Announcements

- HW3 Due tonight
- HW4 posted
- No class Thursday (Thanksgiving)

Mixtures of Gaussians

Machine Learning – CSE546

Kevin Jamieson
University of Washington

November 20, 2016

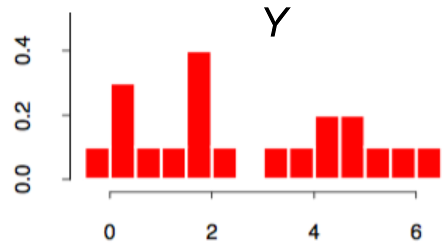
©Kevin Jamieson 2017

Mixture models

$$\begin{aligned} Y_1 &\sim N(\mu_1, \sigma_1^2), \\ Y_2 &\sim N(\mu_2, \sigma_2^2), \\ Y &= (1-\Delta) \cdot Y_1 + \Delta \cdot Y_2, \end{aligned}$$

$\Delta \in \{0,1\}$ with $\Pr(\Delta = 1) = \pi$

$$\theta = (\pi, \theta_1, \theta_2) = (\pi, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2)$$



$\mathbf{Z} = \{y_i\}_{i=1}^n$ is observed data

$\boldsymbol{\Delta} = \{\Delta_i\}_{i=1}^n$ is unobserved data

If $\phi_\theta(x)$ is Gaussian density with parameters $\theta = (\mu, \sigma^2)$ then

$$\ell(\theta; \mathbf{Z}, \boldsymbol{\Delta}) = \sum_{i=1}^n (1-\Delta_i) \log[(1-\pi)\phi_{\theta_1}(y_i)] + \Delta_i \log(\pi \phi_{\theta_2}(y_i)]$$

$$\gamma_i(\theta) = \mathbb{E}[\Delta_i | \theta, \mathbf{Z}] =$$

©Kevin Jamieson 2017

Mixture models

Algorithm 8.1 *EM Algorithm for Two-component Gaussian Mixture.*

1. Take initial guesses for the parameters $\hat{\mu}_1, \hat{\sigma}_1^2, \hat{\mu}_2, \hat{\sigma}_2^2, \hat{\pi}$ (see text).

2. *Expectation Step*: compute the responsibilities

$$\hat{\gamma}_i = \frac{\hat{\pi}\phi_{\hat{\theta}_2}(y_i)}{(1-\hat{\pi})\phi_{\hat{\theta}_1}(y_i) + \hat{\pi}\phi_{\hat{\theta}_2}(y_i)}, \quad i = 1, 2, \ldots, N. \tag{8.42}$$

3. *Maximization Step*: compute the weighted means and variances:

$$\hat{\mu}_1 = \frac{\sum_{i=1}^N (1-\hat{\gamma}_i) y_i}{\sum_{i=1}^N (1-\hat{\gamma}_i)}, \qquad \hat{\sigma}_1^2 = \frac{\sum_{i=1}^N (1-\hat{\gamma}_i)(y_i - \hat{\mu}_1)^2}{\sum_{i=1}^N (1-\hat{\gamma}_i)},$$

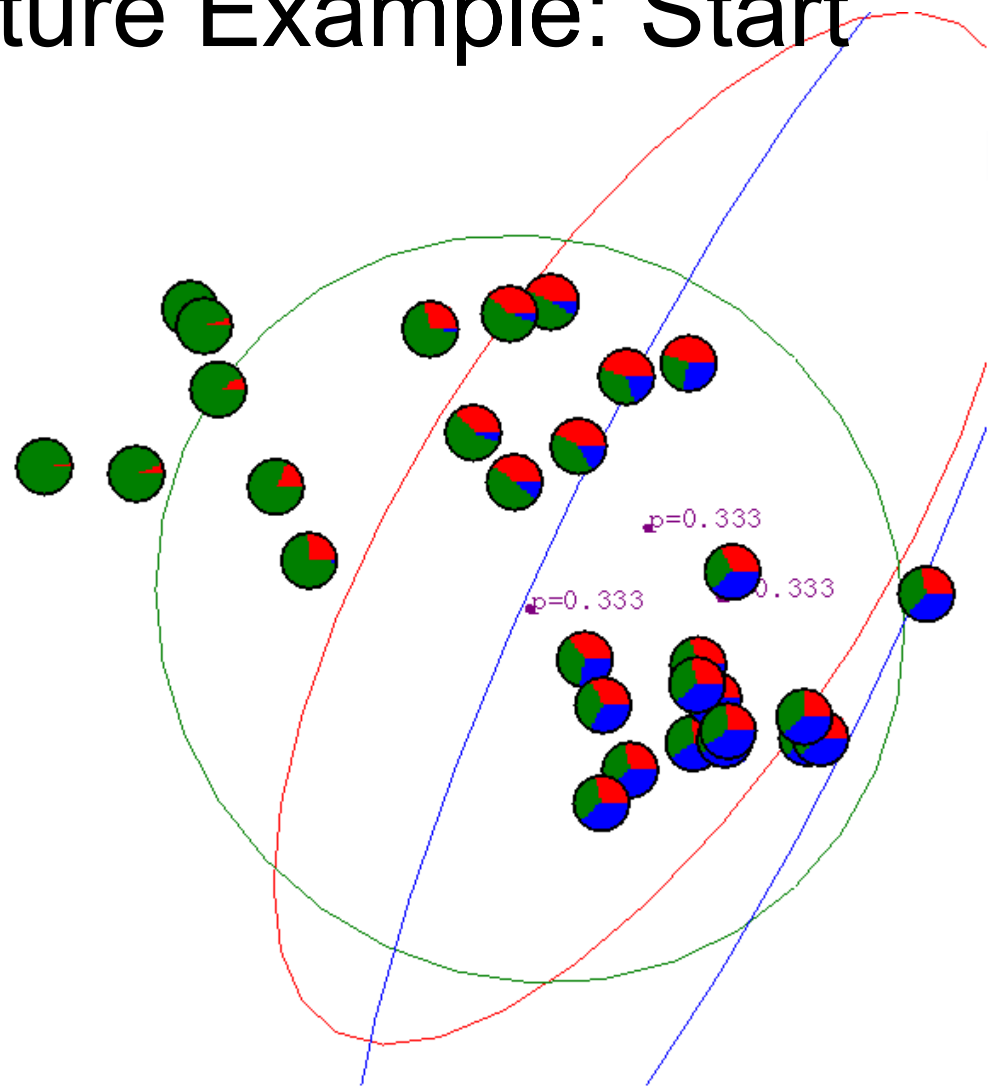
$$\hat{\mu}_2 = \frac{\sum_{i=1}^N \hat{\gamma}_i y_i}{\sum_{i=1}^N \hat{\gamma}_i}, \qquad \hat{\sigma}_2^2 = \frac{\sum_{i=1}^N \hat{\gamma}_i (y_i - \hat{\mu}_2)^2}{\sum_{i=1}^N \hat{\gamma}_i},$$

and the mixing probability $\hat{\pi} = \sum_{i=1}^N \hat{\gamma}_i / N$.

4. Iterate steps 2 and 3 until convergence.

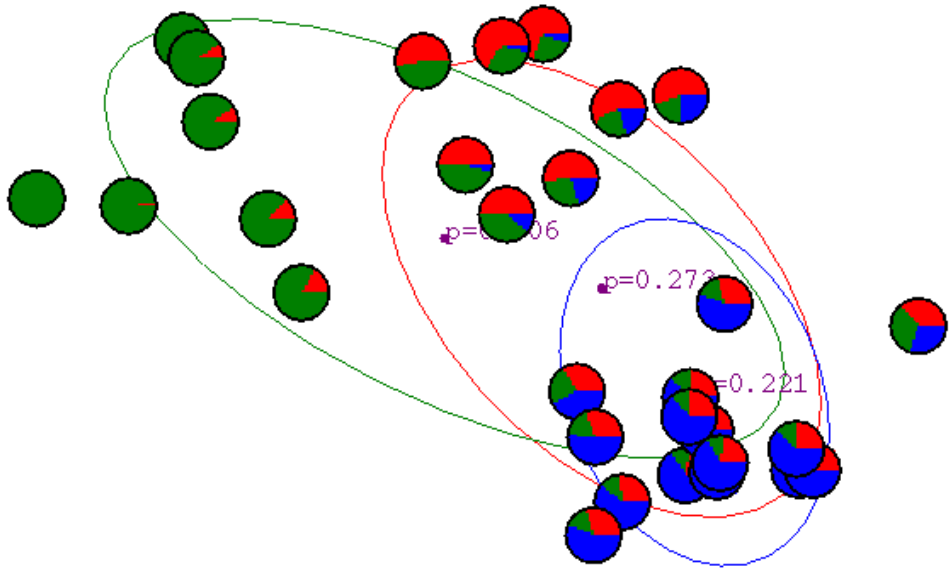
©Kevin Jamieson 2017

Gaussian Mixture Example: Start



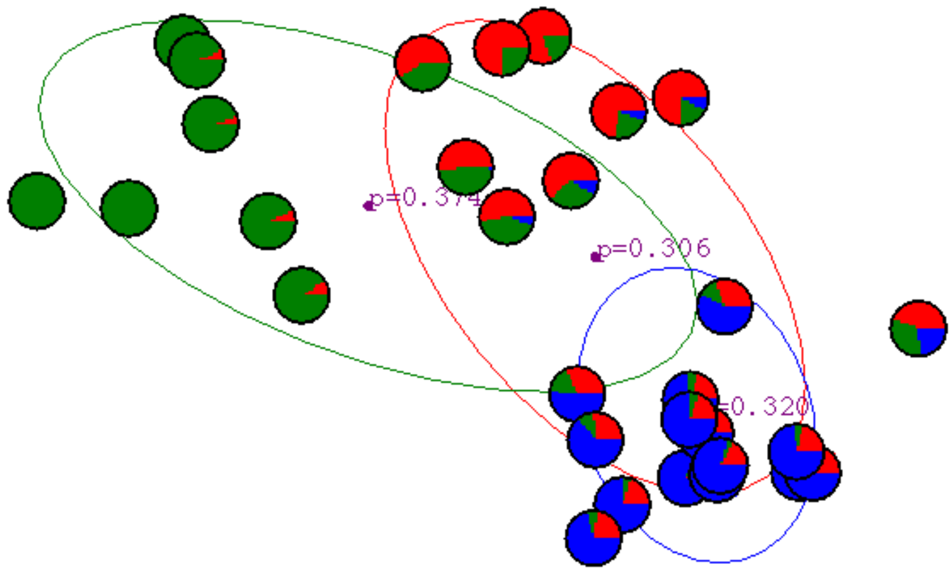
©Kevin Jamieson 2017

After first iteration

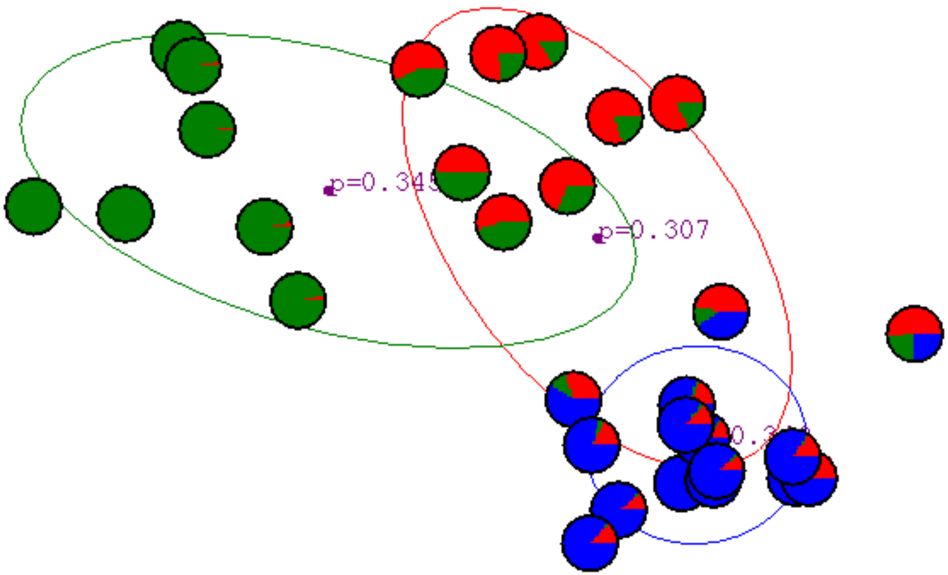


©Kevin Jamieson 2017

After 2nd iteration

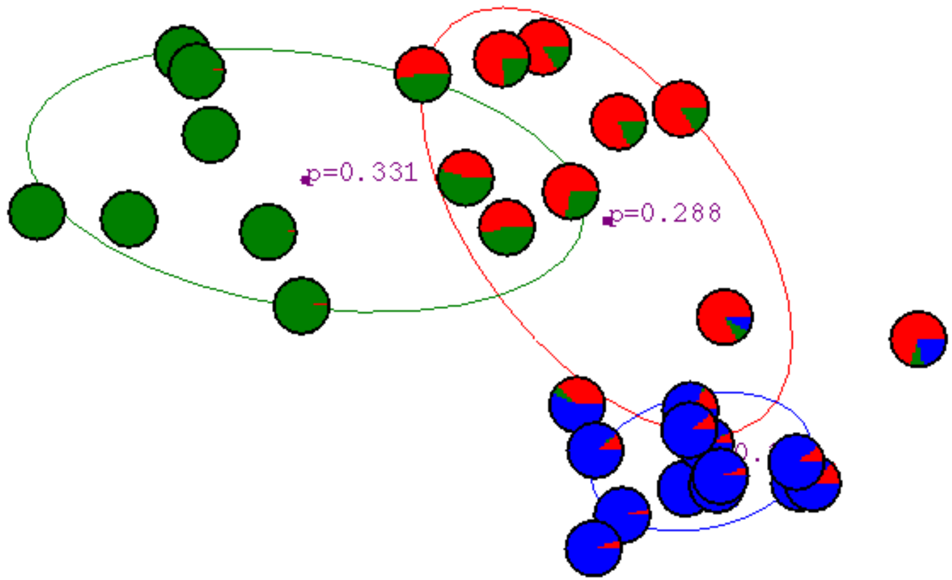


After 3rd iteration

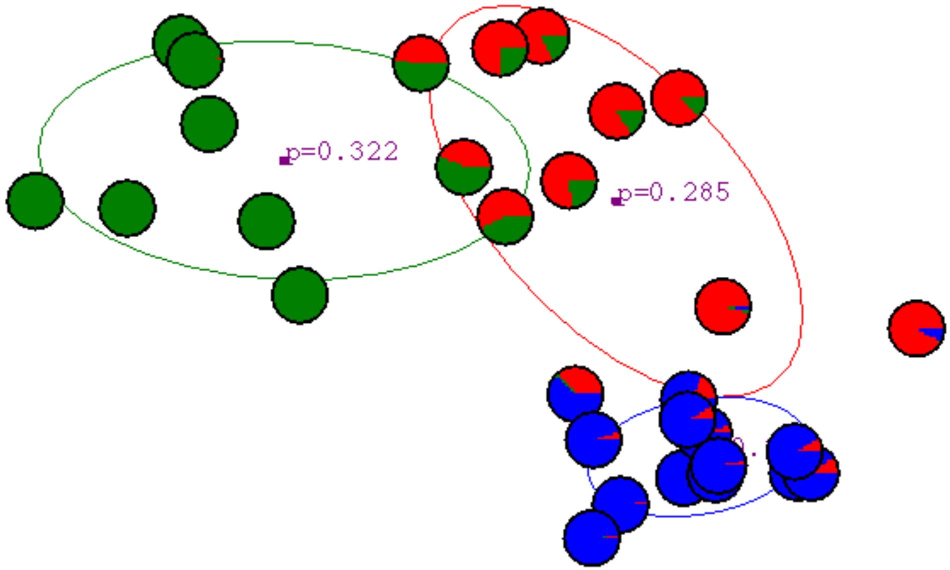


©Kevin Jamieson 2017

After 4th iteration

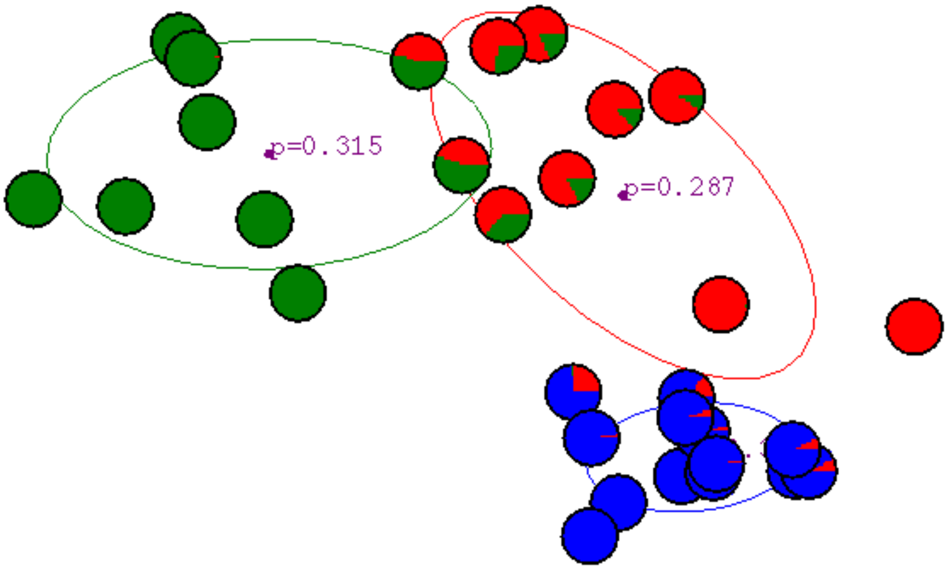


After 5th iteration



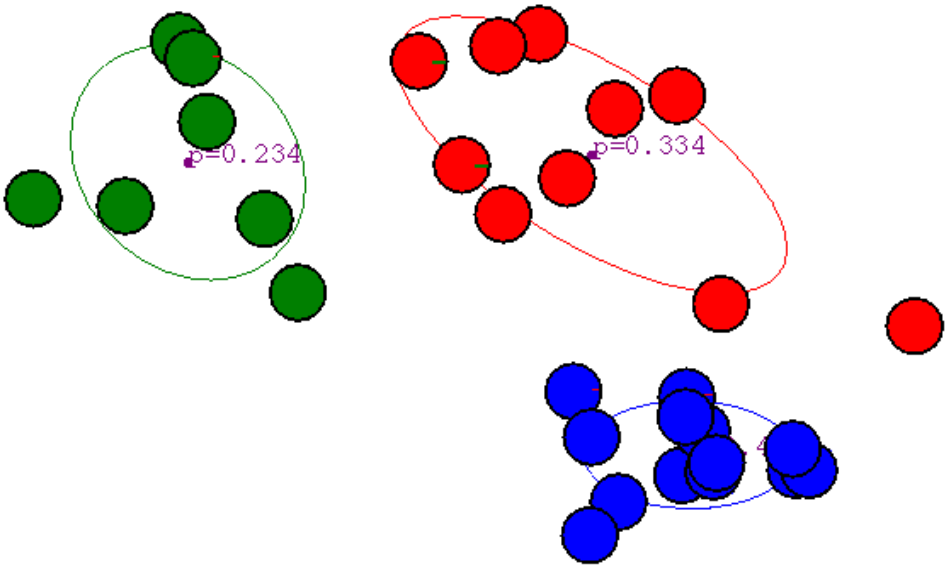
©Kevin Jamieson 2017

After 6th iteration



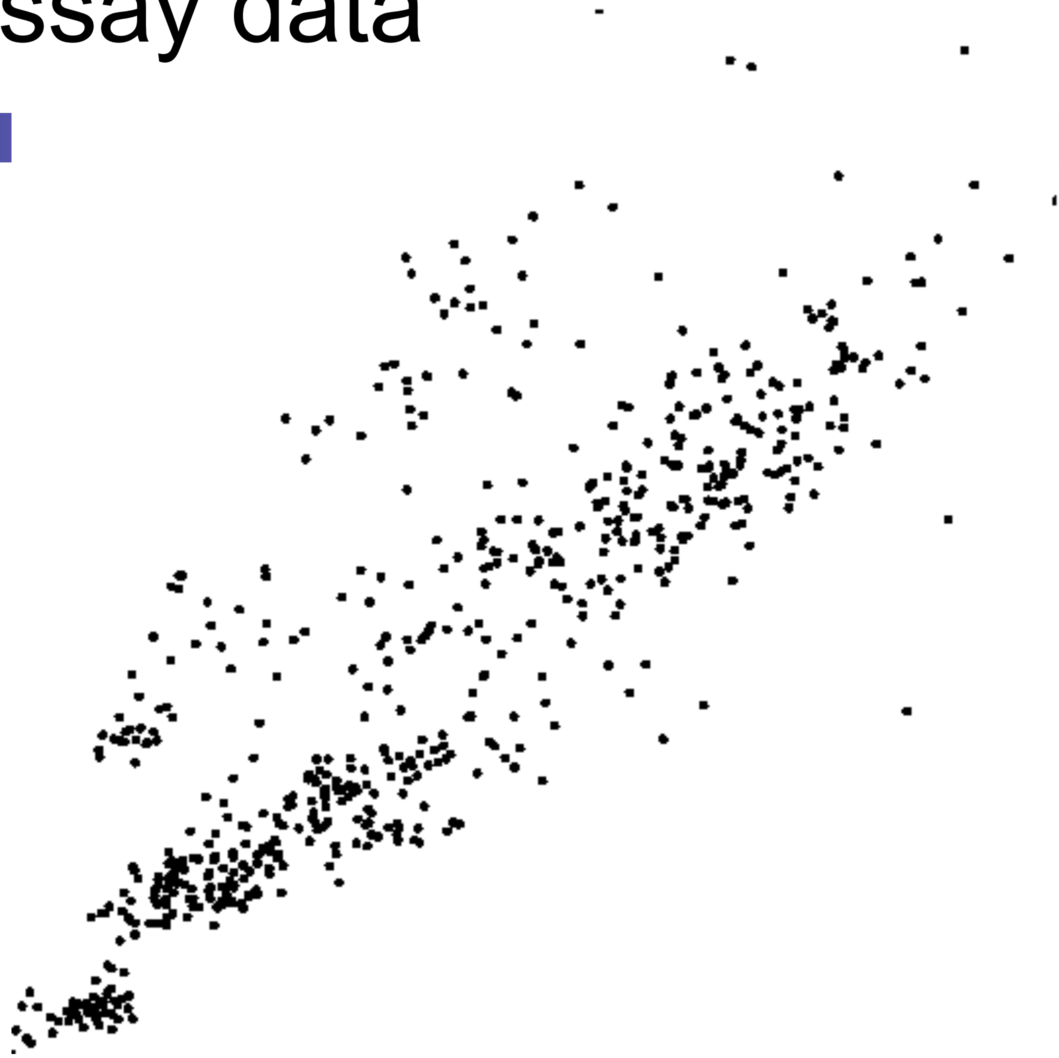
©Kevin Jamieson 2017

After 20th iteration



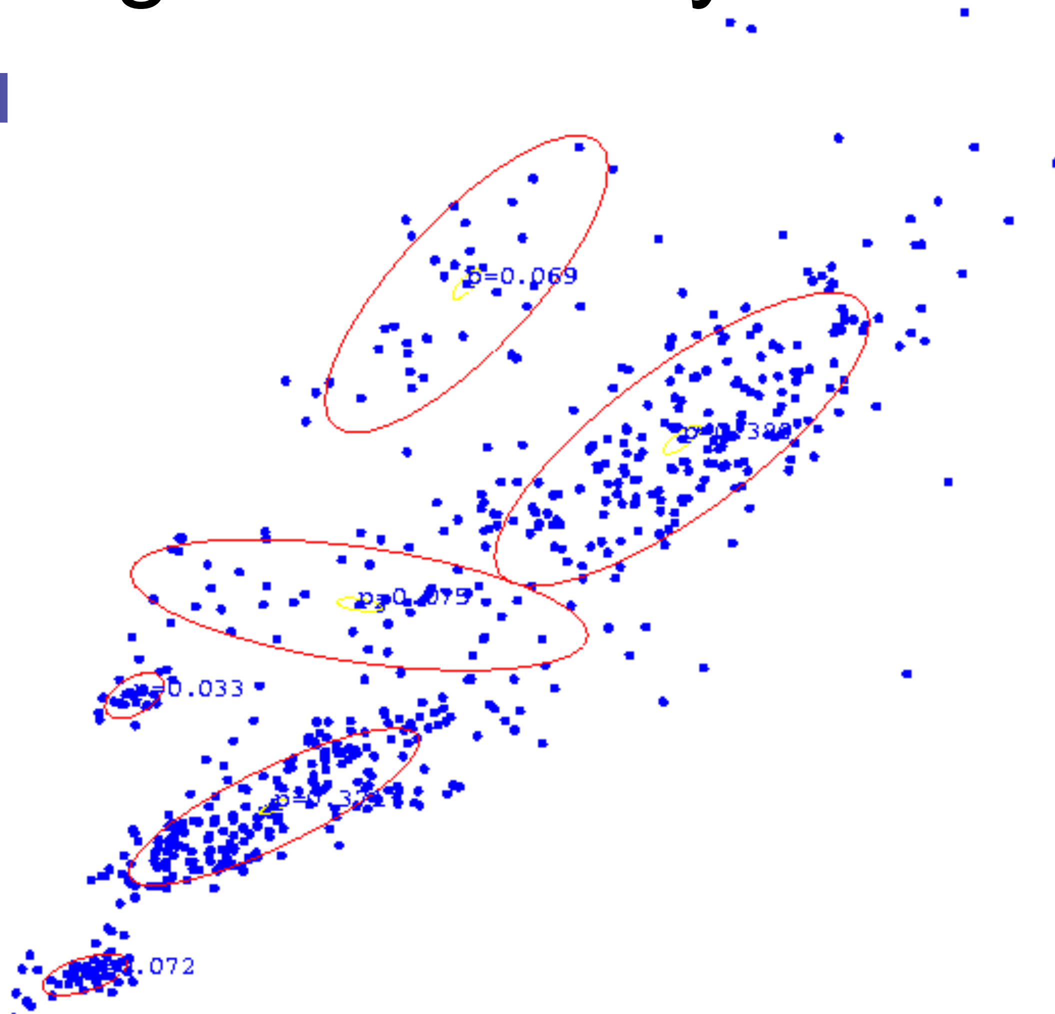
©Kevin Jamieson 2017

Some Bio Assay data



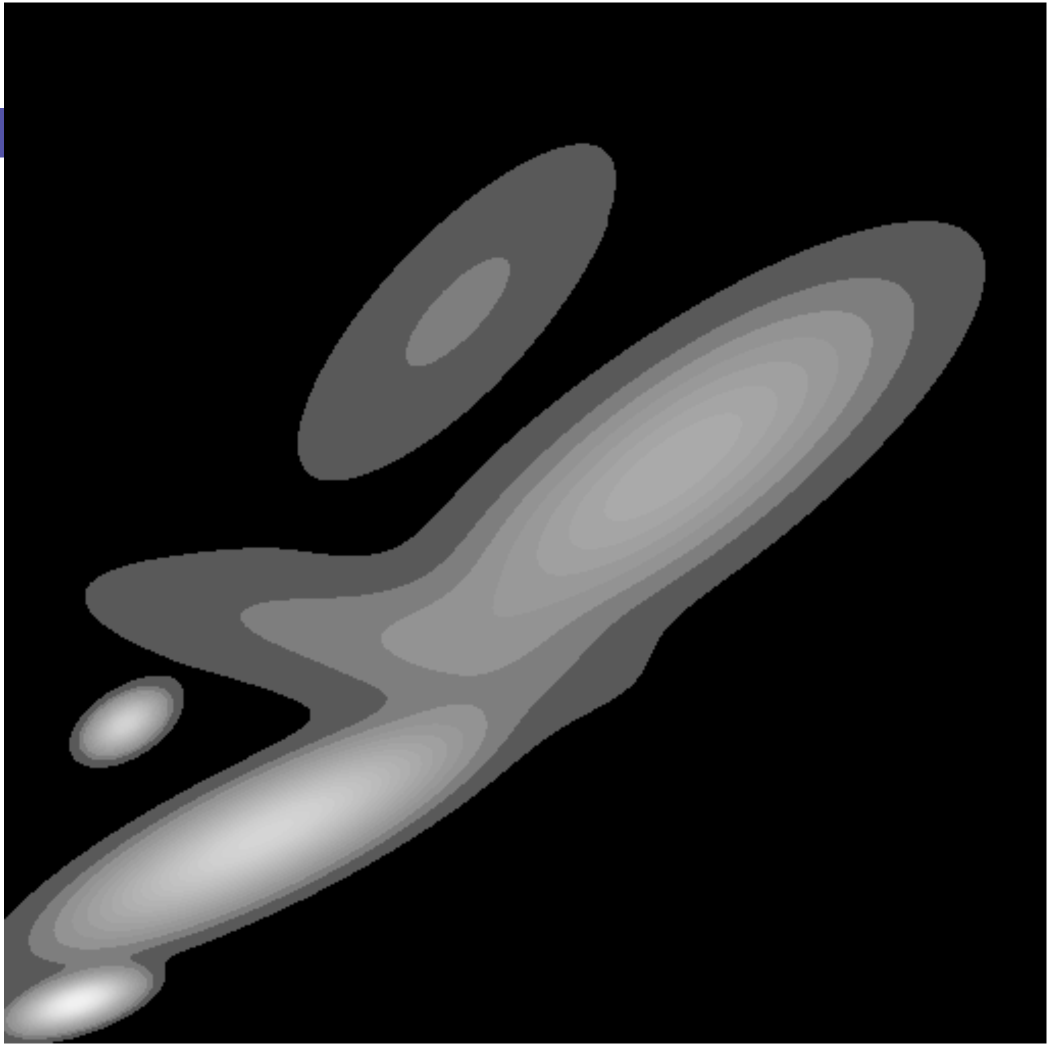
©Kevin Jamieson 2017

GMM clustering of the assay data



©Kevin Jamieson 2017

Resulting Density Estimator



©Kevin Jamieson 2017

Expectation Maximization Algorithm

Observe data $x_1, \ldots, x_n$ drawn from a distribution $p(\cdot|\theta_*)$ for some $\theta_* \in \Theta$

$$\widehat{\theta}_{MLE} = \arg\max_\theta \sum_{i=1}^n \log(p(x_i|\theta))$$

$$\sum_{i=1}^n \log(p(x_i|\theta)) = \sum_{i=1}^n \log\left(\sum_j p(x_i, z_i = j|\theta)\right)$$

(Introduce hidden data zi)

$$= \sum_{i=1}^n \log\left(\sum_j q_i(z_i = j|\theta') \frac{p(x_i, z_i = j|\theta)}{q_i(z_i = j|\theta')}\right)$$

(Introduce dummy distribution qi, variable θ')

$$\geq \sum_{i=1}^n \sum_j q_i(z_i = j|\theta') \log\left(\frac{p(x_i, z_i = j|\theta)}{q_i(z_i = j|\theta')}\right)$$

(Jensen's inequality, log() is concave)

$$= \sum_{i=1}^n \sum_j q_i(z_i = j|\theta') \log\left(p(x_i, z_i = j|\theta)\right) + \underline{\sum_{i=1}^n \sum_j q_i(z_i = j|\theta') \log\left(\frac{1}{q_i(z_i = j|\theta')}\right)}$$

Does not depend on θ!

Expectation Maximization Algorithm

Observe data $\mathbf{X} = [x_1, \ldots, x_n]$ drawn from a distribution $p(\cdot|\theta_*)$ for some $\theta_* \in \Theta$

$$\widehat{\theta}_{MLE} = \arg\max_{\theta} \sum_{i=1}^{n} \log(p(x_i|\theta))$$

$$\boxed{\sum_{i=1}^{n} \log(p(x_i|\theta)) \geq \sum_{i=1}^{n} \sum_{j} q_i(z_i = j|\theta') \log\left(p(x_i, z_i = j|\theta)\right)}$$

True for *any* choice of θ' and distribution $q_i(z_i = j|\theta')$

Set $q_i(z_i = j|\theta') = p(z_i = j|\theta', \mathbf{X})$

©Kevin Jamieson 2017

Expectation Maximization Algorithm

Observe data $x_1, \ldots, x_n$ drawn from a distribution $p(\cdot|\theta_*)$ for some $\theta_* \in \Theta$

$$\widehat{\theta}_{MLE} = \arg\max_\theta \sum_{i=1}^n \log(p(x_i|\theta))$$

$$\boxed{\sum_{i=1}^n \log(p(x_i|\theta)) \geq \sum_{i=1}^n \sum_j p(z_i = j|\theta', \mathbf{X}) \log\left(p(x_i, z_i = j|\theta)\right) =: Q(\theta, \theta')}$$

Initial guess for $\theta^{(0)}$, for each step k:

E-step: compute $$Q(\theta, \theta^{(k)}) = \sum_{i=1}^n \mathbb{E}_{z_i}\left[\log\left(p(x_i, z_i|\theta)\right) \middle| \theta^{(k)}, \mathbf{X}\right]$$

M-step: find $$\theta^{(k+1)} = \arg\max_\theta Q(\theta, \theta^{(k)})$$

©Kevin Jamieson 2017

Expectation Maximization Algorithm

Initial guess for $\theta^{(0)}$, for each step k:

E-step: compute $$Q(\theta, \theta^{(k)}) = \sum_{i=1}^{n} \mathbb{E}_{z_i}\left[\log\left(p(x_i, z_i|\theta)\right) \Big| \theta^{(k)}, \mathbf{X}\right]$$

M-step: find $$\theta^{(k+1)} = \arg\max_{\theta} Q(\theta, \theta^{(k)})$$

Example: Observe $x_1, \ldots, x_n \sim (1-\pi)\mathcal{N}(\mu_1, \sigma_1^2) + \pi\mathcal{N}(\mu_2, \sigma_2^2)$

$z_i = j$ if i is in mixture component j for $j \in \{1, 2\}$ $\qquad \theta = (\pi, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2)$

$$\begin{aligned}
&\mathbb{E}_{z_i}[\log(p(x_i, z_i|\theta)|\theta^{(k)}, \mathbf{X}] \\
&= p(z_i = 1|\theta^{(k)}, x_i)\log\left(p(x_i, z_i = 1|\theta)\right) + p(z_i = 2|\theta^{(k)}, x_i)\log\left(p(x_i, z_i = 2|\theta)\right) \\
&= p(z_i = 1|\theta^{(k)}, x_i)\log\left(p(x_i|z_i = 1, \theta)p(z_i = 1|\theta)\right) + p(z_i = 2|\theta^{(k)}, x_i)\log\left(p(x_i|z_i = 2, \theta)p(z_i = 2|\theta)\right) \\
&= \frac{\phi(x_i|\mu_1^{(k)}, \sigma_1^{2(k)})}{\phi(x_i|\mu_1^{(k)}, \sigma_1^{2(k)}) + \phi(x_i|\mu_2^{(k)}, \sigma_2^{2(k)})}\log\left(\phi(x_i|\mu_1, \sigma_1^2)(1-\pi)\right) + \frac{\phi(x_i|\mu_2^{(k)}, \sigma_2^{2(k)})}{\phi(x_i|\mu_1^{(k)}, \sigma_1^{2(k)}) + \phi(x_i|\mu_2^{(k)}, \sigma_2^{2(k)})}\log\left(\phi(x_i|\mu_2, \sigma_2^2)\pi\right)
\end{aligned}$$

©Kevin Jamieson 2017

Expectation Maximization Algorithm

- EM used to solve **Latent Factor Models**

- Also used to solve **missing data** problems
- Also known as Baum-Welch algorithm for Hidden Markov Models
- In general, EM is **non-convex** so it can get stuck in local minima.

©Kevin Jamieson 2017

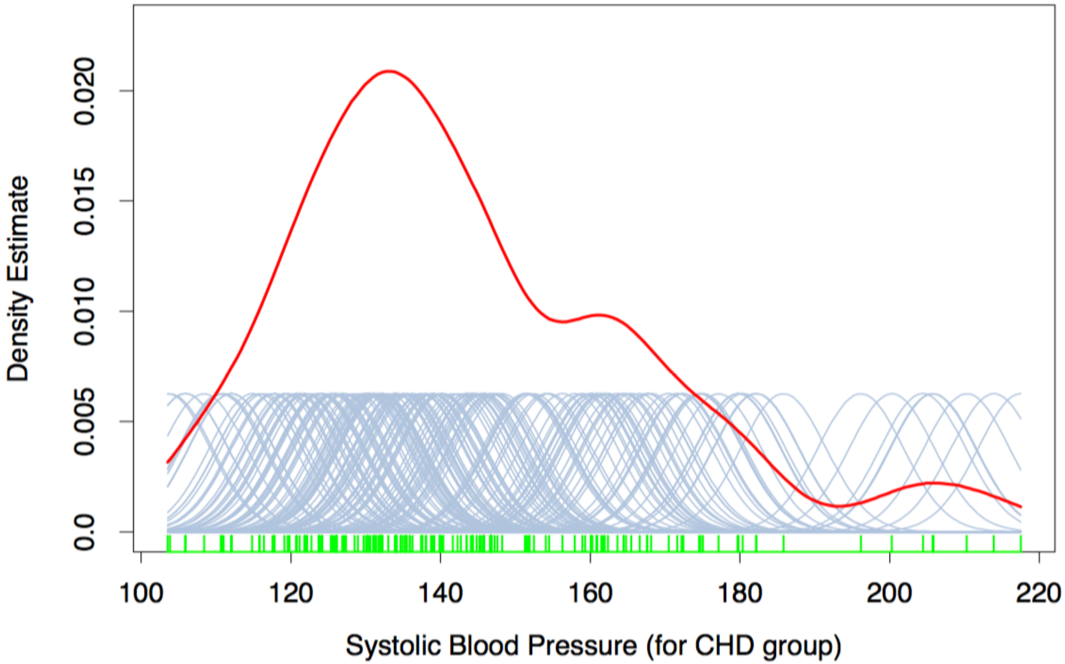
Density Estimation

Machine Learning – CSE546
Kevin Jamieson
University of Washington

November 20, 2016

©Kevin Jamieson 2017

Kernel Density Estimation

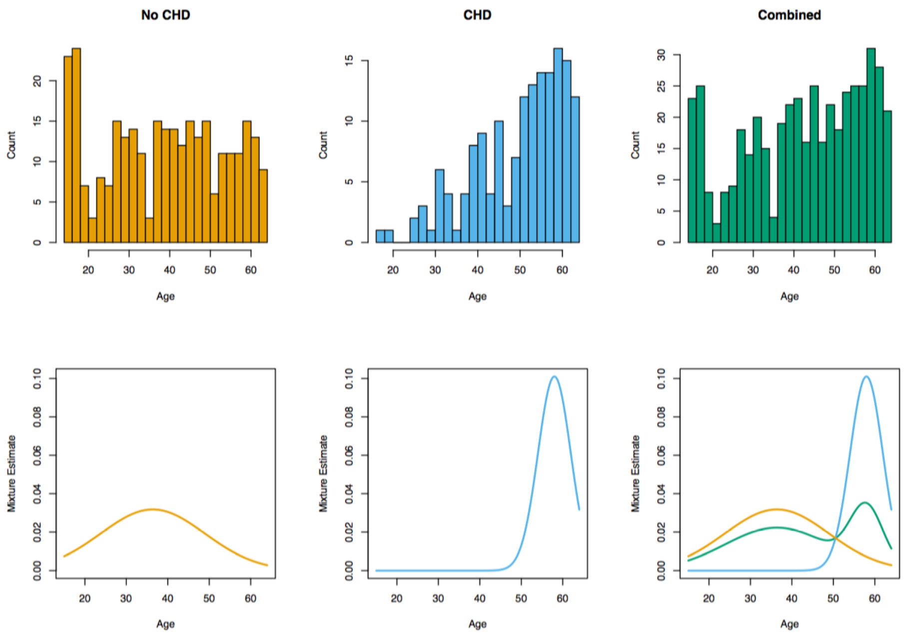


$$f(x) = \sum_{m=1}^{M} \alpha_m \phi(x; \mu_m, \Sigma_m)$$

A very “lazy” GMM

©Kevin Jamieson 2017

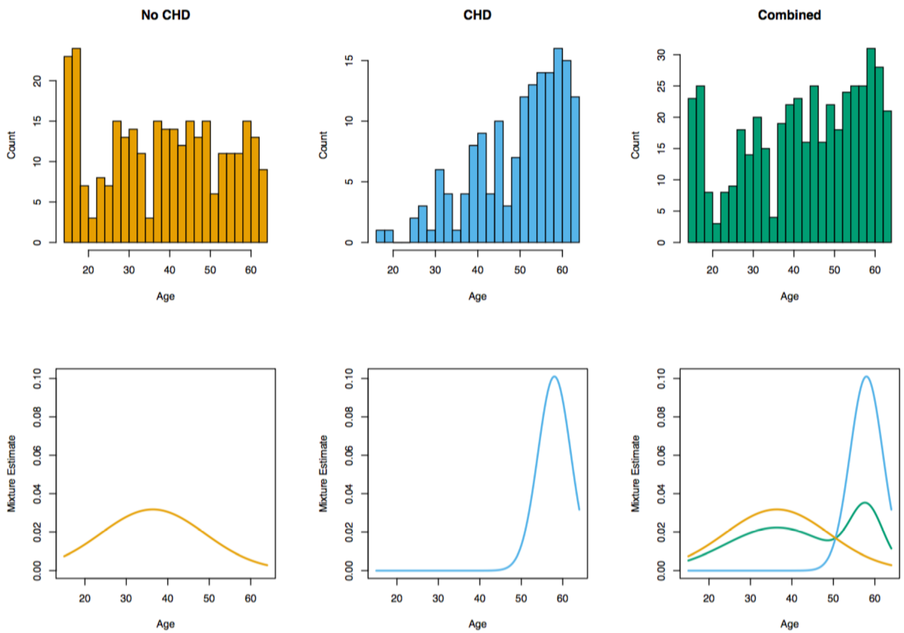
Kernel Density Estimation



$$f(x) = \sum_{m=1}^{M} \alpha_m \phi(x; \mu_m, \Sigma_m)$$

©Kevin Jamieson 2017

Kernel Density Estimation



What is the Bayes optimal classification rule?

$$\hat{r}_{im} = \frac{\hat{\alpha}_m \phi(x_i; \hat{\mu}_m, \hat{\Sigma}_m)}{\sum_{k=1}^{M} \hat{\alpha}_k \phi(x_i; \hat{\mu}_k, \hat{\Sigma}_k)}$$

$$f(x) = \sum_{m=1}^{M} \alpha_m \phi(x; \mu_m, \Sigma_m)$$

Predict $\arg\max_m \widehat{r}_{im}$

©Kevin Jamieson 2017

Generative vs Discriminative

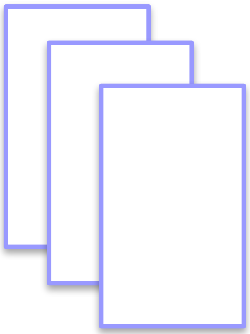
©Kevin Jamieson 2017

Basic Text Modeling

Machine Learning – CSE4546
Kevin Jamieson
University of Washington

November 20, 2017

Bag of Words



n documents/articles with lots of text

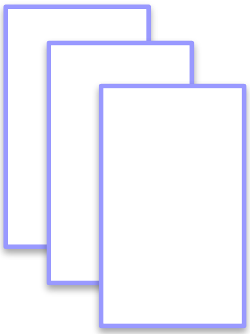
Questions:
- How to get a feature representation of each article?
- How to cluster documents into topics?

Bag of words model:

*i*th document: $x_i \in \mathbb{R}^D$

$$x_{i,j} = \text{ proportion of times } j\text{th word occurred in } i\text{th document}$$

Bag of Words



n documents/articles with lots of text

Questions:
- How to get a feature representation of each article?
- How to cluster documents into topics?

Bag of words model:

*i**th* document: $x_i \in \mathbb{R}^D$

$$x_{i,j} = \text{ proportion of times } j\text{th word occurred in } i\text{th document}$$

Given vectors, run k-means or Gaussian mixture model to find k clusters/topics

Nonnegative matrix factorization (NMF)

$A \in \mathbb{R}^{m \times n}$ $\quad A_{i,j} = \text{frequency of } j\text{th word in document } i$

Nonnegative Matrix factorization:

$$\min_{W \in \mathbb{R}_+^{m \times d}, H \in \mathbb{R}_+^{n \times d}} \|A - WH^T\|_F^2$$

d is number of topics

Also see latent Dirichlet factorization (LDA)

Nonnegative matrix factorization (NMF)

$A \in \mathbb{R}^{m \times n}$ $\quad A_{i,j} = \text{frequency of } j\text{th word in document } i$

Nonnegative Matrix factorization:

$$\min_{W \in \mathbb{R}_+^{m \times d},\, H \in \mathbb{R}_+^{n \times d}} \|A - WH^T\|_F^2$$

d is number of topics

Each column of H represents a cluster of a topic,
Each row W is some weights a combination of topics

Also see latent Dirichlet factorization (LDA)

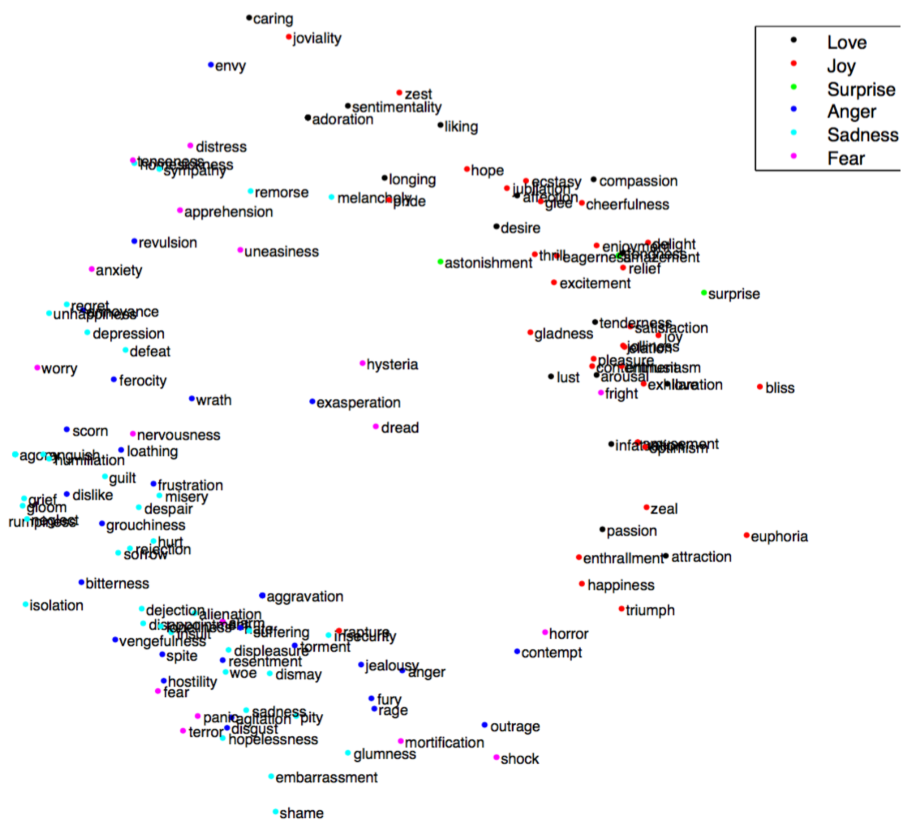
Word embeddings, word2vec

Previous section presented methods to **embed documents** into a latent space

Alternatively, we can **embed words** into a latent space

This embedding came from directly querying for relationships.

word2vec is a popular unsupervised learning approach that just uses a text corpus (e.g. [nytimes.com]())

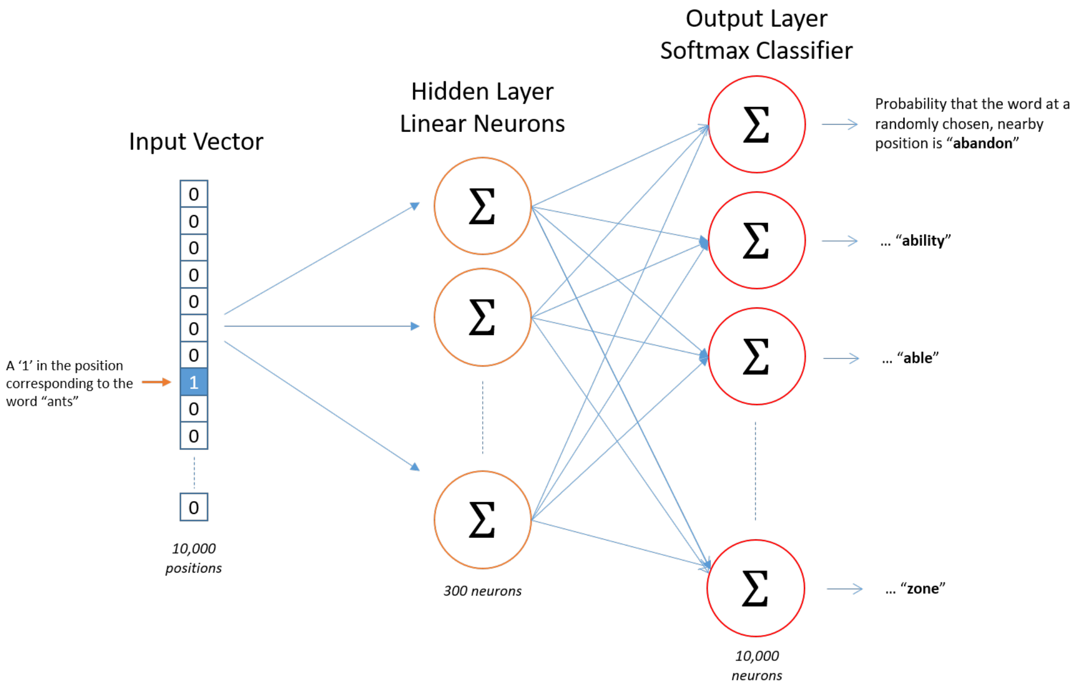


Word embeddings, word2vec

Source Text	Training Samples
The quick brown fox jumps over the lazy dog.	(the, quick) (the, brown)
The quick brown fox jumps over the lazy dog.	(quick, the) (quick, brown) (quick, fox)
The quick brown fox jumps over the lazy dog.	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
The quick brown fox jumps over the lazy dog.	(fox, quick) (fox, brown) (fox, jumps) (fox, over)

slide: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

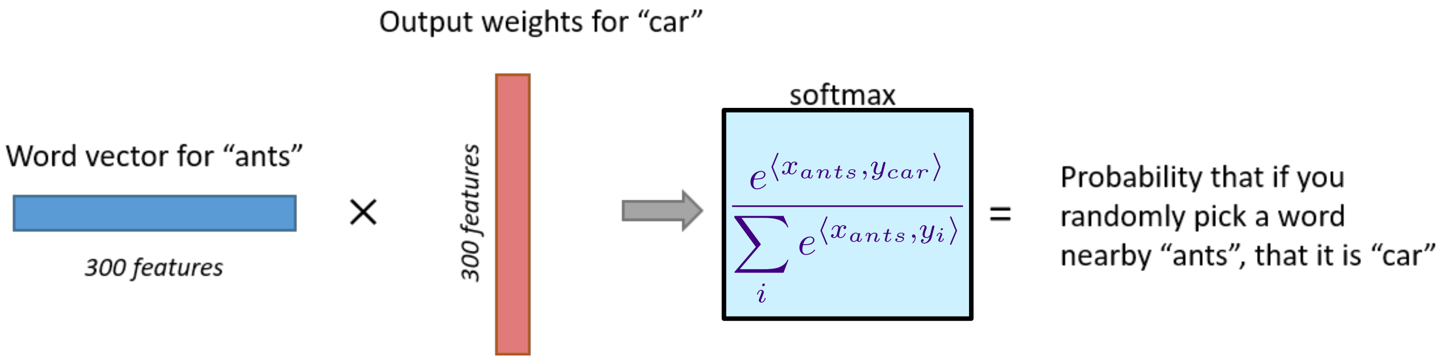
Word embeddings, word2vec



Training neural network to predict co-occuring words. Use first layer weights as embedding, throw out output layer

slide: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Word embeddings, word2vec

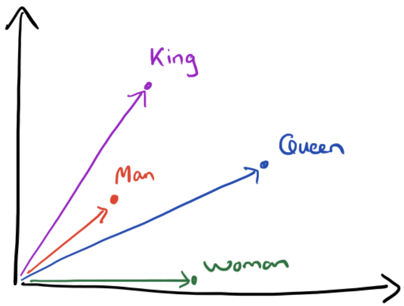


Training neural network to predict co-occuring words. Use first layer weights as embedding, throw out output layer

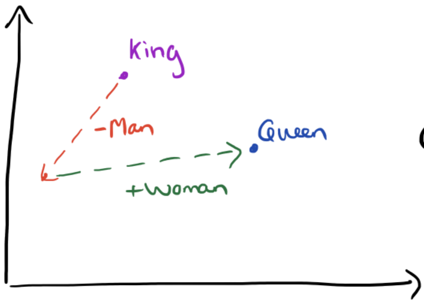
slide: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

word2vec outputs

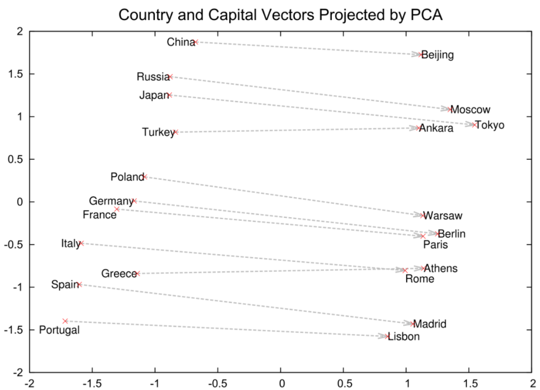
king - man + woman = queen



Word Vectors



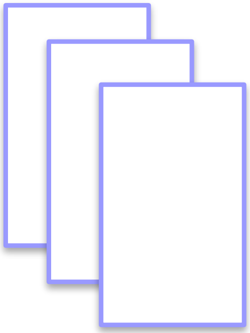
Vector Composition



country - capital

slide: https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/

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 $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$

Two Hearted Ale - Weighted Bag of Words:

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:

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

Two Hearted Ale - Nearest Neighbors:

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

$||x_k - x_i|| < ||x_k - x_j||$ 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.

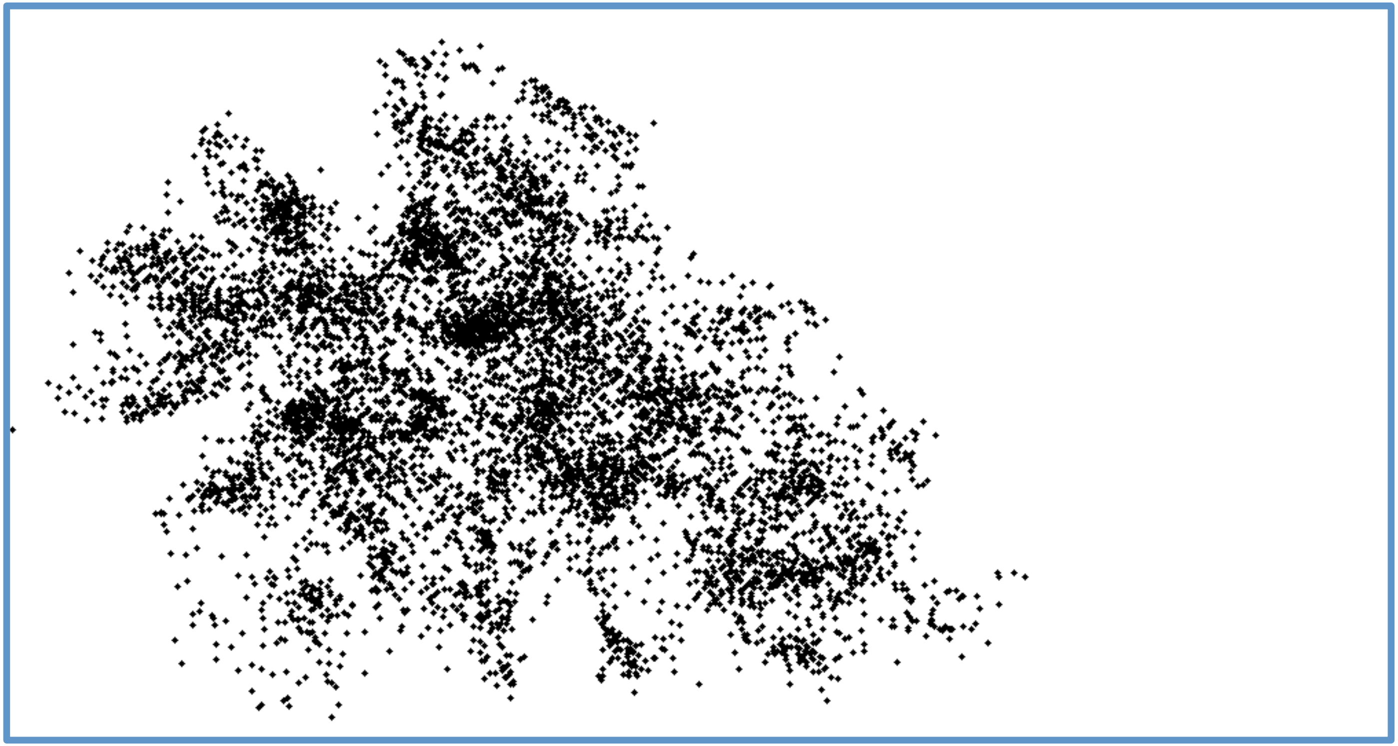
(20 minutes on my laptop) (d=2,err=6%) (d=3,err=4%)

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

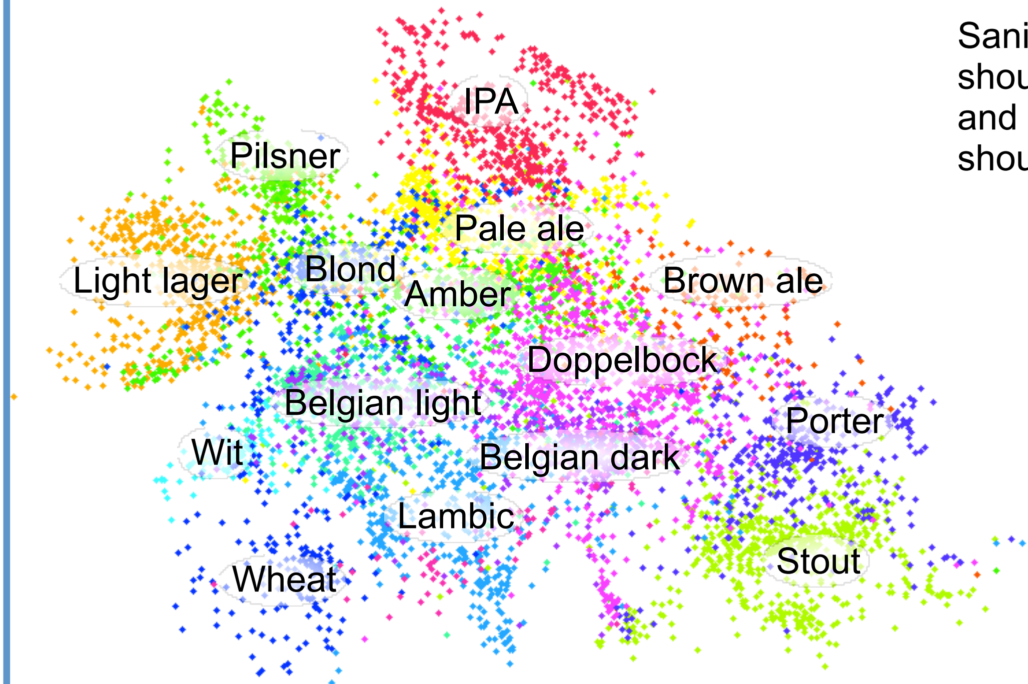
Get 100 nearest neighbors using cosine distance

Non-metric multidimensional scaling

Embedding in d dimensions

BeerMapper - Under the Hood

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

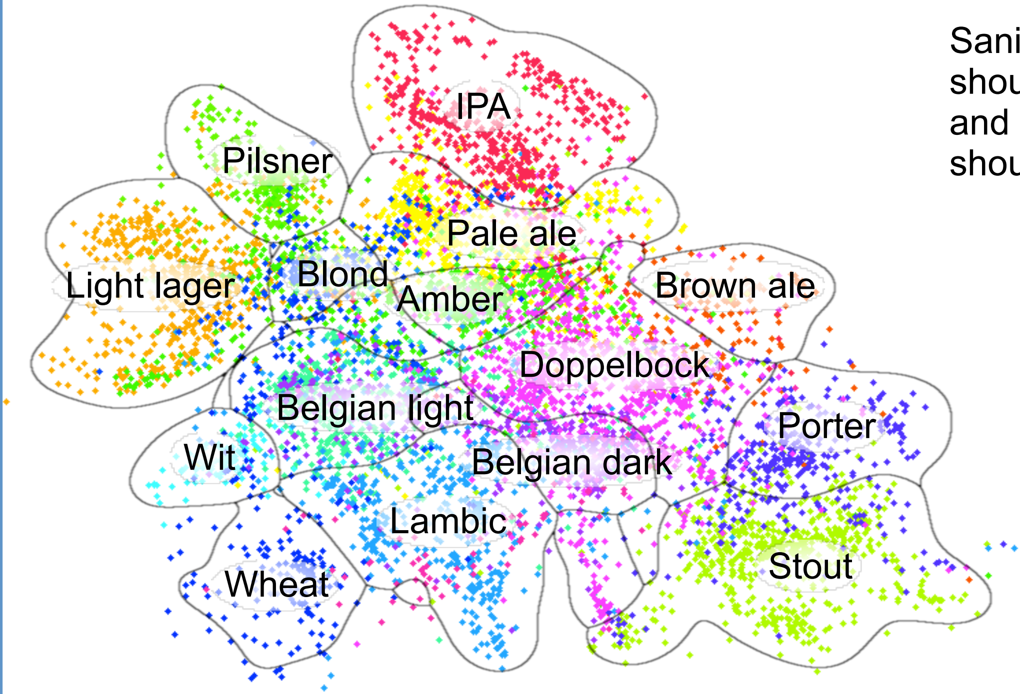


Sanity check: styles should cluster together and similar styles should be close.

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

Feature generation for images

Machine Learning – CSE4546

Kevin Jamieson
University of Washington

November 20, 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)

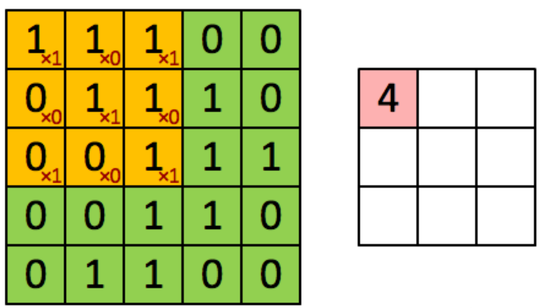
$$(I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

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
$I * K$

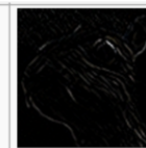
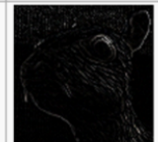
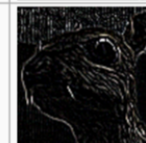
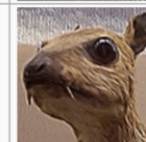
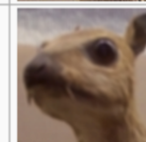
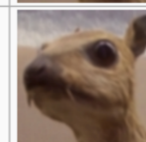
Slide credit: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Convolution of images

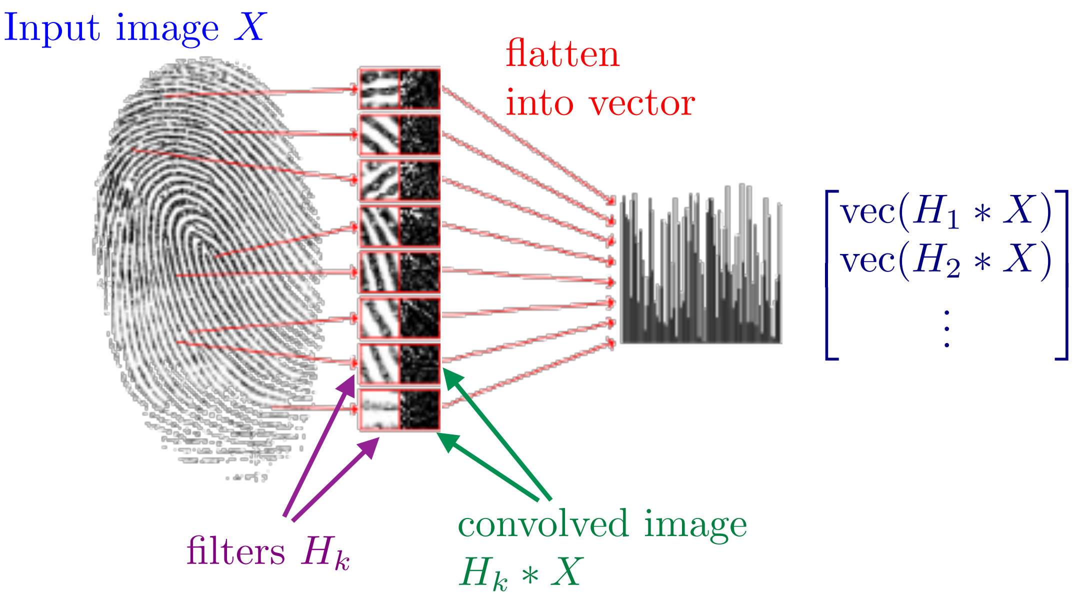
(Note to EEs: deep learning uses the word “convolution” to mean what is usually known as “cross-correlation”, e.g., neither signal is flipped)

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

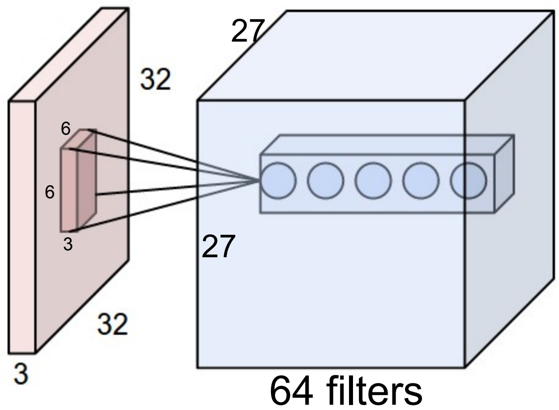
Image I

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)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

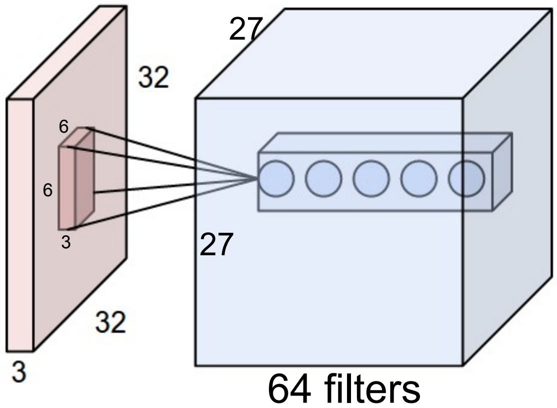
Convolution of images



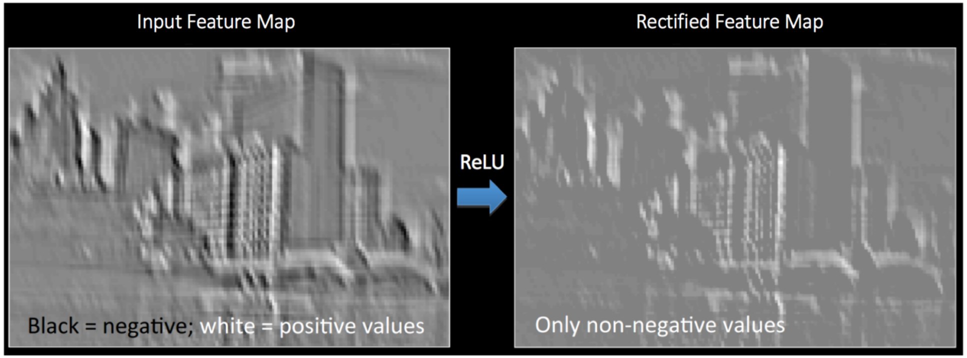
Stacking convolved images



Stacking convolved images



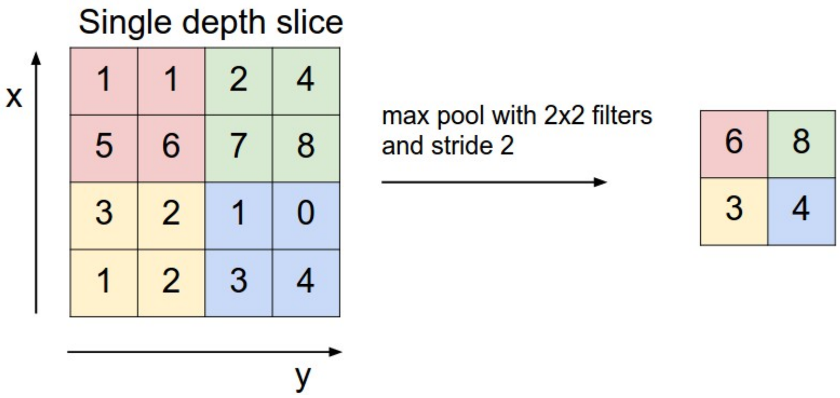
Apply Non-linearity to the output of each layer, Here: ReLu (rectified linear unit)

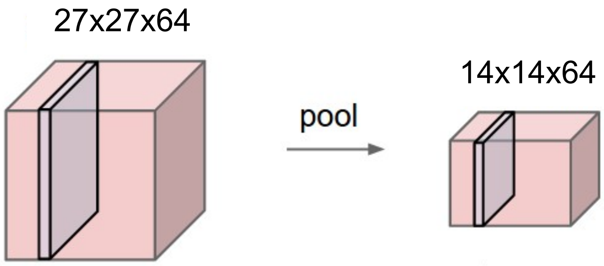


Other choices: sigmoid, arctan

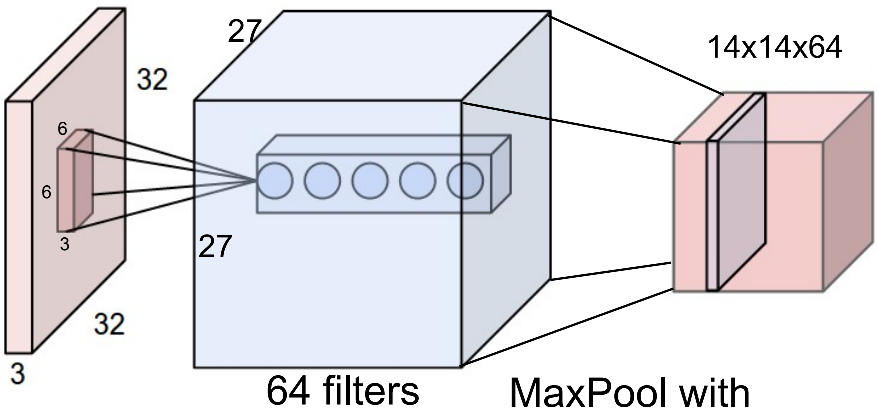
Pooling

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





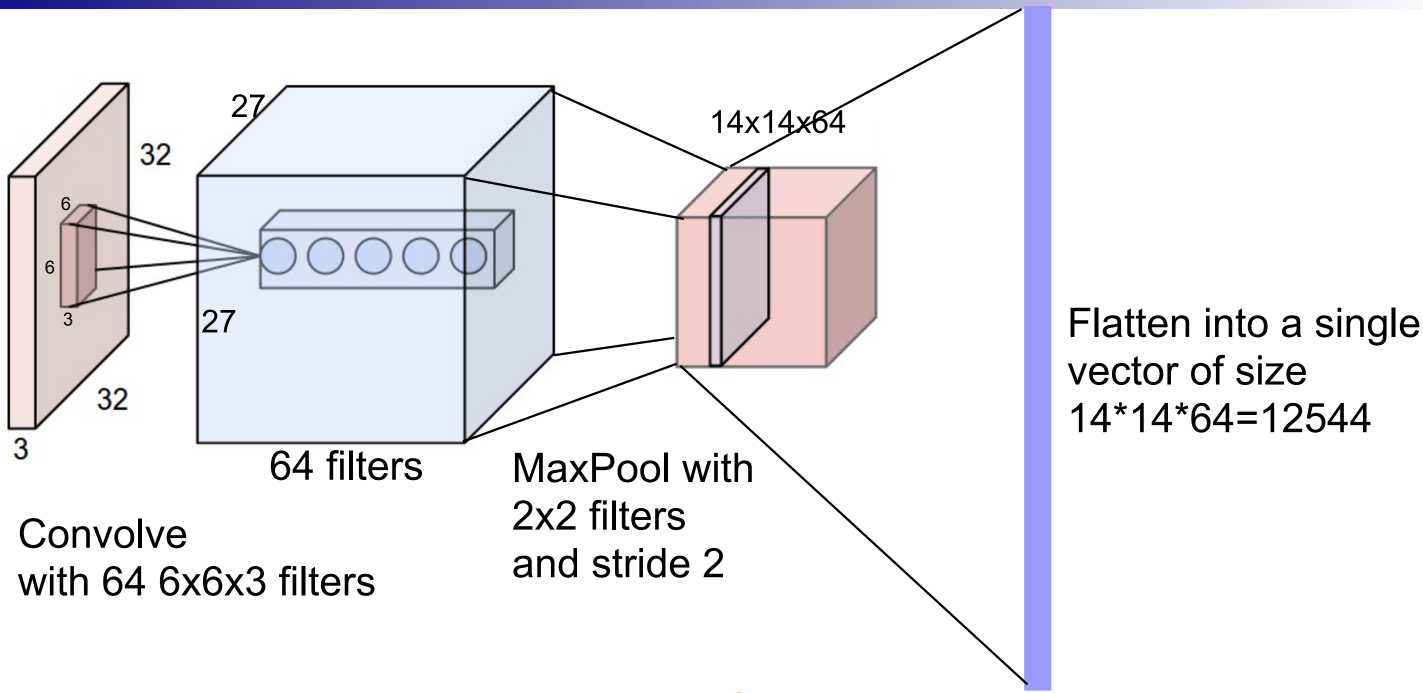
Pooling Convolution layer



Convolve
with 64 6x6x3 filters

MaxPool with
2x2 filters
and stride 2

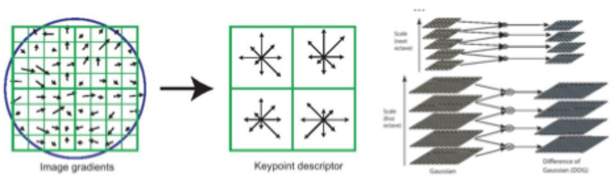
Full feature pipeline



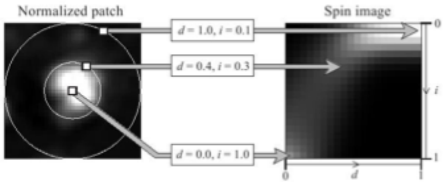
How do we choose all the hyperparameters?

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

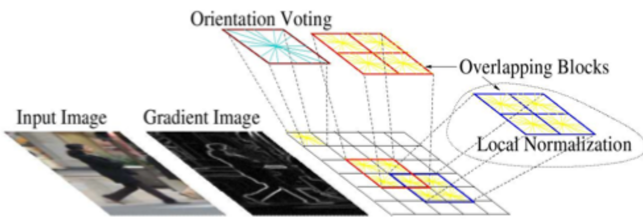
Some hand-created image features



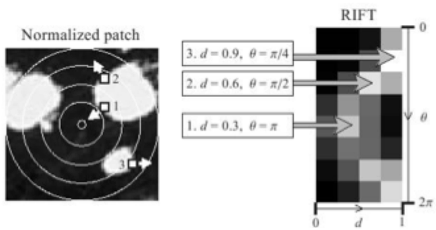
SIFT



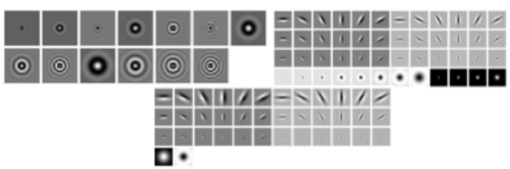
Spin Image



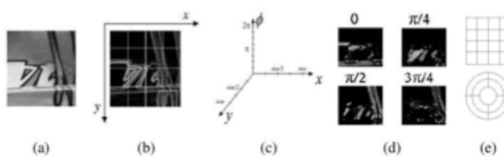
HoG



RIFT



Texton



GLOH

Slide from Honglak Lee

ML Street Fight

Machine Learning – CSE546
Kevin Jamieson
University of Washington

November 20, 2017

©Kevin Jamieson 2017

Mini case study

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

Methods of standardization:

Mini case study

Dealing with class imbalance:

Mini case study

Dealing with outliers:

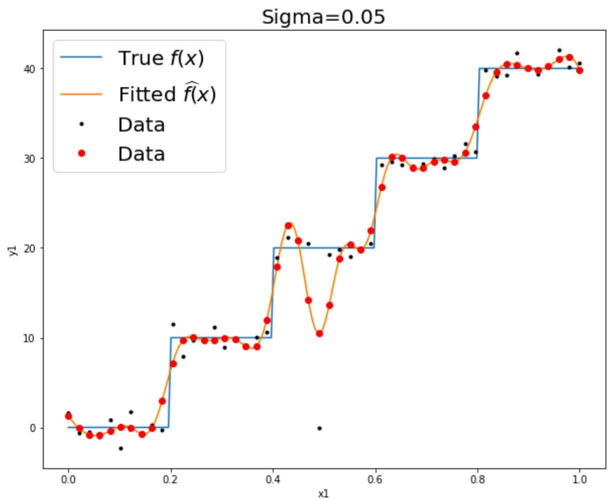
Mini case study

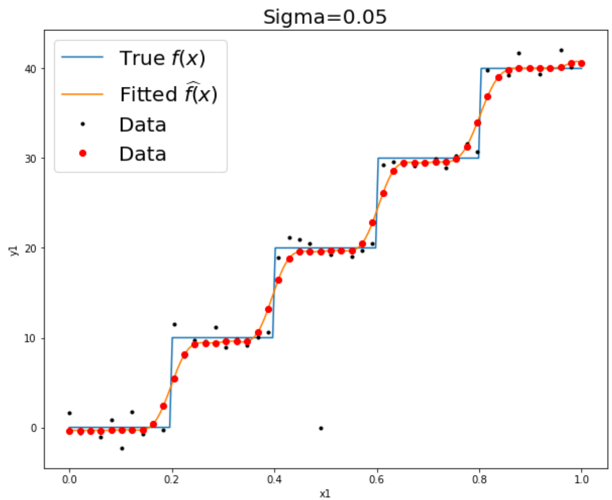
Dealing with outliers:

$$\ell_{huber}(z) = \begin{cases} \frac{1}{2}z^2 & \text{if } |z| \leq 1 \\ |z| - \frac{1}{2} & \text{otherwise} \end{cases}$$

$$\arg\min_{\alpha} \sum_{i=1}^{n} \left(\sum_j k(x_i, x_j)\alpha_j - y_i \right)^2 + \lambda \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j)$$

$$\arg\min_{\alpha} \sum_{i=1}^{n} \ell_{huber} \left(\sum_j k(x_i, x_j)\alpha_j - y_i \right) + \lambda \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j)$$





Mini case study

Dealing with hyperparameters:

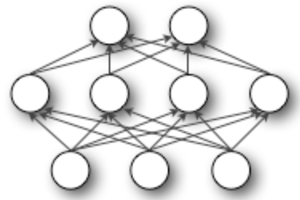
Hyperparameter Optimization

Machine Learning – CSE546

Kevin Jamieson
University of Washington

November 20, 2017

©Kevin Jamieson 2017



$N_{out} = 10$

N_{hid}

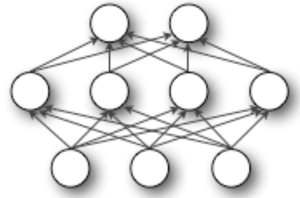
$N_{in} = 784$

hyperparameters

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

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

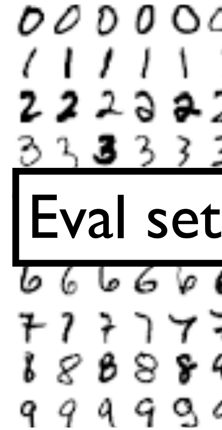
hidden nodes $N_{hid} \in [10^1, 10^3]$



$N_{out} = 10$

N_{hid}

$N_{in} = 784$



$\widehat{f}$

Hyperparameters

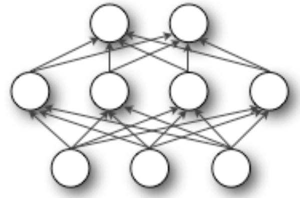
$(10^{-1.6}, 10^{-2.4}, 10^{1.7})$

hyperparameters

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

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

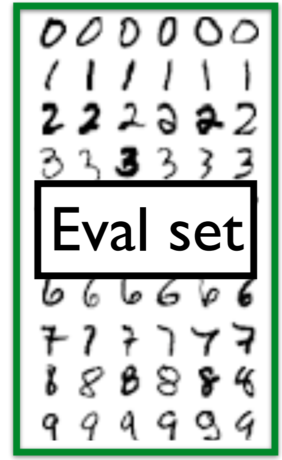
$\#$ hidden nodes $N_{hid} \in [10^1, 10^3]$



$N_{out} = 10$

N_{hid}

$N_{in} = 784$



$\widehat{f}$

Hyperparameters

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

Eval-loss

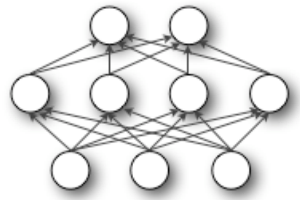
$\boxed{0.0577}$

hyperparameters

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

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

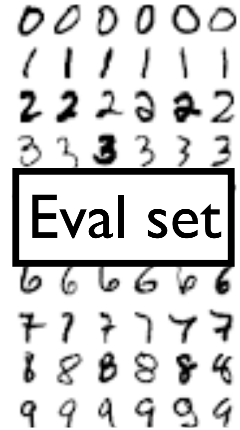
$\#$ hidden nodes $N_{hid} \in [10^1, 10^3]$



$N_{out} = 10$

N_{hid}

$N_{in} = 784$



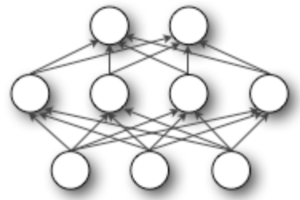
Hyperparameters	Eval-loss
$(10^{-1.6}, 10^{-2.4}, 10^{1.7})$	0.0577
$(10^{-1.0}, 10^{-1.2}, 10^{2.6})$	0.182
$(10^{-1.2}, 10^{-5.7}, 10^{1.4})$	0.0436
$(10^{-2.4}, 10^{-2.0}, 10^{2.9})$	0.0919
$(10^{-2.6}, 10^{-2.9}, 10^{1.9})$	0.0575
$(10^{-2.7}, 10^{-2.5}, 10^{2.4})$	0.0765
$(10^{-1.8}, 10^{-1.4}, 10^{2.6})$	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 $\lambda \in [10^{-6}, 10^{-1}]$

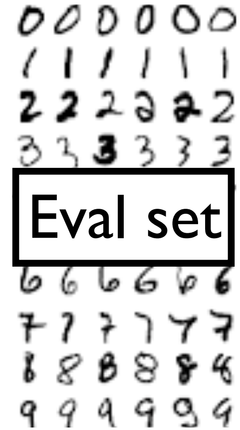
hidden nodes $N_{hid} \in [10^1, 10^3]$



$N_{out} = 10$

N_{hid}

$N_{in} = 784$

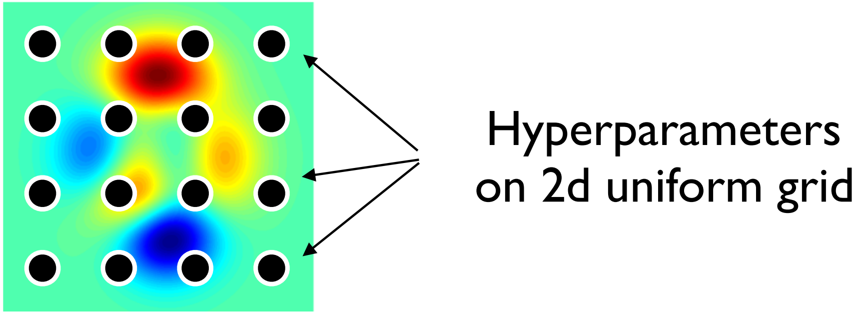


Hyperparameters	Eval-loss
$(10^{-1.6}, 10^{-2.4}, 10^{1.7})$	0.0577
$(10^{-1.0}, 10^{-1.2}, 10^{2.6})$	0.182
$(10^{-1.2}, 10^{-5.7}, 10^{1.4})$	0.0436
$(10^{-2.4}, 10^{-2.0}, 10^{2.9})$	0.0919
$(10^{-2.6}, 10^{-2.9}, 10^{1.9})$	0.0575
$(10^{-2.7}, 10^{-2.5}, 10^{2.4})$	0.0765
$(10^{-1.8}, 10^{-1.4}, 10^{2.6})$	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?

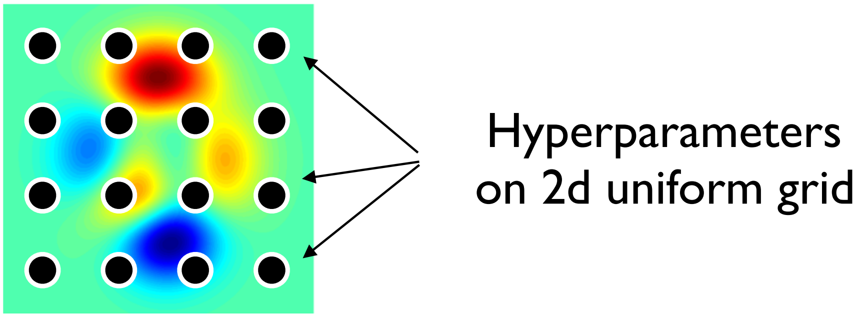
How do we choose hyperparameters to train and evaluate?

Grid search:

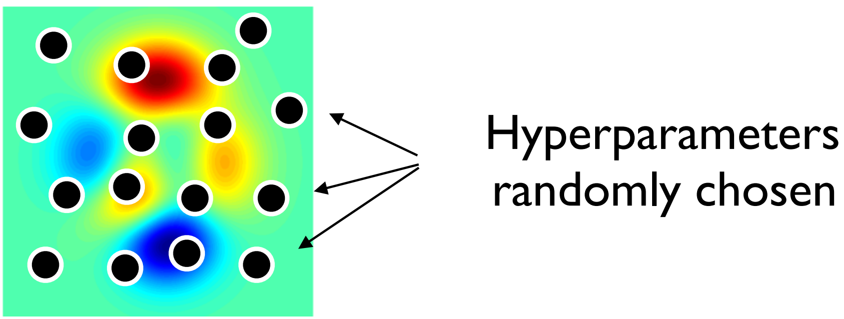


How do we choose hyperparameters to train and evaluate?

Grid search:

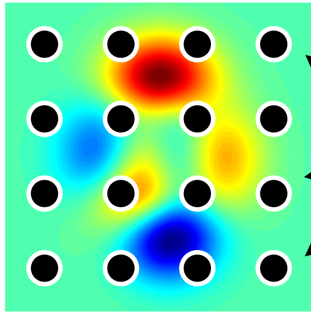


Random search:



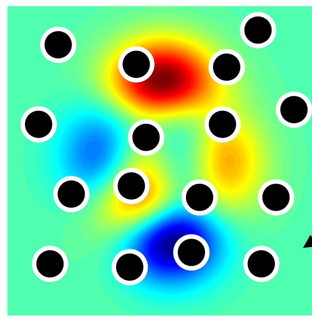
How do we choose hyperparameters to train and evaluate?

Grid search:



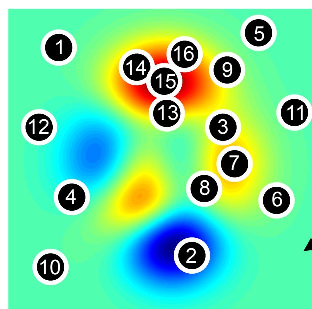
Hyperparameters
on 2d uniform grid

Random search:



Hyperparameters
randomly chosen

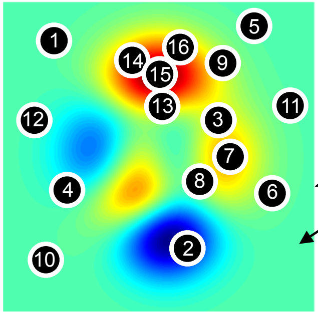
Bayesian Optimization:



Hyperparameters
adaptively chosen

Bayesian Optimization:

How does it work?



Hyperparameters ***adaptively*** chosen

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.

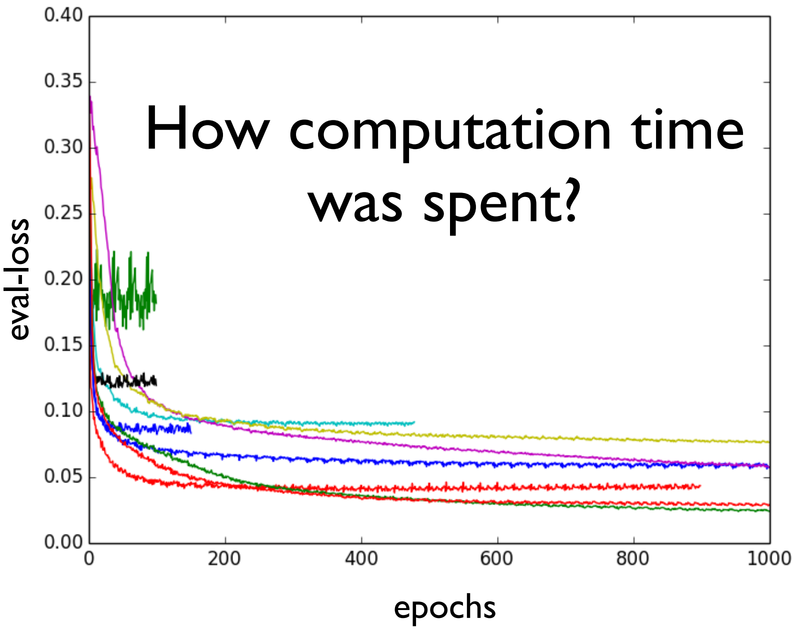
Alekh Agarwal, Peter Bartlett, and John Duchi. Oracle inequalities for computationally adaptive model selection. COLT, 2012.

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.

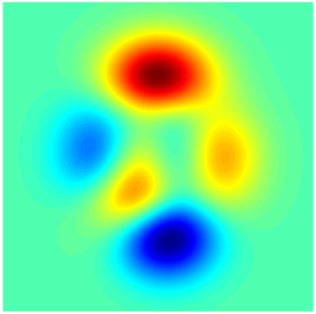
Li, Jamieson, DeSalvo, Rostamizadeh, Talwalkar. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. ICLR 2016.

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?