Kernels

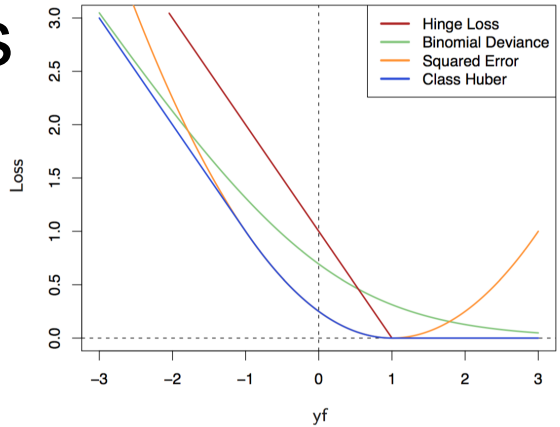
Machine Learning – CSE446

Kevin Jamieson
University of Washington

May 13, 2019

©2018 Kevin Jamieson

Machine Learning Problems



- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^{n} \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$

- Learning a model's parameters:

$$\sum_{i=1}^{n} \ell_i(w)$$

Each $\ell_i(w)$ is convex.

$$\text{Hinge Loss: } \ell_i(w) = \max\{0, 1 - y_i x_i^T w\}$$

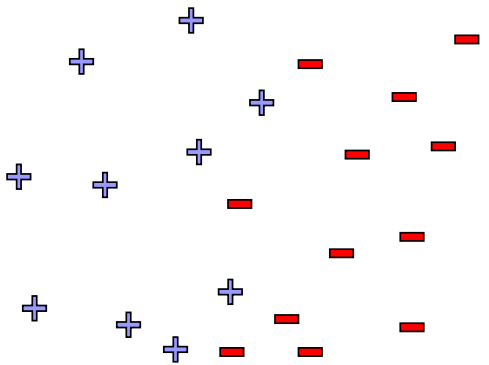
$$\text{Logistic Loss: } \ell_i(w) = \log(1 + \exp(-y_i\, x_i^T w))$$

$$\text{Squared error Loss: } \ell_i(w) = (y_i - x_i^T w)^2$$

All in terms of inner products! Even nearest neighbor can use inner products!

©Kevin Jamieson 2018

What if the data is not linearly separable?



Use features of features of features of features….

$$\phi(x) : \mathbb{R}^d \rightarrow \mathbb{R}^p$$

Write least squares in terms of Φ:

$$\widehat{w} = \arg\min_w \sum_{i=1}^{n} (y_i - \phi(x_i)^T w)^2 + \lambda ||w||_2^2$$

©2018 Kevin Jamieson

Kernel Trick

$\Phi = [\phi(x_1) \cdots \phi(x_n)]^T \in \mathbb{R}^{n \times p}$

$$\hat{w} = \arg\min_w \sum_{i=1}^{n} (y_i - \phi(x_i)^T w)^2 + \lambda ||w||_2^2$$

$\hat{w}^T \phi(x)$

$$= \arg\min_w \|y - \Phi^\top w\|_2^2 + \lambda \|w\|_2^2$$

$w = \Phi^T \alpha$

Claim: We can write $\hat{w} = \Phi^\top \hat{\alpha}$ with

$$\hat{\alpha} = \arg\min_\alpha \|y - \underbrace{\Phi\Phi^\top}_{n \times n} \alpha\|_2^2 + \lambda \alpha^\top \underbrace{\Phi\Phi^\top}_{n \times n} \alpha$$

$[\Phi\Phi^T]_{i,j} = \phi(x_i)^T \phi(x_j)$

$w^T \phi(x)$

©2018 Kevin Jamieson

Kernel Trick

$$\widehat{w} = \arg\min_w \sum_{i=1}^{n} (y_i - \phi(x_i)^T w)^2 + \lambda ||w||_2^2$$

$$= \arg\min_w \|y - \Phi^\top w\|_2^2 + \lambda \|w\|_2^2$$

Claim: We can write $\widehat{w} = \Phi^\top \widehat{\alpha}$ with

$$\widehat{\alpha} = \arg\min_\alpha \|y - \Phi\Phi^\top \alpha\|_2^2 + \lambda \alpha^\top \Phi\Phi^\top \alpha$$

Proof: write $\widehat{w} = P_\Phi \widehat{w} + (I - P_\Phi)\widehat{w}$ where P_Φ is the projection onto $\{\Phi^\top \beta : \beta \in \mathbb{R}^n\}$. Use facts:
i) $\Phi^\top \widehat{w} = \Phi^\top P_\Phi \widehat{w}$
ii) $\|\widehat{w}\|_2^2 = \|P_\Phi \widehat{w}\|_2^2 + \|(I - P_\Phi)\widehat{w}\|_2^2$
to argue $\|(I - P_\Phi)\widehat{w}\|_2^2 = 0 \implies \widehat{w} = P_\Phi \widehat{w} = \Phi^\top \widehat{\alpha}$

©2018 Kevin Jamieson

Kernel Trick

$$\widehat{w} = \arg\min_{w} \sum_{i=1}^{n} (y_i - \phi(x_i)^T w)^2 + \lambda ||w||_2^2$$

$$= \arg\min_{w} \|y - \Phi^\top w\|_2^2 + \lambda \|w\|_2^2$$

Claim: We can write $\widehat{w} = \Phi^\top \widehat{\alpha}$ with

$$\widehat{\alpha} = \arg\min_{\alpha} \|y - \Phi\Phi^\top \alpha\|_2^2 + \lambda \alpha^\top \Phi\Phi^\top \alpha$$

$$= \arg\min_{\alpha} \|y - \mathbf{K}\alpha\|_2^2 + \lambda \alpha^\top \mathbf{K}\alpha$$

$$\mathbf{K}_{i,j} = [\Phi\Phi^\top]_{i,j} = \phi(x_i)^\top \phi(x_j) \overset{\text{def}}{=} K(x_i, x_j)$$

©2018 Kevin Jamieson

General Solution

$(\Phi\Phi^T)^T = (\Phi^T)^T\Phi^T = \Phi\Phi^T$

$(AB)^T = B^TA^T$

$K^T = K$

$$\mathbf{K}_{i,j} = [\Phi\Phi^\top]_{i,j} = \phi(x_i)^\top\phi(x_j) = K(x_i, x_j)$$

$$\widehat{\alpha} = \arg\min_{\alpha} \|\mathbf{K}\alpha - y\|_2^2 + \lambda\alpha^\top\mathbf{K}\alpha$$

$$\nabla_\alpha(\cdot) = 0 = 2K^T(K\alpha - y) + 2\lambda K\alpha$$

$$= 2K\left[K\alpha - y + \lambda I\alpha\right]$$

$$= 2K\left[(K + \lambda I)\alpha - y\right]$$

$$\hat{\alpha} = (K + \lambda I)^{-1}y$$

©2018 Kevin Jamieson

General Solution

$$\mathbf{K}_{i,j} = [\Phi\Phi^\top]_{i,j} = \phi(x_i)^\top \phi(x_j) = K(x_i, x_j)$$

$$\widehat{\alpha} = \arg\min_{\alpha} \|\mathbf{K}\alpha - y\|_2^2 + \lambda \alpha^\top \mathbf{K}\alpha$$

$$\phi(x) = \begin{bmatrix} \phi(x_1) & \cdots & \phi(x_n) \end{bmatrix}^\top \phi(x) = \begin{bmatrix} K(x_1, x) \\ \vdots \\ K(x_n, x) \end{bmatrix}$$

$$\begin{aligned} 0 = \nabla_\alpha(\cdot) &= 2\mathbf{K}^\top(\mathbf{K}\alpha - y) + 2\lambda\mathbf{K}\alpha \\ &= 2\mathbf{K}[(\mathbf{K} + \lambda I)\alpha - y] \end{aligned}$$

$$\widehat{\alpha} = (\mathbf{K} + \lambda I)^{-1} y$$

Predict new point x as: $\widehat{w}^\top \phi(x) = \widehat{\alpha}^\top \Phi \phi(x) = \sum_{i=1}^{n} K(x_i, x)\widehat{\alpha}_i$

©2018 Kevin Jamieson

General Solution

$$\mathbf{K}_{i,j} = [\Phi\Phi^\top]_{i,j} = \phi(x_i)^\top \phi(x_j) = K(x_i, x_j)$$

$$\widehat{\alpha} = \arg\min_{\alpha} \|\mathbf{K}\alpha - y\|_2^2 + \lambda \alpha^\top \mathbf{K}\alpha$$

$$\begin{aligned} 0 = \nabla_\alpha(\cdot) &= 2\mathbf{K}^\top(\mathbf{K}\alpha - y) + 2\lambda\mathbf{K}\alpha \\ &= 2\mathbf{K}[(\mathbf{K} + \lambda I)\alpha - y] \end{aligned}$$

$$\widehat{\alpha} = (\mathbf{K} + \lambda I)^{-1} y$$

Thus, we have that: $\widehat{w} = (\Phi^\top\Phi + \lambda I_p)^{-1}\Phi^\top y$

but also that: $\widehat{w} = \Phi^\top\widehat{\alpha} = \Phi^\top(\Phi\Phi^\top + \lambda I_n)^{-1} y$

©2018 Kevin Jamieson

Why regularization?

$$\mathbf{K}_{i,j} = [\Phi\Phi^\top]_{i,j} = \phi(x_i)^\top \phi(x_j) = K(x_i, x_j)$$

$$\widehat{\alpha} = \arg\min_{\alpha} \|\mathbf{K}\alpha - y\|_2^2 + \lambda \alpha^\top \mathbf{K}\alpha$$

$$\widehat{\alpha} = (\mathbf{K} + \lambda I)^{-1} y$$

$$\widehat{w}^\top \phi(x) = \widehat{\alpha}^\top \Phi x = \sum_{i=1}^{n} K(x_i, x)\widehat{\alpha}_i$$

What if $\lambda = 0$?

©2018 Kevin Jamieson

Common kernels

Linear: $K(u,v) = u^T v$

- Polynomials of degree exactly d

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^d$$

- Polynomials of degree up to d

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^d$$

- Gaussian (squared exponential) kernel (RBF)

$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||_2^2}{2\sigma^2}\right)$$

- Sigmoid

$$K(\mathbf{u}, \mathbf{v}) = \tanh(\eta \mathbf{u} \cdot \mathbf{v} + \nu)$$

©2018 Kevin Jamieson

Mercer's Theorem

- When do we have a valid Kernel K(x,x')?
- Sufficient:

  $K(x, x')$ is a valid kernel if there exists $\phi(x)$ such that $K(x, x') = \phi(x)^T \phi(x')$

- Mercer's Theorem:

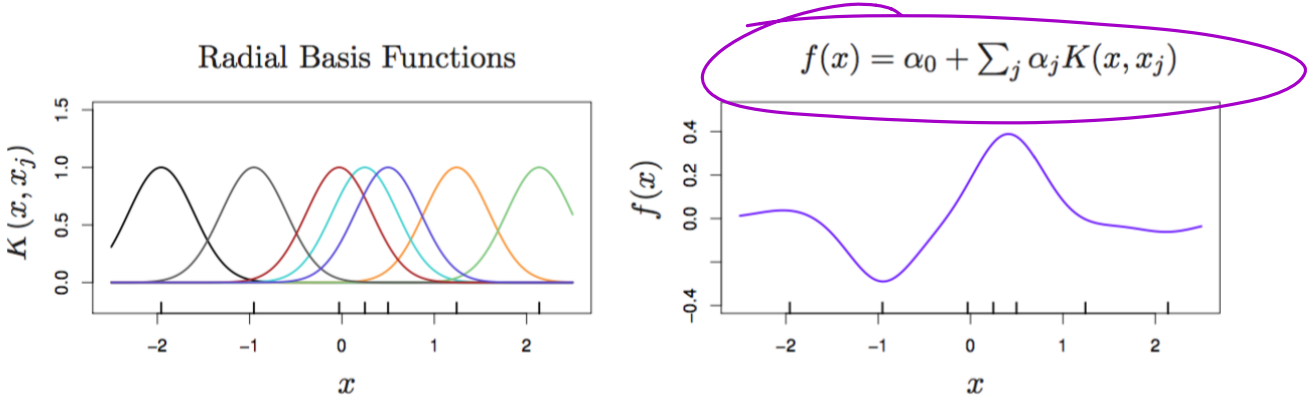
  $K(x, x')$ is a valid kernel if and only if $\mathbf{K}$ is symmetric and positive semi-definite for any pointset $(x_1, \ldots, x_n)$ where $\mathbf{K}_{i,j} = K(x_i, x_j)$.

©2018 Kevin Jamieson

RBF Kernel

$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||_2^2}{2\sigma^2}\right)$$

- Note that this is like weighting "bumps" on each point like kernel smoothing but now we **learn** the weights



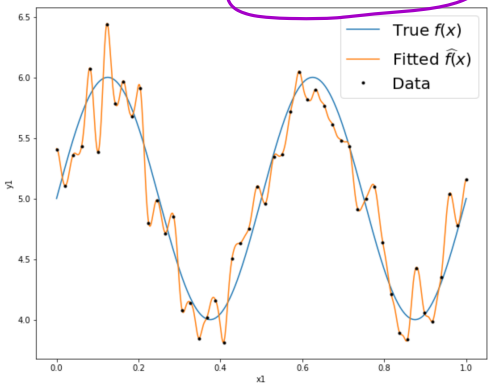
©2018 Kevin Jamieson

RBF Kernel

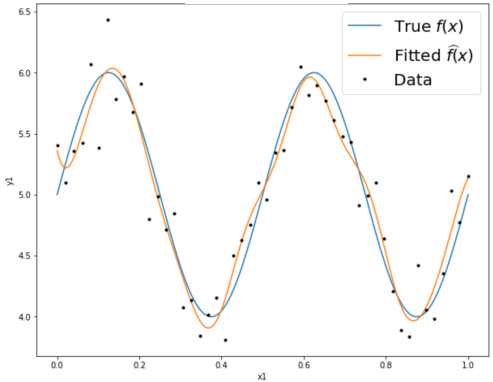
$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||_2^2}{2\sigma^2}\right)$$

The bandwidth sigma has an enormous effect on fit:

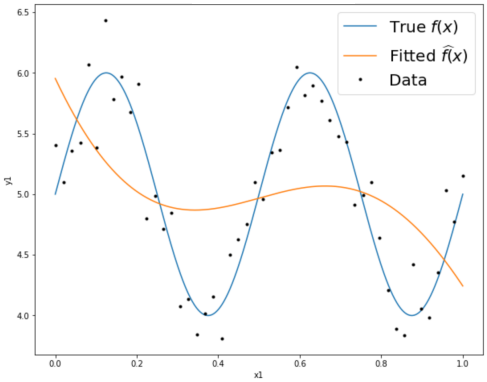
$\sigma = 10^{-2}$ $\lambda = 10^{-4}$



$\sigma = 10^{-1}$ $\lambda = 10^{-4}$



$\sigma = 10^{-0}$ $\lambda = 10^{-4}$



$$\widehat{f}(x) = \sum_{i=1}^{n} \widehat{\alpha}_i K(x_i, x)$$

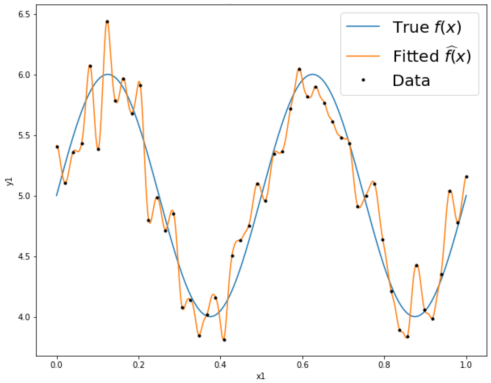
©2018 Kevin Jamieson

RBF Kernel

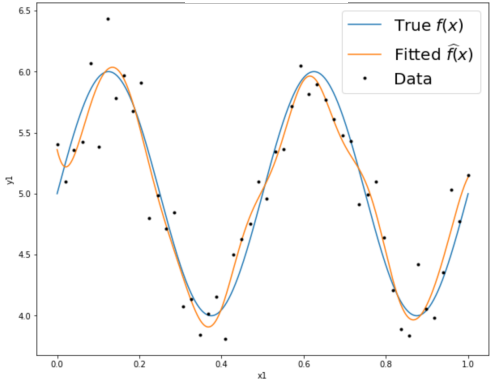
$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||_2^2}{2\sigma^2}\right)$$

The bandwidth sigma has an enormous effect on fit:

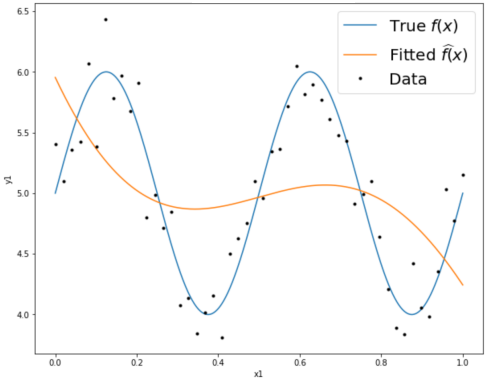
$\sigma = 10^{-2}$ $\lambda = 10^{-4}$



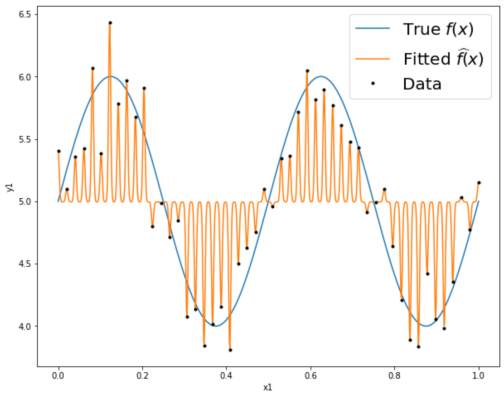
$\sigma = 10^{-1}$ $\lambda = 10^{-4}$



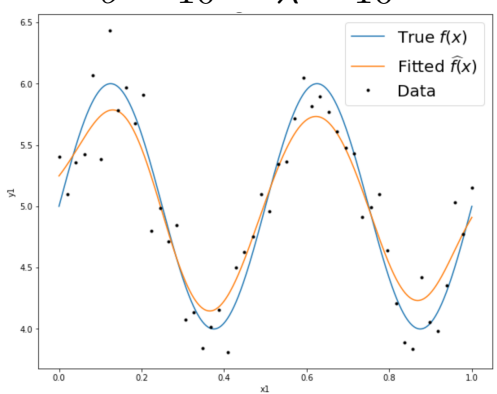
$\sigma = 10^{-0}$ $\lambda = 10^{-4}$



$\sigma = 10^{-3}$ $\lambda = 10^{-4}$



$\sigma = 10^{-1}$ $\lambda = 10^{-0}$



$$\widehat{f}(x) = \sum_{i=1}^{n} \widehat{\alpha}_i K(x_i, x)$$

©2018 Kevin Jamieson

RBF kernel and random features

$$2\cos(\alpha)\cos(\beta) = \cos(\alpha+\beta) + \cos(\alpha-\beta)$$

$$e^{jz} = \cos(z) + j\sin(z)$$

Recall HW1 where we used the feature map:

$$\phi(x) = \begin{bmatrix} \sqrt{2}\cos(w_1^T x + b_1) \\ \vdots \\ \sqrt{2}\cos(w_p^T x + b_p) \end{bmatrix} \qquad \begin{aligned} w_k &\sim \mathcal{N}(0, 2\gamma I) \\ b_k &\sim \text{uniform}(0, \pi) \end{aligned}$$

$$\mathbb{E}\left[\frac{1}{p}\phi(x)^T\phi(y)\right] = \frac{1}{p}\sum_{k=1}^{p}\mathbb{E}[2\cos(w_k^T x + b_k)\cos(w_k^T y + b_k)]$$

$$= \mathbb{E}_{w,b}[2\cos(w^T x + b)\cos(w^T y + b)]$$

$$= \mathbb{E}_{w,b}\left[\cos(w^T(x-y)) + \cos(w^T(x+y)+2b)\right]$$

(the $\cos(w^T(x+y)+2b)$ term $\to 0$)

RBF kernel and random features

$$2\cos(\alpha)\cos(\beta) = \cos(\alpha+\beta) + \cos(\alpha-\beta)$$

$$e^{jz} = \cos(z) + j\sin(z)$$

Recall HW1 where we used the feature map:

$$\phi(x) = \begin{bmatrix} \sqrt{2}\cos(w_1^T x + b_1) \\ \vdots \\ \sqrt{2}\cos(w_p^T x + b_p) \end{bmatrix} \qquad \begin{aligned} w_k &\sim \mathcal{N}(0, 2\gamma I) \\ b_k &\sim \text{uniform}(0, \pi) \end{aligned}$$

$$\begin{aligned} \mathbb{E}[\frac{1}{p}\phi(x)^T\phi(y)] &= \frac{1}{p}\sum_{k=1}^{p}\mathbb{E}[2\cos(w_k^T x + b_k)\cos(w_k^T y + b_k)] \\ &= \mathbb{E}_{w,b}[2\cos(w^T x + b)\cos(w^T y + b)] \\ &= e^{-\gamma||x-y||_2^2} \end{aligned}$$

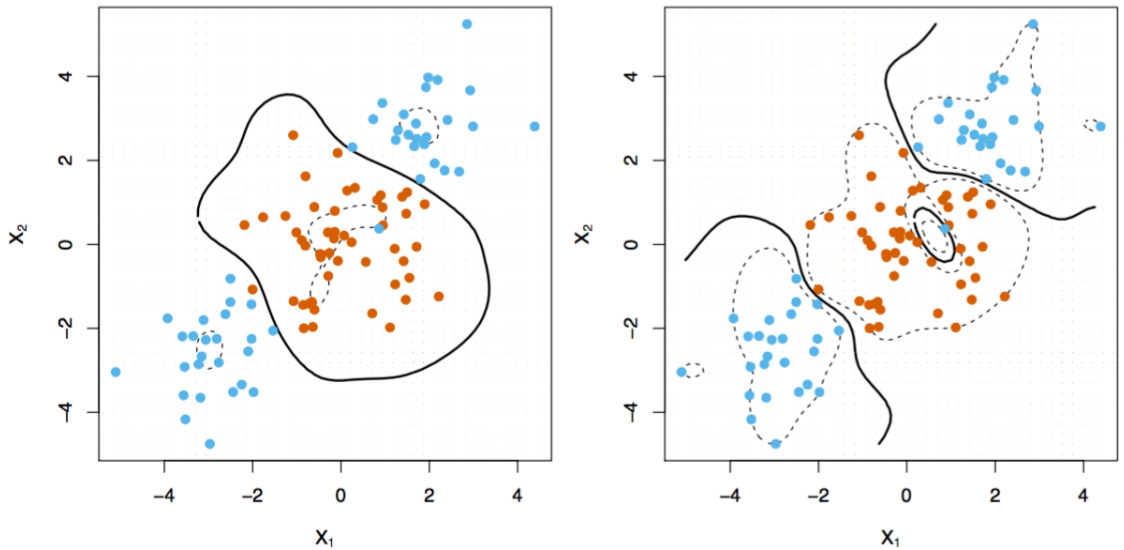
[Rahimi, Recht NIPS 2007]
"NIPS Test of Time Award, 2018"

©2018 Kevin Jamieson

RBF Classification

$$\widehat{w} = \sum_{i=1}^{n} \max\{0, 1 - y_i(b + x_i^T w)\} + \lambda ||w||_2^2$$

$$\min_{\alpha,b} \sum_{i=1}^{n} \max\{0, 1 - y_i(b + \sum_{j=1}^{n} \alpha_j \langle x_i, x_j \rangle)\} + \lambda \sum_{i,j=1}^{n} \alpha_i \alpha_j \underbrace{\langle x_i, x_j \rangle}_{K(x_i, x_j)}$$



Wait, infinite dimensions?

- Isn't everything separable there? How are we not overfitting?

- Regularization! Fat shattering (R/margin)^2

©2018 Kevin Jamieson

String Kernels

Example from Efron and Hastie, 2016

Amino acid sequences of different lengths:

x1

IPTSALVKETLALLSTHRTLLIANETLRIPVPVHKNHQLCTEEIFQGIGTLESQTVQGGTV
ERLFKNLSLIKKYIDGQKKKCGEERRRVNQFLDY**LQE**FLGVMNTEWI

x2

PHRRDLCSRSIWLARKIRSDLTALTESYVKHQGLWSELTEAER**LQE**NLQAYRTFHVLLA
RLLEDQQVHFTPTEGDFHQAIHTLLLQVAAFAYQIEELMILLEYKIPRNEADGMLFEKK
LWGLKV**LQE**LSQWTVRSIHDLRFISSHQTGIP

All subsequences of length 3 (of possible 20 amino acids) $20^3 = 8{,}000$

$$h^3_{\text{LQE}}(x_1) = 1 \text{ and } h^3_{\text{LQE}}(x_2) = 2.$$

©2018 Kevin Jamieson

Bootstrap

Machine Learning – CSE446

Kevin Jamieson
University of Washington

May 13, 2019

©2018 Kevin Jamieson

Limitations of CV

- An 80/20 split throws out a relatively large amount of data if only have, say, 20 examples.
- Test error is informative, but how accurate is this number? (e.g., 3/5 heads vs. 30/50)
- How do I get confidence intervals on statistics like the median or variance of a distribution?
- Instead of the error for the entire dataset, what if I want to study the error for a *particular example* x?

©2018 Kevin Jamieson

Limitations of CV

- An 80/20 split throws out a relatively large amount of data if only have, say, 20 examples.
- Test error is informative, but how accurate is this number? (e.g., 3/5 heads vs. 30/50)
- How do I get confidence intervals on statistics like the median or variance of a distribution?
- Instead of the error for the entire dataset, what if I want to study the error for a *particular example* x?

The Bootstrap: Developed by Efron in 1979.

©2018 Kevin Jamieson

Bootstrap: basic idea

Given dataset drawn iid samples with CDF F_Z: $F_Z(x) = P(Z \leq x)$

$$\mathcal{D} = \{z_1, \ldots, z_n\} \overset{i.i.d.}{\sim} F_Z$$

We compute a *statistic* of the data to get: $\widehat{\theta} = t(\mathcal{D})$

©2018 Kevin Jamieson

Bootstrap: basic idea

Given dataset drawn iid samples with CDF F_Z:

$$\mathcal{D} = \{z_1, \ldots, z_n\} \overset{i.i.d.}{\sim} F_Z$$

We compute a *statistic* of the data to get: $\widehat{\theta} = t(\mathcal{D})$

For b=1,…,B define the *b*th ***bootstrapped*** dataset as drawing *n* samples **with replacement** from *D*

$$\mathcal{D}^{*b} = \{z_1^{*b}, \ldots, z_n^{*b}\} \overset{i.i.d.}{\sim} \widehat{F}_{Z,n}$$

and the *b*th bootstrapped statistic as: $\theta^{*b} = t(\mathcal{D}^{*b})$

©2018 Kevin Jamieson

Bootstrap: basic idea

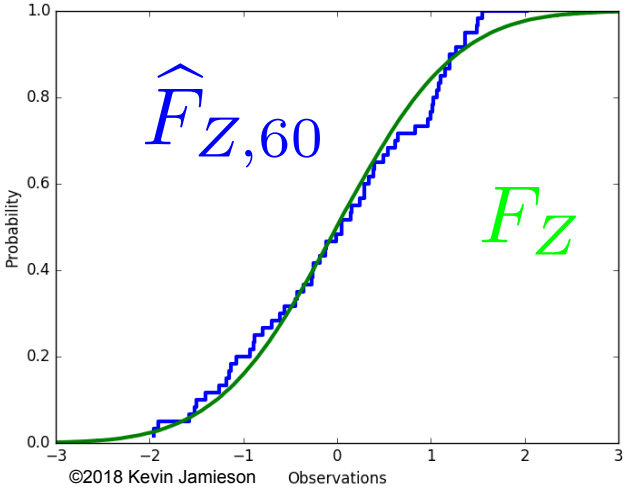
Given dataset drawn iid samples with CDF F_Z:

$$\mathcal{D} = \{z_1, \ldots, z_n\} \overset{i.i.d.}{\sim} F_Z \qquad \widehat{\theta} = t(\mathcal{D})$$

For b=1,…,B, samples sampled **with replacement** from *D*

$$\mathcal{D}^{*b} = \{z_1^{*b}, \ldots, z_n^{*b}\} \overset{i.i.d.}{\sim} \widehat{F}_{Z,n} \quad \theta^{*b} = t(\mathcal{D}^{*b})$$

$$\widehat{F}_{Z,n}(x) = \frac{1}{n} \sum \mathbb{I}\{z_i \leq x\}$$



©2018 Kevin Jamieson

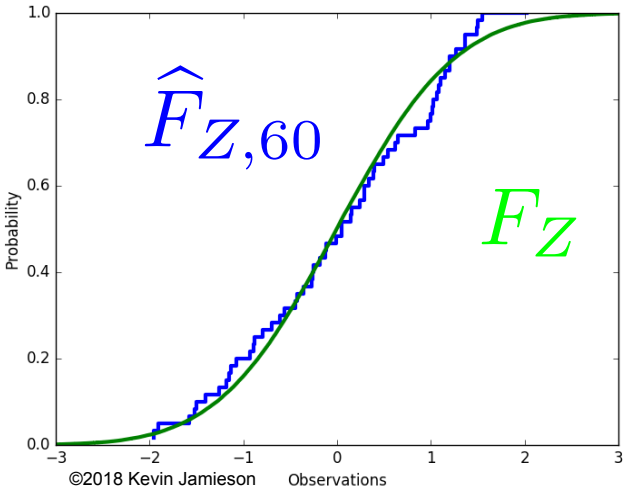
Bootstrap: basic idea

Given dataset drawn iid samples with CDF F_Z:

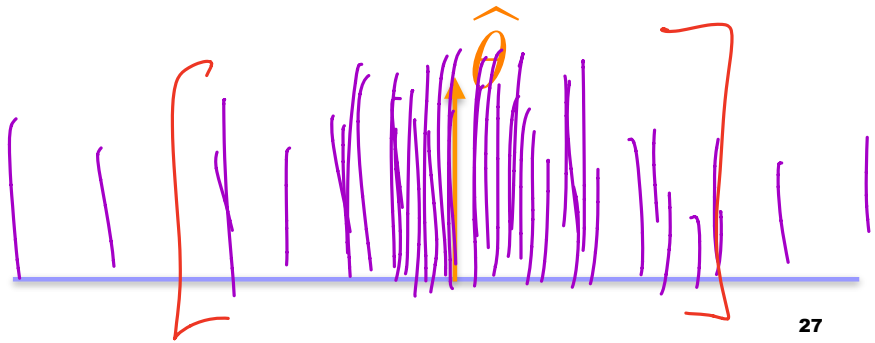
$$\mathcal{D} = \{z_1, \ldots, z_n\} \overset{i.i.d.}{\sim} F_Z \qquad \widehat{\theta} = t(\mathcal{D})$$

For b=1,…,B, samples sampled **with replacement** from *D*

$$\mathcal{D}^{*b} = \{z_1^{*b}, \ldots, z_n^{*b}\} \overset{i.i.d.}{\sim} \widehat{F}_{Z,n} \quad \theta^{*b} = t(\mathcal{D}^{*b})$$



$$\max_x |\widehat{F}_n(x) - F(x)| \to 0 \quad \text{as } n \to \infty$$

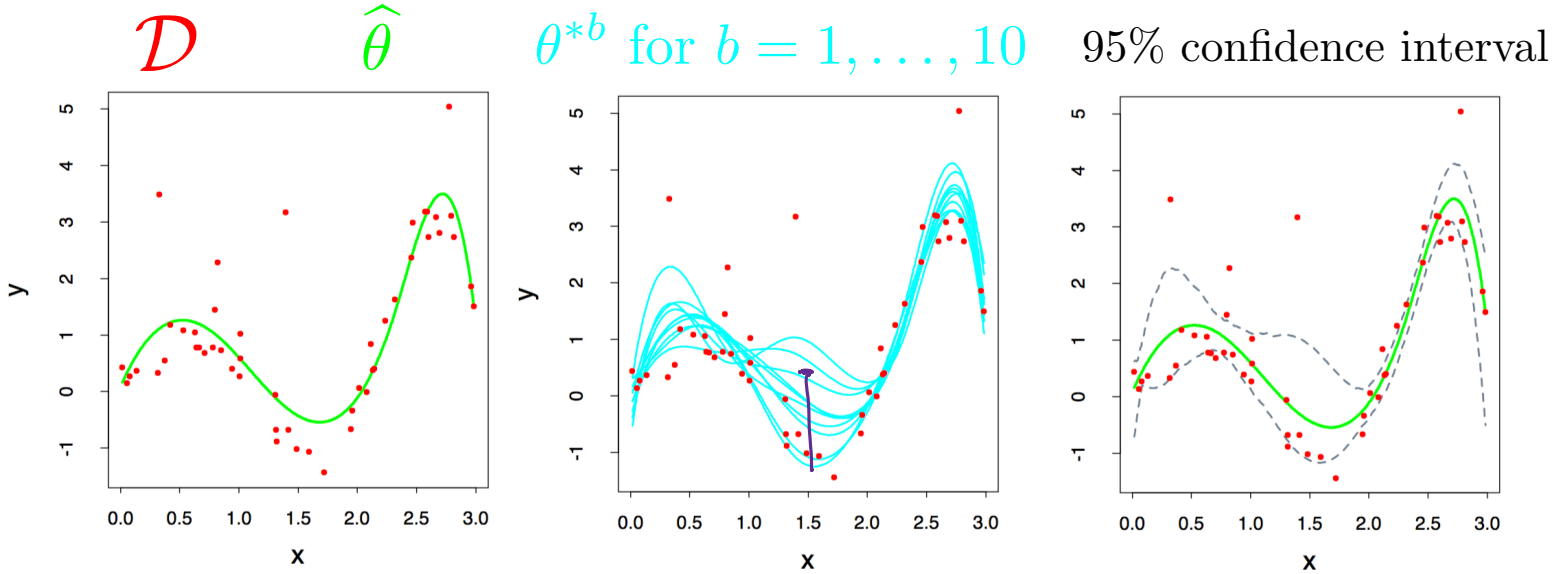


©2018 Kevin Jamieson

Applications

Common applications of the bootstrap:

- Estimate parameters that escape simple analysis like the variance or median of an estimate
- Confidence intervals
- Estimates of error for a particular example:



Figures from Hastie et al

©2018 Kevin Jamieson

Takeaways

Advantages:

- Bootstrap is **very** generally applicable. Build a confidence interval around ***anything***
- **Very** simple to use
- Appears to give meaningful results even when the amount of data is very small
- Very strong **asymptotic theory** (as num. examples goes to infinity)

©2018 Kevin Jamieson

Takeaways

Advantages:

- Bootstrap is **very** generally applicable. Build a confidence interval around ***anything***
- **Very** simple to use
- Appears to give meaningful results even when the amount of data is very small
- Very strong **asymptotic theory** (as num. examples goes to infinity)

Disadvantages

- Very few meaningful finite-sample guarantees
- Potentially **computationally intensive**
- Reliability relies on test statistic and rate of convergence of empirical CDF to true CDF, which is unknown
- Poor performance on “extreme statistics” (e.g., the max)

Not perfect, but better than nothing.

©2018 Kevin Jamieson

Warm up: risk prediction with logistic regression

- Boss gives you a bunch of data on loans defaulting or not:

$$\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$$

- You model the data as: $P(Y = y|x, w) = \dfrac{1}{1 + \exp(-y\, w^T x)}$
- And compute the maximum likelihood estimator:

$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^n P(y_i|x_i, w)$$

For a new loan application x, boss recommends to give loan if your model says they will repay it with probability at least .95 (i.e. low risk):

Give loan to x if $\dfrac{1}{1 + \exp(-\widehat{w}_{MLE}^T x)} \geq .95$

- One year later only half of loans are paid back and the bank folds. What might have happened?

How would you use the bootstrap to do this differently?

©Kevin Jamieson 2018

Clustering
K-means

Machine Learning – CSE 446

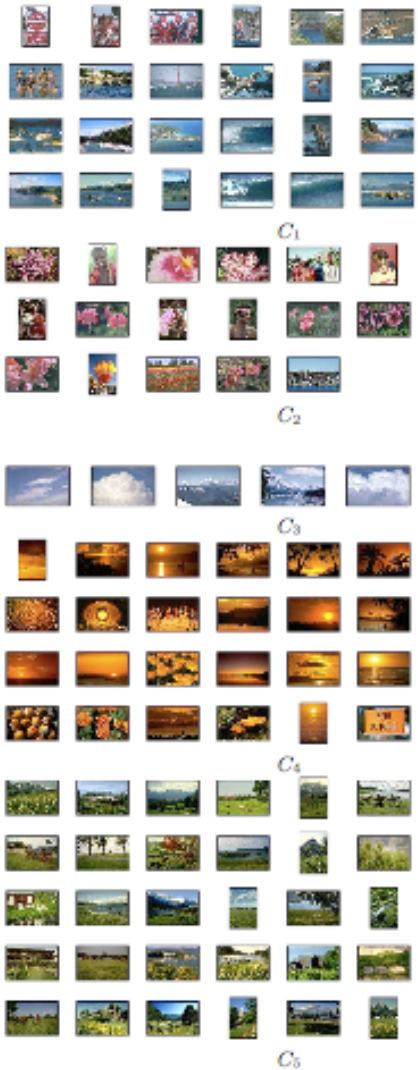
Kevin Jamieson
University of Washington

May 13, 2019

©Kevin Jamieson 2018
©Kevin Jamieson 2018

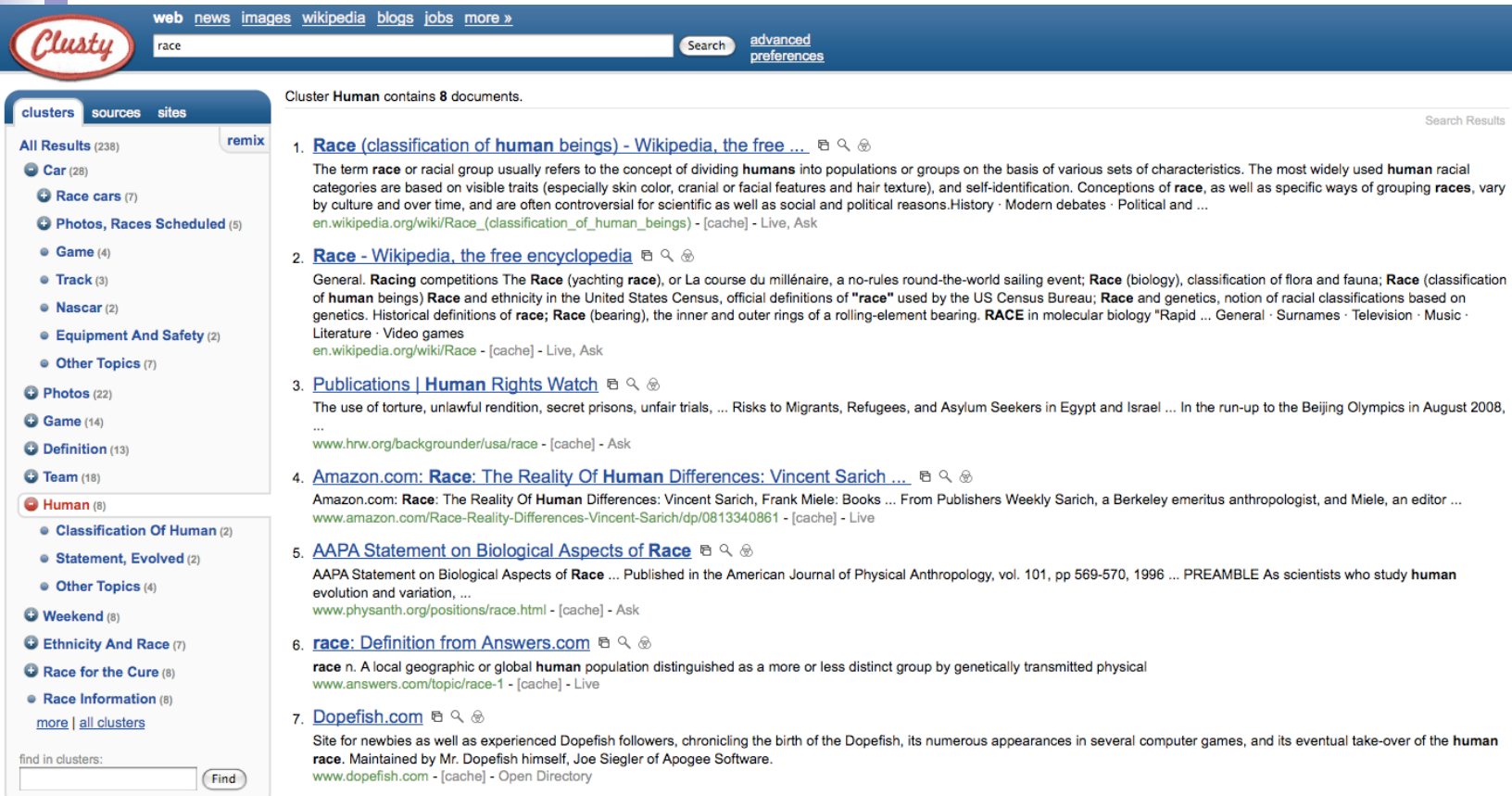
Clustering images

Set of Images



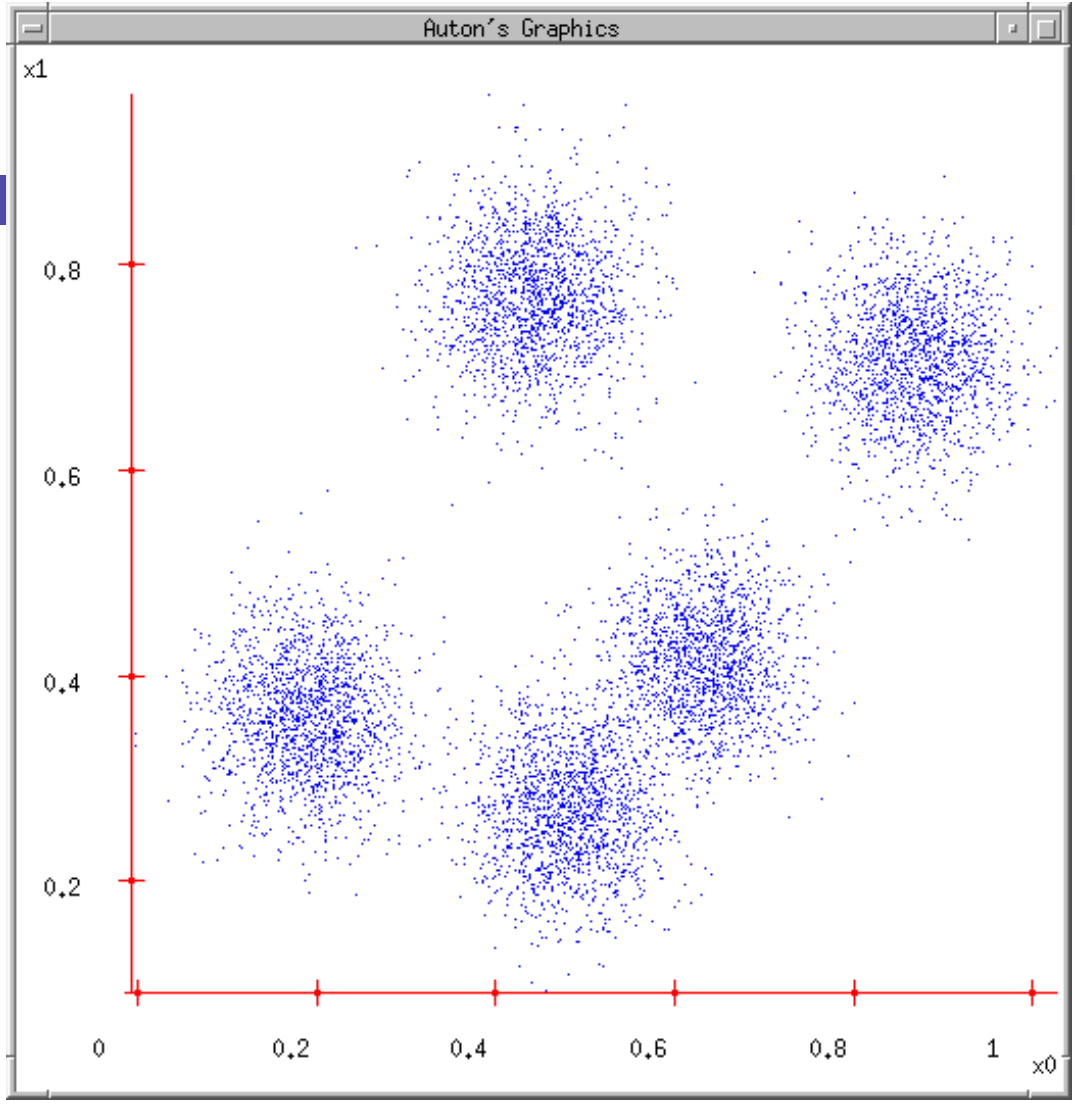
[Goldberger et al.]

Clustering web search results



©Kevin Jamieson 2018
©Kevin Jamieson 2018

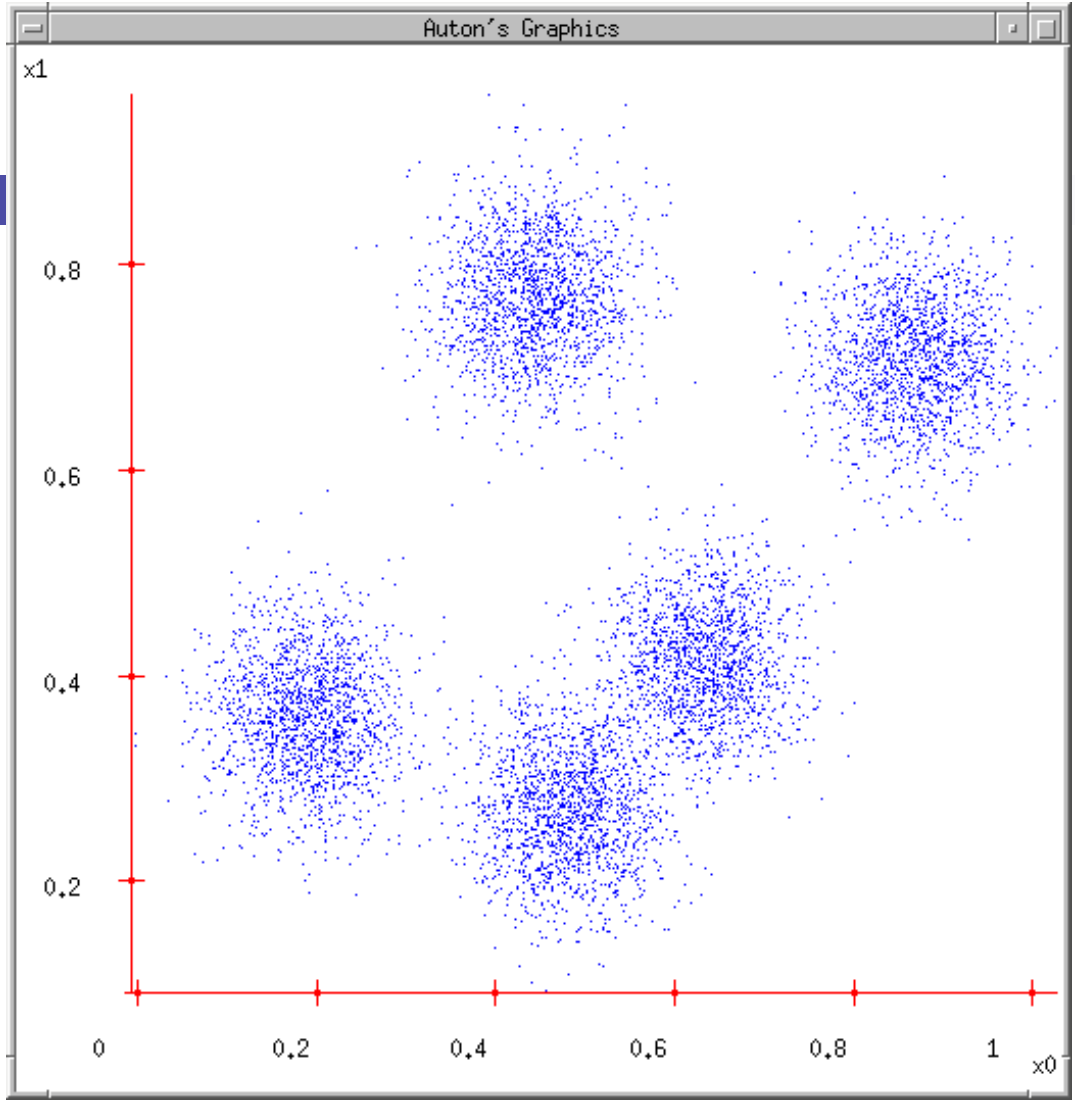
Some Data



©Kevin Jamieson 2018
©Kevin Jamieson 2018

K-means

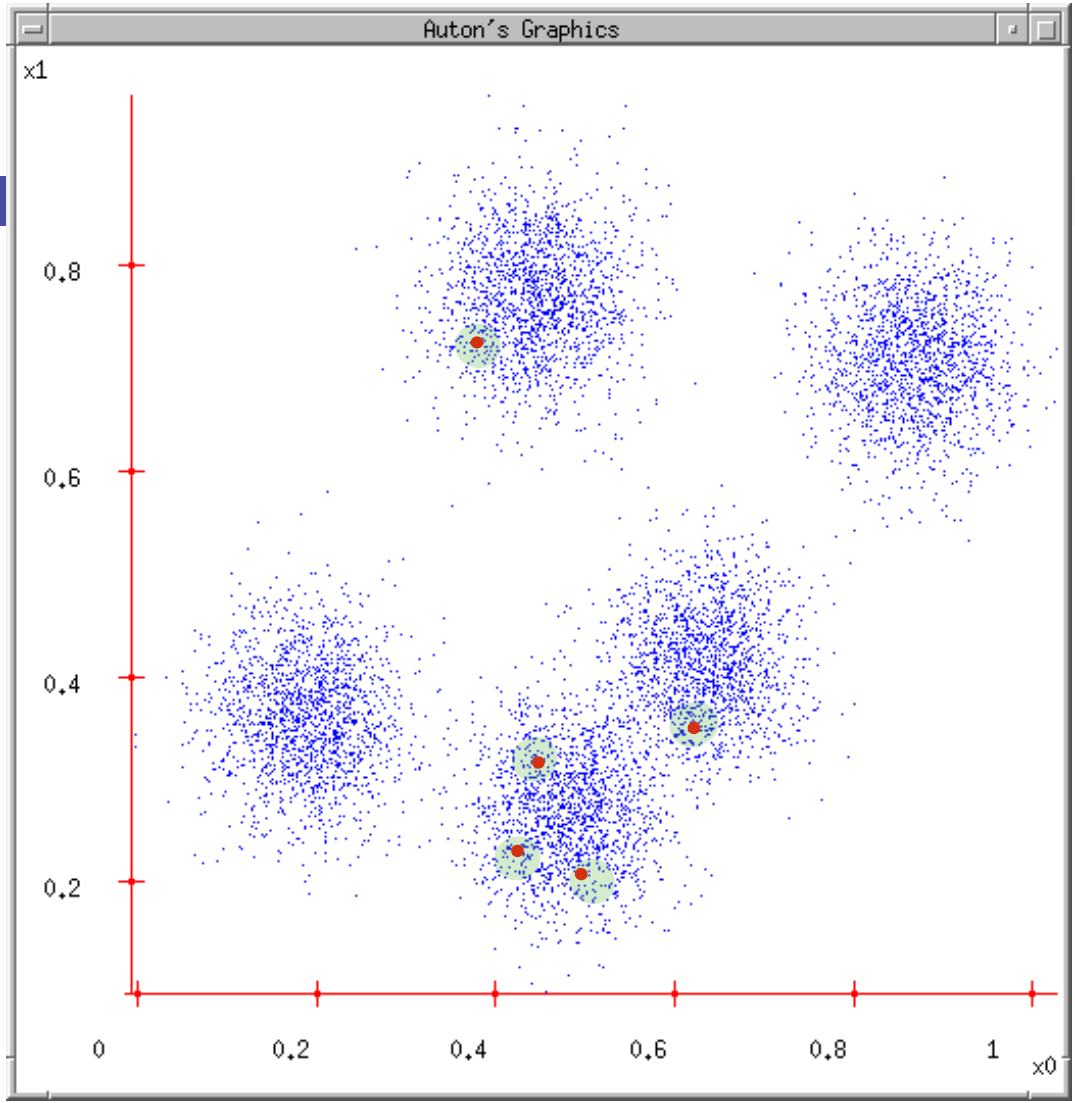
1. Ask user how many clusters they'd like. *(e.g. k=5)*



©Kevin Jamieson 2018
©Kevin Jamieson 2018

K-means

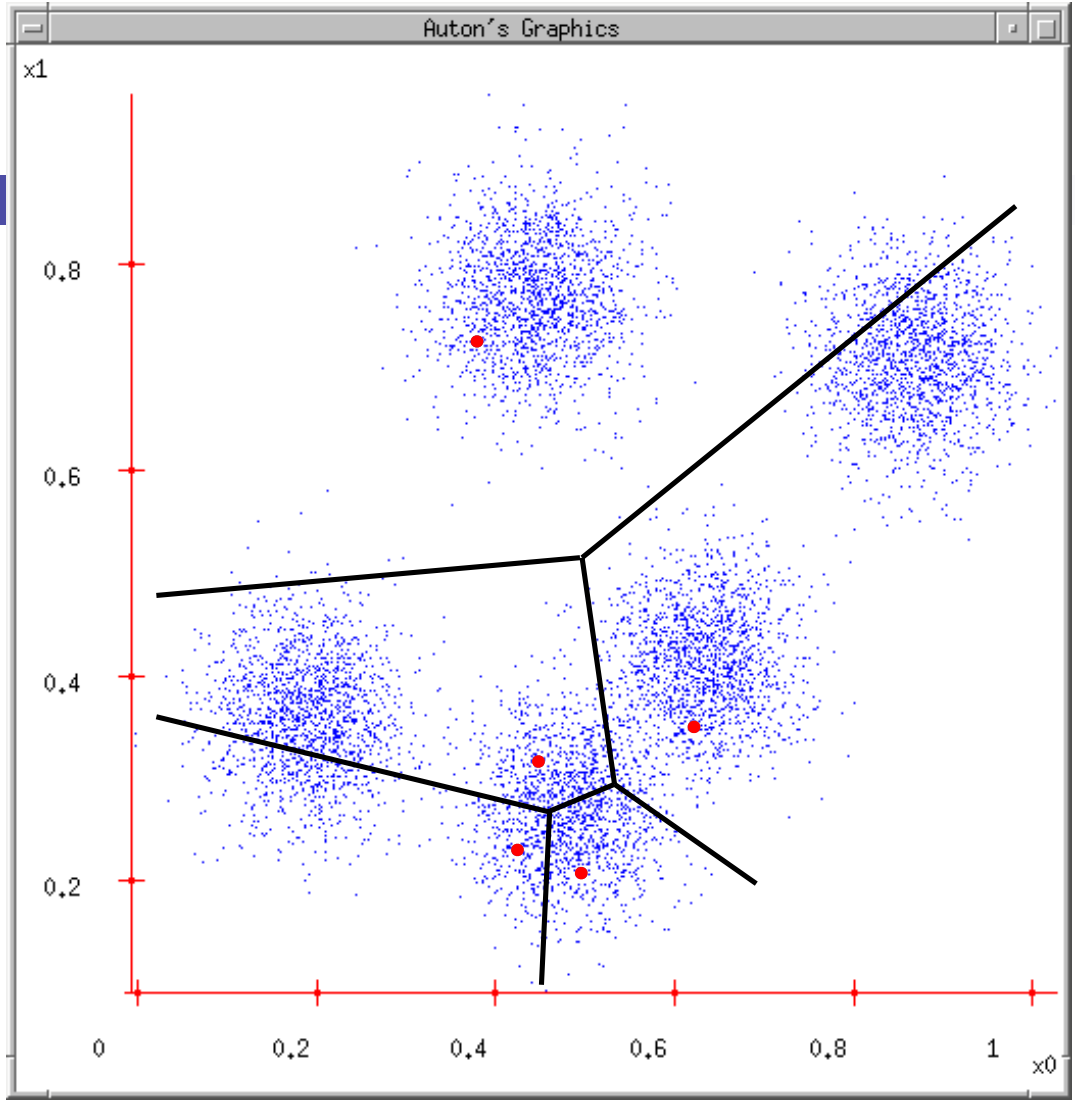
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



©Kevin Jamieson 2018
©Kevin Jamieson 2018

K-means

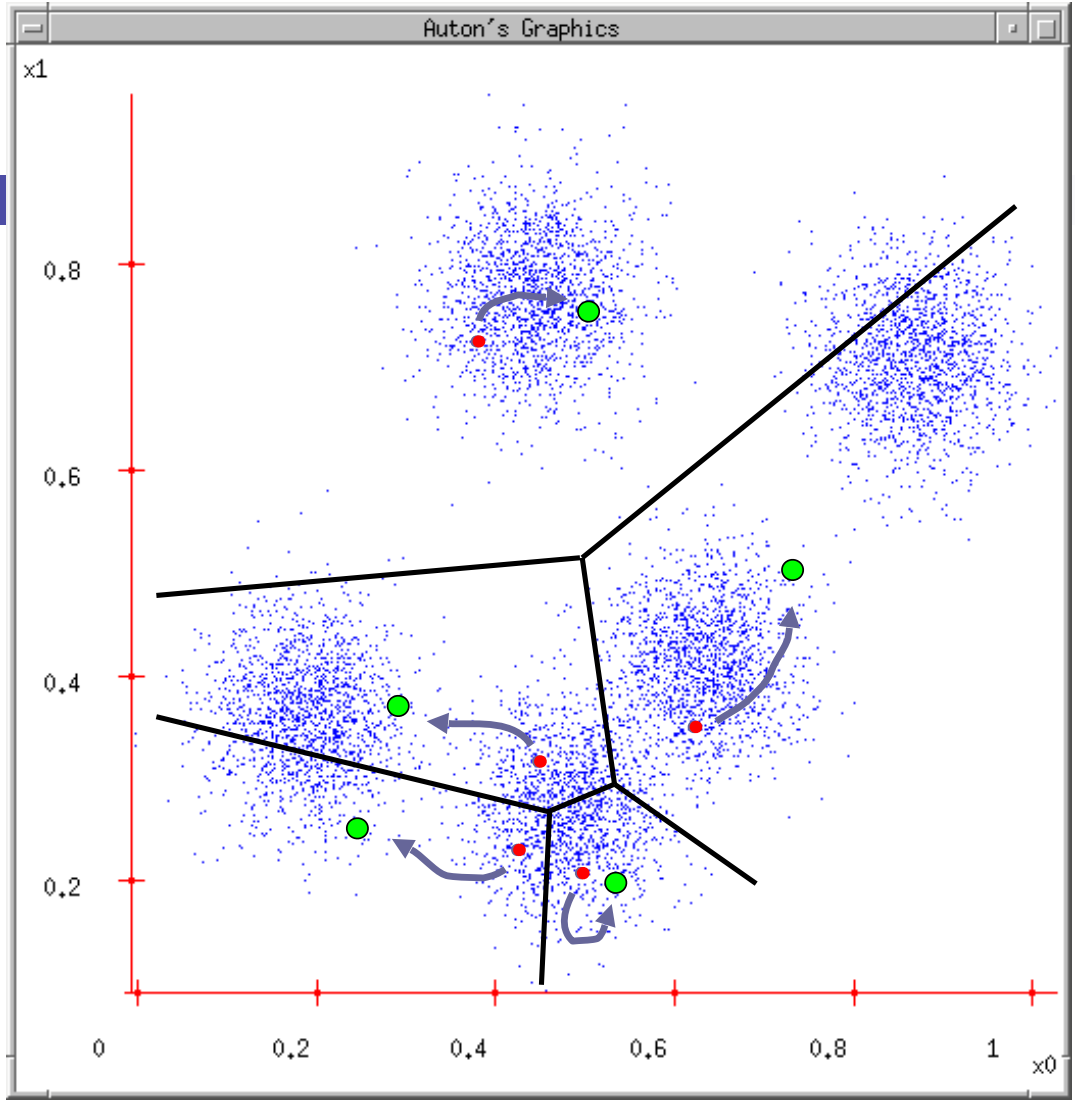
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



©Kevin Jamieson 2018
©Kevin Jamieson 2018

K-means

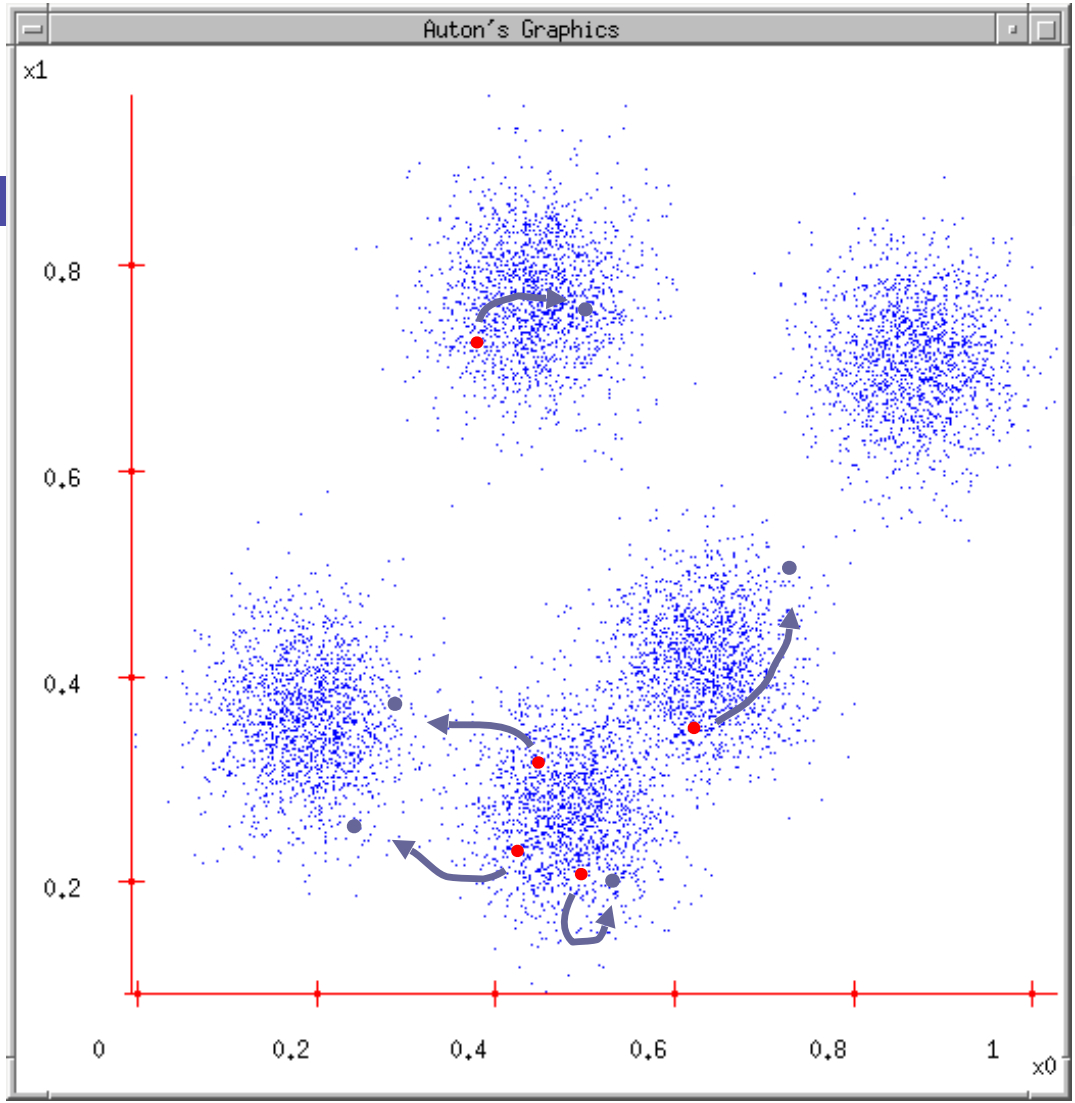
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



©Kevin Jamieson 2018
©Kevin Jamieson 2018

K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!



©Kevin Jamieson 2018
©Kevin Jamieson 2018

K-means

- Randomly initialize *k* centers
  - $\mu^{(0)} = \mu_1^{(0)}, \ldots, \mu_k^{(0)}$
- **Classify**: Assign each point j∈{1,…N} to nearest center:
  - $C^{(t)}(j) \leftarrow \arg\min_i ||\mu_i - x_j||^2$
- **Recenter**: μ_i becomes centroid of its point:
  - $\mu_i^{(t+1)} \leftarrow \arg\min_\mu \sum_{j:C(j)=i} ||\mu - x_j||^2$
  - Equivalent to $\mu_i \leftarrow$ average of its points!

©Kevin Jamieson 2018

Does K-means converge??? Part 1

- Optimize potential function:

$$\min_{\mu} \min_{C} F(\mu, C) = \min_{\mu} \min_{C} \sum_{i=1}^{k} \sum_{j:C(j)=i} ||\mu_i - x_j||^2$$

- Fix μ, optimize C

©Kevin Jamieson 2018

Does K-means converge??? Part 2

- Optimize potential function:

$$\min_{\mu} \min_{C} F(\mu, C) = \min_{\mu} \min_{C} \sum_{i=1}^{k} \sum_{j:C(j)=i} ||\mu_i - x_j||^2$$

- Fix C, optimize μ

©Kevin Jamieson 2018

Does K-means converge??? Part 2

- Optimize potential function:

$$\min_{\mu} \min_{C} F(\mu, C) = \min_{\mu} \min_{C} \sum_{i=1}^{k} \sum_{j:C(j)=i} ||\mu_i - x_j||^2$$

- Fix C, optimize μ

Objective function decreases at every step $\implies$ No configuration repeated
Only $\binom{n}{k} \approx n^k$ unique configurations $\implies$ convergence in finite # iterations

©Kevin Jamieson 2018

Vector Quantization, Fisher Vectors

Vector Quantization (for compression)

1. Represent image as grid of patches
2. Run k-means on the patches to build code book
3. Represent each patch as a code word.

FIGURE 14.9. *Sir Ronald A. Fisher (1890 – 1962) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a 1024×1024 grayscale image at 8 bits per pixel. The center image is the result of 2 × 2 block VQ, using 200 code vectors, with a compression rate of 1.9 bits/pixel. The right image uses only four code vectors, with a compression rate of 0.50 bits/pixel*

Vector Quantization, Fisher Vectors

Vector Quantization (for compression)

1. Represent image as grid of patches
2. Run k-means on the patches to build code book
3. Represent each patch as a code word.

FIGURE 14.9. *Sir Ronald A. Fisher (1890 – 1962) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a 1024×1024 grayscale image at 8 bits per pixel. The center image is the result of 2 × 2 block VQ, using 200 code vectors, with a compression rate of 1.9 bits/pixel. The right image uses only four code vectors, with a compression rate of 0.50 bits/pixel*

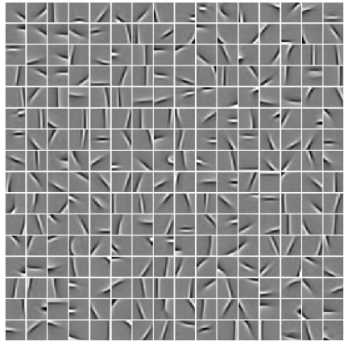
Vector Quantization, Fisher Vectors

Vector Quantization (for compression)

1. Represent image as grid of patches
2. Run k-means on the patches to build code book
3. Represent each patch as a code word.

FIGURE 14.9. *Sir Ronald A. Fisher (1890 – 1962) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a 1024×1024 grayscale image at 8 bits per pixel. The center image is the result of 2 × 2 block VQ, using 200 code vectors, with a compression rate of 1.9 bits/pixel. The right image uses only four code vectors, with a compression rate of 0.50 bits/pixel*

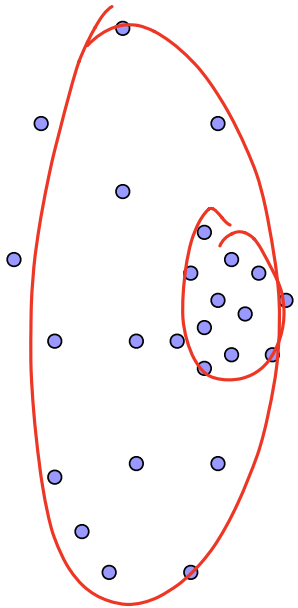
Typical output of k-means on patches



Similar reduced representation can be used as a feature vector

Coates, Ng, *Learning Feature Representations with K-means*, 2012

(One) bad case for k-means



- Clusters may overlap
- Some clusters may be “wider” than others

©Kevin Jamieson 2017

K-means summary

- Greedy algorithm that is sensitive to initialization
- Better initializations than “uniform at random”: Arthur and Vassilvitskii *k-means++: The Advantages of Careful Seeding*
- The centers can take values of data points or arbitrary vectors
- Extremely useful subroutine/pre-processing step for other algorithms
- If clusters vary widely in size, use more sophisticated method (mixture of Gaussians)

©Kevin Jamieson 2017