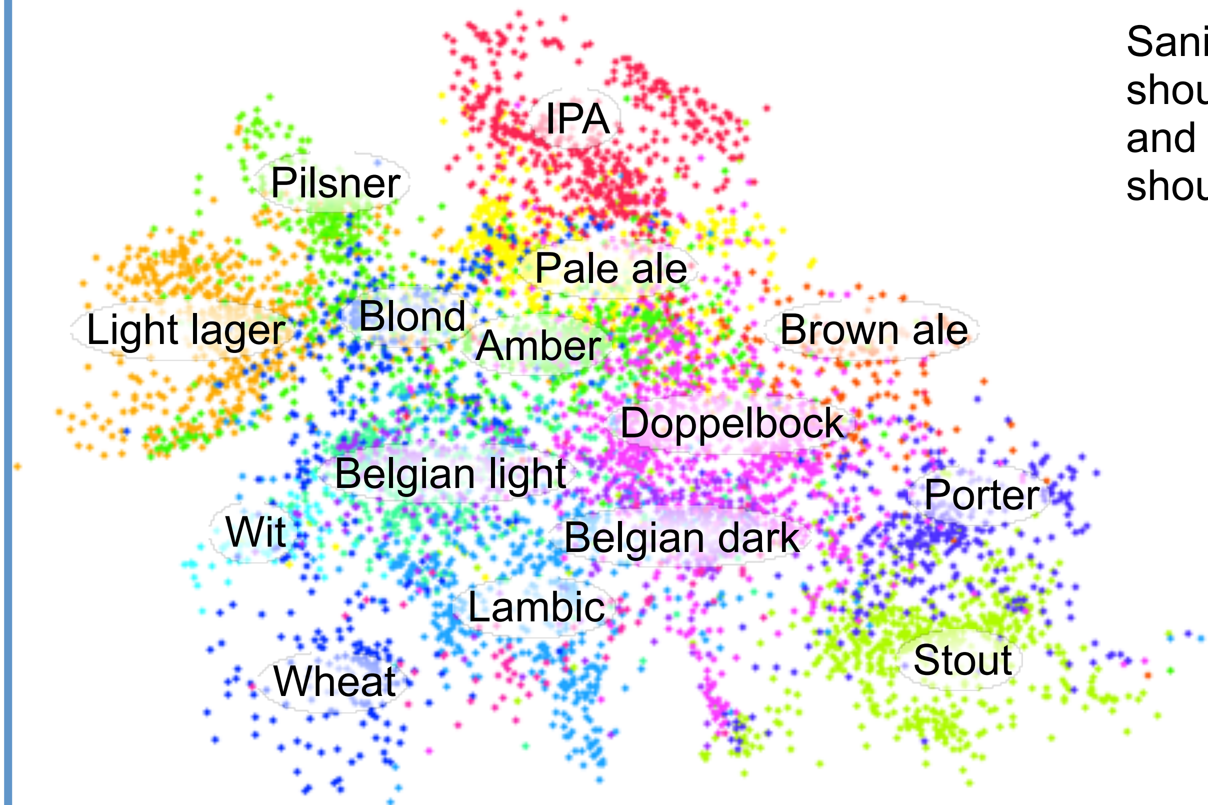
Get 100 nearest neighbors using cosine distance

Non-metric multidimensional scaling

Embedding in d dimensions

BeerMapper - Under the Hood

Algorithm requires feature representations of the beers $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$



Sanity check: styles should cluster together and similar styles should be close.

Reviews for each beer

Bag of Words weighted by TF*IDF

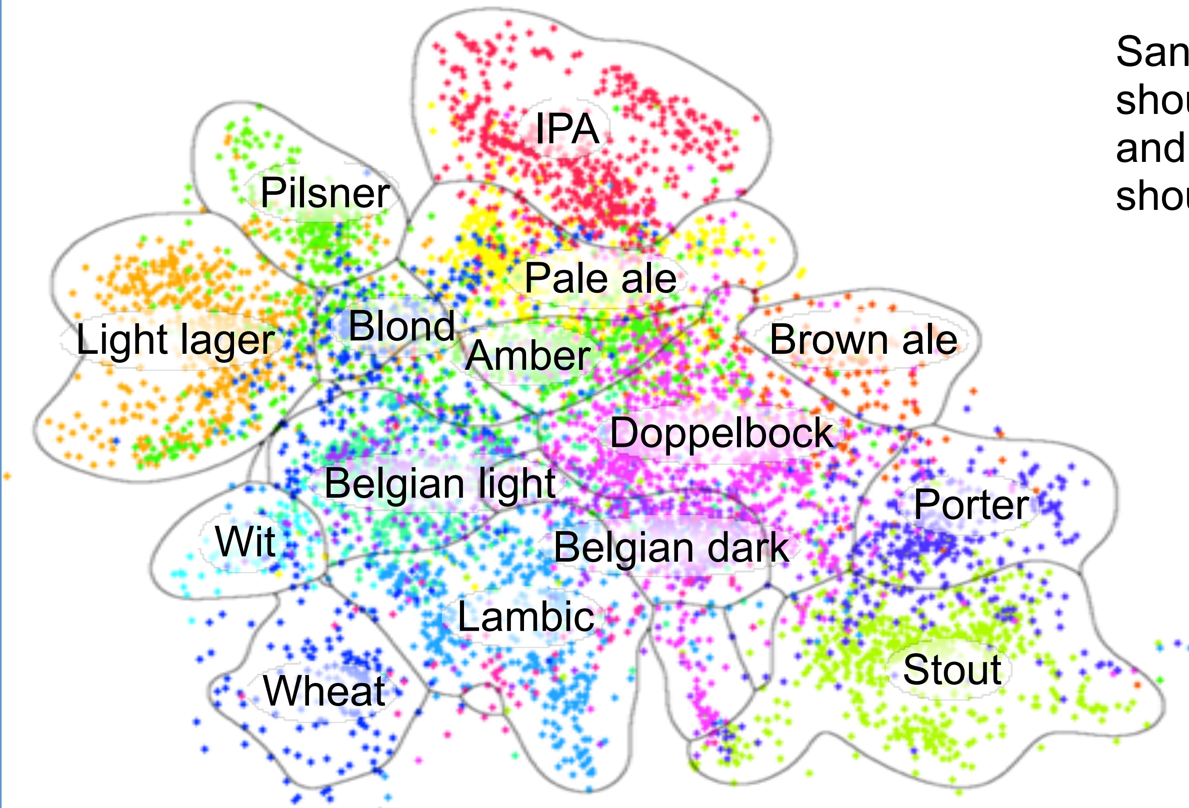
Get 100 nearest neighbors using cosine distance

Non-metric multidimensional scaling

Embedding in d dimensions

BeerMapper - Under the Hood

Algorithm requires feature representations of the beers $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$



Sanity check: styles should cluster together and similar styles should be close.

Reviews for each beer → Bag of Words weighted by TF*IDF → Get 100 nearest neighbors using cosine distance → Non-metric multidimensional scaling → Embedding in d dimensions

Other document modeling

Matrix factorization:

1. Construct word x document matrix of counts
2. Compute non-negative matrix factorization
3. Use factorization to represent documents
4. Cluster documents into topics

Also see latent Dirichlet factorization (LDA)