CS M148 –

Data Science Fundamentals

Lecture #16: Neural Networks & Unsupervised Learning

Baharan Mirzasoleiman

UCLA Computer Science

Announcements

HW 2

- Grades are posted
- Regrade requests close 3/6 11pm

HW 4 (last one!)

- Posted, due Monday March 7 at 2pm

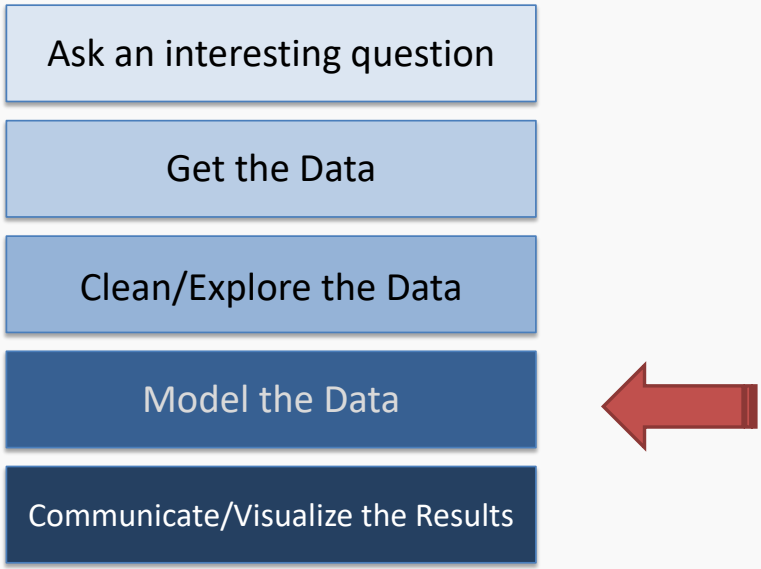
Project 3 (last one!)

- Posted, due Monday March 14 at 11:59pm

Let’s quickly review what we saw last time

Still here!

The Data Science Process



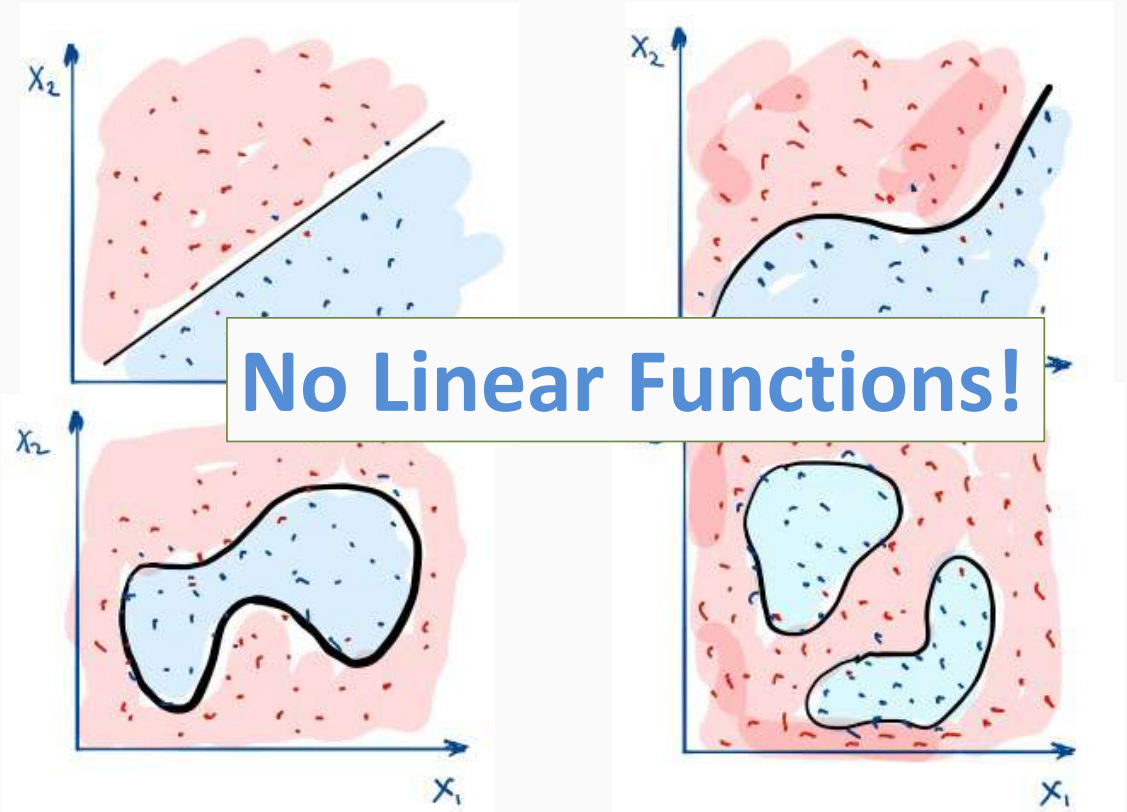
Classification & Regression

Neural Networks

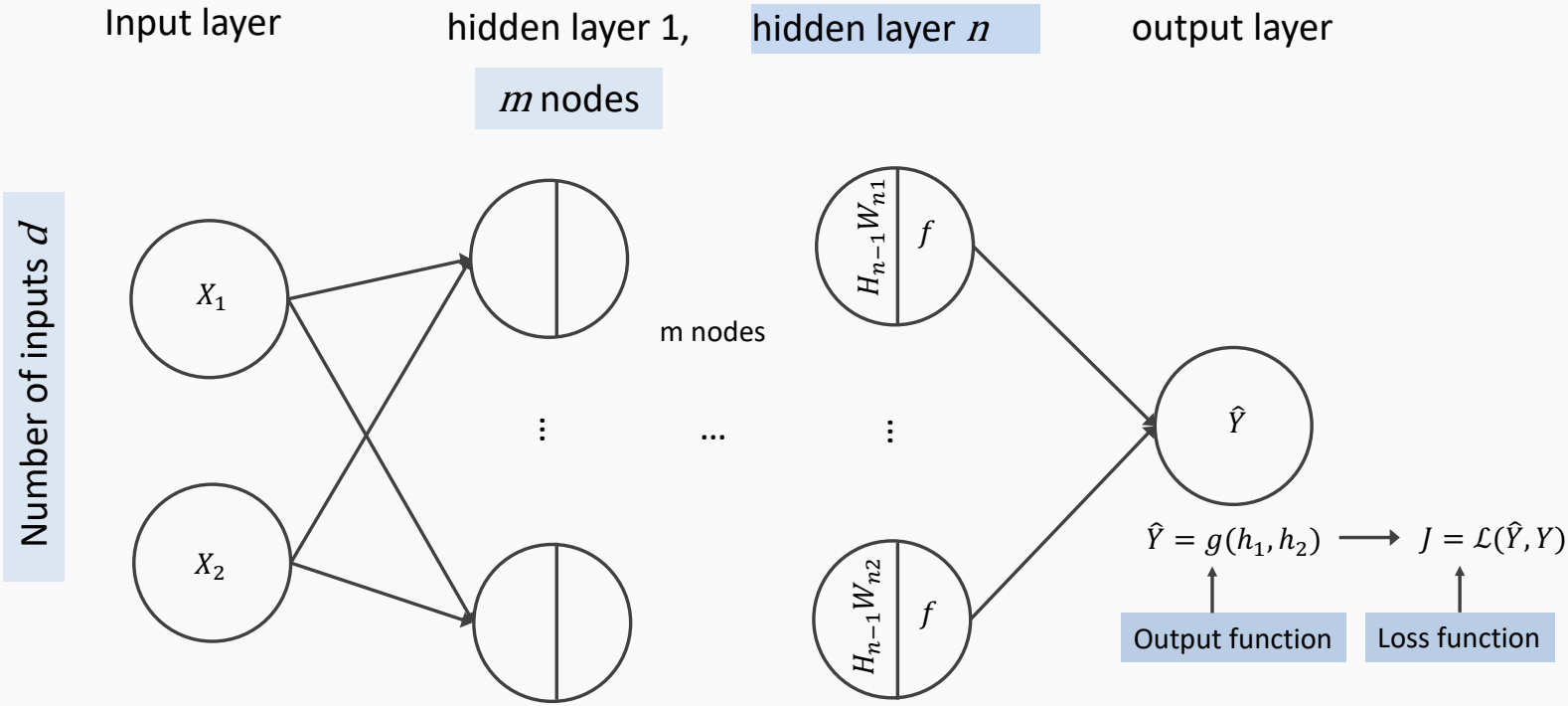
Outline

1. Review of basic concepts
2. Optimization
3. Regularization

So what's the big deal about Neural Networks?



Anatomy of artificial neural network (ANN)



Number of inputs is specified by the data

Loss Function

Do not need to design separate loss functions if we follow the probabilistic modeling approach, i.e. minimize the –ve likelihood function.

Examples:

- Distribution is **Normal** then –ve log-likelihood is **MSE** :

$$p(y_i|W;x_i) = \frac{1}{\sqrt{\{2\pi^2\sigma\}}} e^{-\frac{(y_i-\hat{y}_i)^2}{2\sigma^2}}$$

$$\mathcal{L}(W;X,Y) = \sum_i (y_i - \hat{y}_i)^2$$

- Distribution is **Bernoulli** then –ve log-likelihood is **Binary Cross-Entropy**:

$$p(y_i|W;x_i) = p_i^{y_i}(1-p_i)^{1-y_i}$$

$$\mathcal{L}(W;X,Y) = -\sum_i [y_i \log p_i + (1-y_i)\log(1-p_i)]$$

Loss Function

- For y that follow **Multinoulli,** then likelihood is:

$$p(y_i|W;x_i) = \prod_k p\,(y_i = k)^{\mathbb{I}(y_i=k)}$$

Cross-Entropy

$$\mathcal{L}(W;X,Y) = -\sum_i \sum_k \mathbb{I}(y_i = k)\log p(y_i = k)$$

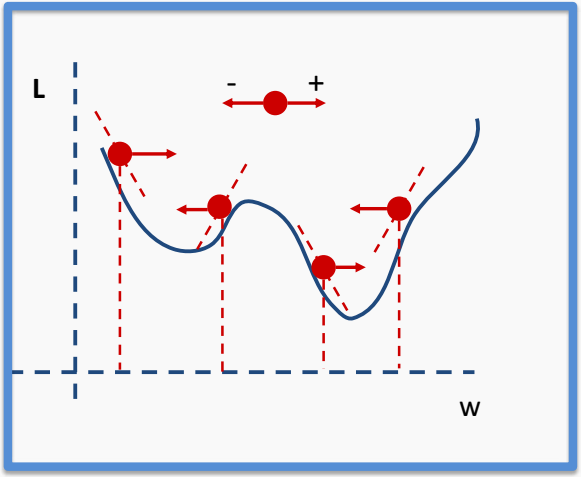
where k is the class and $\mathbb{I}(y_i = k)$ is the indicator function.

Optimizer: Learning/Training Neural Networks

Gradient Descent

- Algorithm for optimization of first order to finding a minimum of a function.
- It is an iterative method.
- *L* is decreasing much faster in the direction of the negative derivative.
- The learning rate is controlled by the magnitude of η.

$$w^{(i+1)} = w^{(i)} - \eta \frac{d\mathcal{L}}{dw}$$



Considerations

- **We still need to calculate the derivatives.**
- We need to know what is the learning rate or how to set it.
- Local vs global minima.
- The full likelihood function includes summing up all individual '*errors'.* Unless you are a statistician, sometimes this includes hundreds of thousands of examples.

Chain Rule

Chain rule for computing gradients:

$$y = g(x) \qquad z = f(y) = f\big(g(x)\big)$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

$$\boldsymbol{y} = g(\boldsymbol{x}) \qquad z = f(\boldsymbol{y}) = f\big(g(\boldsymbol{x})\big)$$

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}$$

- For longer chains:

$$\frac{\partial z}{\partial x_i} = \sum_{j_1} \cdots \sum_{j_m} \frac{\partial z}{\partial y_{j_1}} \cdots \frac{\partial y_{j_m}}{\partial x_i}$$

Logistic Regression derivatives

For logistic regression, the –ve log of the likelihood is:

$$\mathcal{L} = \sum_i \mathcal{L}_i = -\sum_i \log L_i = -\sum_i [y_i \log p_i + (1 - y_i) \log(1 - p_i)]$$

$$\mathcal{L}_i = -y_i \log \frac{1}{1 + e^{-W^T X}} - (1 - y_i) \log(1 - \frac{1}{1 + e^{-W^T X}})$$

To simplify the analysis let us split it into two parts,

$$\mathcal{L}_i = \mathcal{L}_i^A + \mathcal{L}_i^B$$

So the derivative with respect to *W* is:

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_i \frac{\partial \mathcal{L}_i}{\partial W} = \sum_i (\frac{\partial \mathcal{L}_i^A}{\partial W} + \frac{\partial \mathcal{L}_i^B}{\partial W})$$

$$\mathcal{L}_i^A = -y_i \log \frac{1}{1 + e^{-W^T X}}$$

Variables	Partial derivatives	Partial derivatives
$\xi_1 = -W^T X$	$\frac{\partial \xi_1}{\partial W} = -X$	$\frac{\partial \xi_1}{\partial W} = -X$
$\xi_2 = e^{\xi_1} = e^{-W^T X}$	$\frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1}$	$\frac{\partial \xi_2}{\partial \xi_1} = e^{-W^T X}$
$\xi_3 = 1 + \xi_2 = 1 + e^{-W^T X}$	$\frac{\partial \xi_3}{\partial \xi_2} = 1$	$\frac{\partial \xi_3}{\partial \xi_2} = 1$
$\xi_4 = \frac{1}{\xi_3} = \frac{1}{1 + e^{-W^T X}} = p$	$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\xi_3^2}$	$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{(1 + e^{-W^T X})^2}$
$\xi_5 = \log \xi_4 = \log p = \log \frac{1}{1 + e^{-W^T X}}$	$\frac{\partial \xi_5}{\partial \xi_4} = \frac{1}{\xi_4}$	$\frac{\partial \xi_5}{\partial \xi_4} = 1 + e^{-W^T X}$
$\mathcal{L}_i^A = -y\xi_5$	$\frac{\partial \mathcal{L}}{\partial \xi_5} = -y$	$\frac{\partial \mathcal{L}}{\partial \xi_5} = -y$
$\frac{\partial \mathcal{L}_i^A}{\partial W} = \frac{\partial \mathcal{L}_i}{\partial \xi_5} \frac{\partial \xi_5}{\partial \xi_4} \frac{\partial \xi_4}{\partial \xi_3} \frac{\partial \xi_3}{\partial \xi_2} \frac{\partial \xi_2}{\partial \xi_1} \frac{\partial \xi_1}{\partial W}$		$\frac{\partial \mathcal{L}_i^A}{\partial W} = -yXe^{-W^T X} \frac{1}{(1 + e^{-W^T X})}$

$$-X\sigma(h)\big(1 - \sigma(h)\big)\left[y\frac{1}{p}\right]$$

$$\mathcal{L}_i^B = -(1-y_i)\log[1-\frac{1}{1+e^{-W^TX}}]$$

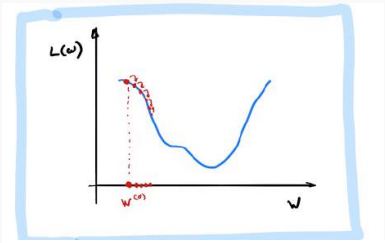
Variables	derivatives	Partial derivatives wrt to X,W
$\xi_1 = -W^TX$	$\frac{\partial \xi_1}{\partial W} = -X$	$\frac{\partial \xi_1}{\partial W} = -X$
$\xi_2 = e^{\xi_1} = e^{-W^TX}$	$\frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1}$	$\frac{\partial \xi_2}{\partial \xi_1} = e^{-W^TX}$
$\xi_3 = 1+\xi_2 = 1+e^{-W^TX}$	$\frac{\partial \xi_3}{\partial \xi_2} = 1$	$\frac{\partial \xi_3}{\partial 2}$ =1
$\xi_4 = \frac{1}{\xi_3} = \frac{1}{1+e^{-W^TX}} = p$	$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\xi_3^2}$	$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\left(1+e^{-W^TX}\right)^2}$
$\xi_5 = 1-\xi_4 = 1-\frac{1}{1+e^{-W^TX}}$	$\frac{\partial \xi_5}{\partial \xi_4} = -1$	$\frac{\partial \xi_5}{\partial \xi_4}$ =-1
$\xi_6 = \log \xi_5 = \log(1-p) = \log\frac{1}{1+e^{-W^TX}}$	$\frac{\partial \xi_6}{\partial \xi_5} = \frac{1}{\xi_5}$	$\frac{\partial \xi_6}{\partial \xi_5} = \frac{1+e^{-W^TX}}{e^{-W^TX}}$
$\mathcal{L}_i^B = (1-y)\xi_6$	$\frac{\partial \mathcal{L}}{\partial \xi_6} = 1-y$	$\frac{\partial \mathcal{L}}{\partial \xi_6} = 1-y$
$\frac{\partial \mathcal{L}_i^B}{\partial W} = \frac{\partial \mathcal{L}_i^B}{\partial \xi_6}\frac{\partial \xi_6}{\partial \xi_5}\frac{\partial \xi_5}{\partial \xi_4}\frac{\partial \xi_4}{\partial \xi_3}\frac{\partial \xi_3}{\partial \xi_2}\frac{\partial \xi_2}{\partial \xi_1}\frac{\partial \xi_1}{\partial W}$		$\frac{\partial \mathcal{L}_i^B}{\partial W} = (1-y)X\frac{1}{\left(1+e^{-W^TX}\right)}$

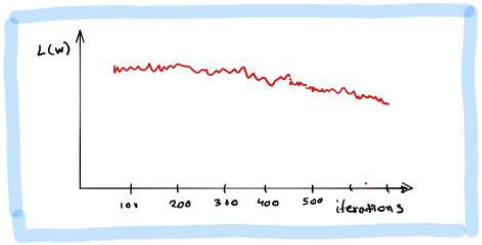
$$-X\sigma(h)\big(1-\sigma(h)\big)[(1-y)\frac{1}{1-p}]$$

Considerations

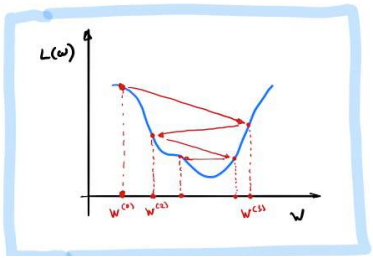
- We still need to calculate the derivatives.
- **We need to know what is the learning rate or how to set it.**
- Local vs global minima.
- The full likelihood function includes summing up all individual '*errors'.* Unless you are a statistician, sometimes this includes hundreds of thousands of examples.

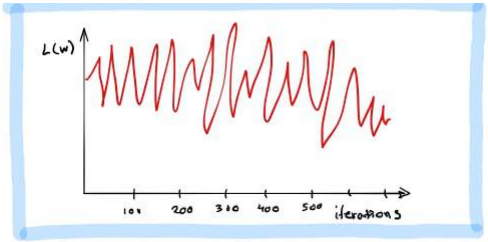
How can we tell when gradient descent is converging? We visualize the loss function at each step of gradient descent. This is called the **trace plot**.



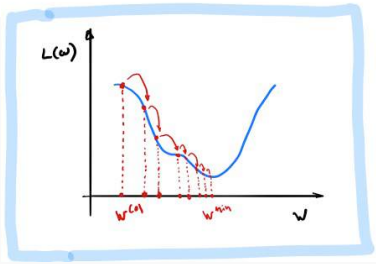


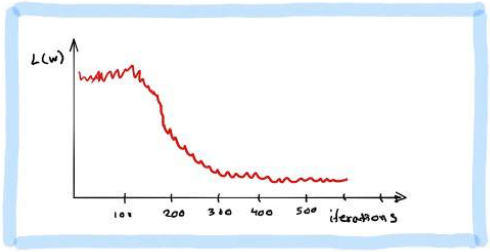
While the loss is decreasing throughout training, it does not look like descent hit the bottom.





Loss is mostly oscillating between values rather than converging.





The loss has decreased significantly during training. Towards the end, the loss stabilizes and it can't decrease further.

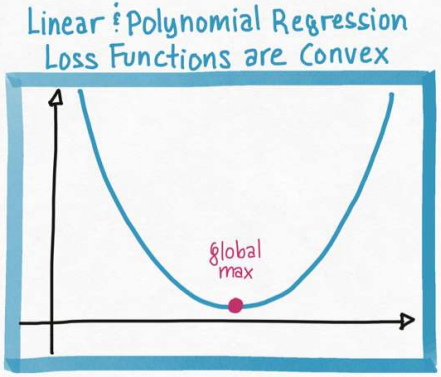
Considerations

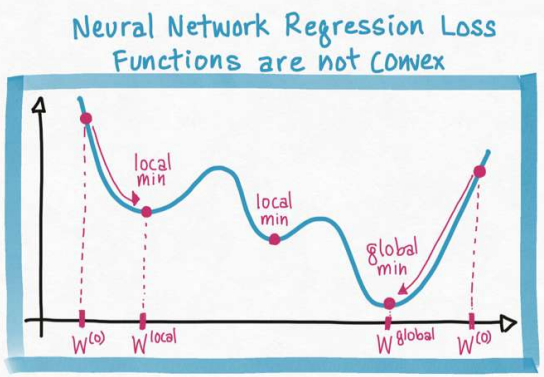
- We still need to calculate the derivatives.
- We need to know what is the learning rate or how to set it.
- **Local vs global minima.**
- The full likelihood function includes summing up all individual '*errors'.* Unless you are a statistician, sometimes this includes hundreds of thousands of examples.

Local vs Global Minima

If we choose η correctly, then gradient descent will converge to a stationary point. But will this point be a **global minimum**?

If the function is convex then the stationary point will be a global minimum.





Local vs Global Minima

No guarantee that we get the global minimum.

Question: What would be a good strategy?

- Random restarts
- Add noise to the loss function

Considerations

- We still need to calculate the derivatives.
- We need to know what is the learning rate or how to set it.
- Local vs global minima.
- **The full likelihood function includes summing up all individual '*errors*'. Unless you are a statistician, sometimes this includes hundreds of thousands of examples.**

Batch and Stochastic Gradient Descent

$$\mathcal{L} = -\sum_i [y_i \log p_i + (1 - y_i) \log(1 - p_i)]$$

Instead of using all the examples for every step, use a subset of them (batch).

For each iteration *k*, use the following loss function to derive the derivatives:

$$\mathcal{L}^k = -\sum_{i \in b^k} [y_i \log p_i + (1 - y_i) \log(1 - p_i)]$$

which is an **approximation** to the full loss function.

DATA

Calculate $L_1 \Rightarrow \frac{\partial L_1}{\partial w} \Rightarrow w^* = w - \eta \frac{\partial L_1}{\partial w}$

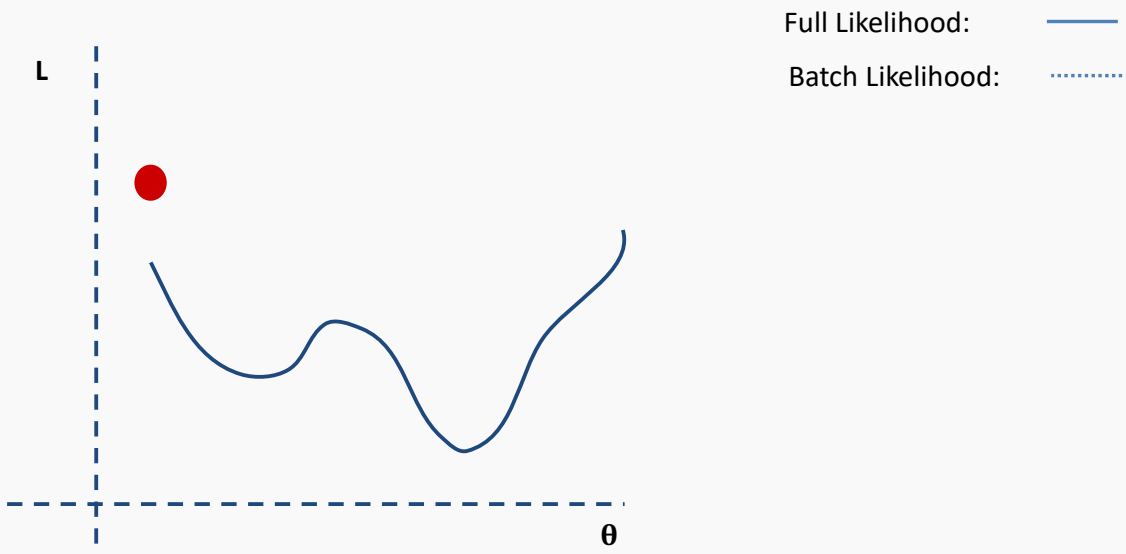
Calculate $L_2 \Rightarrow \frac{\partial L_2}{\partial w} \Rightarrow w^* = w - \eta \frac{\partial L_2}{\partial w}$

$L_B \Rightarrow \frac{\partial L_B}{\partial w} \Rightarrow w^* = w - \eta \frac{\partial L_B}{\partial w}$

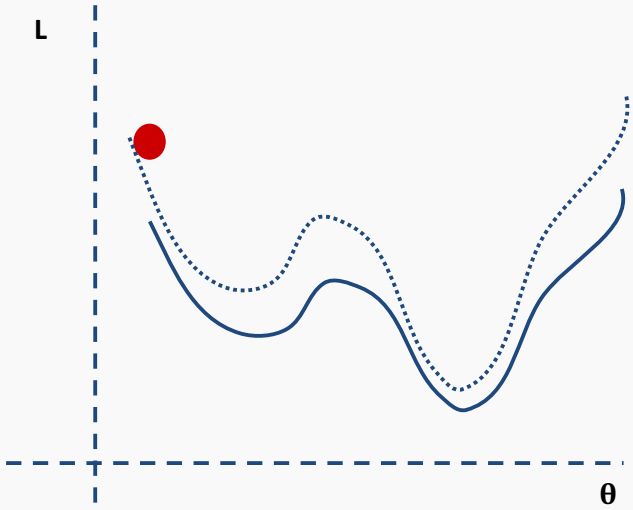
COMPLETE DATA $\Rightarrow$ ONE EPOCH

RESHUFFLE DATA AND REPEAT

Batch and Stochastic Gradient Descent



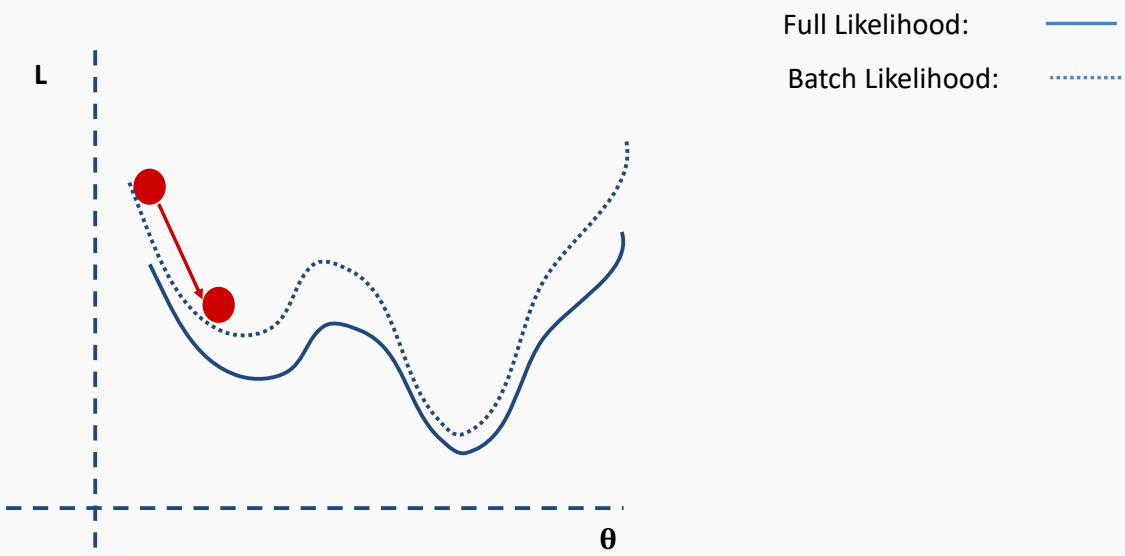
Batch and Stochastic Gradient Descent



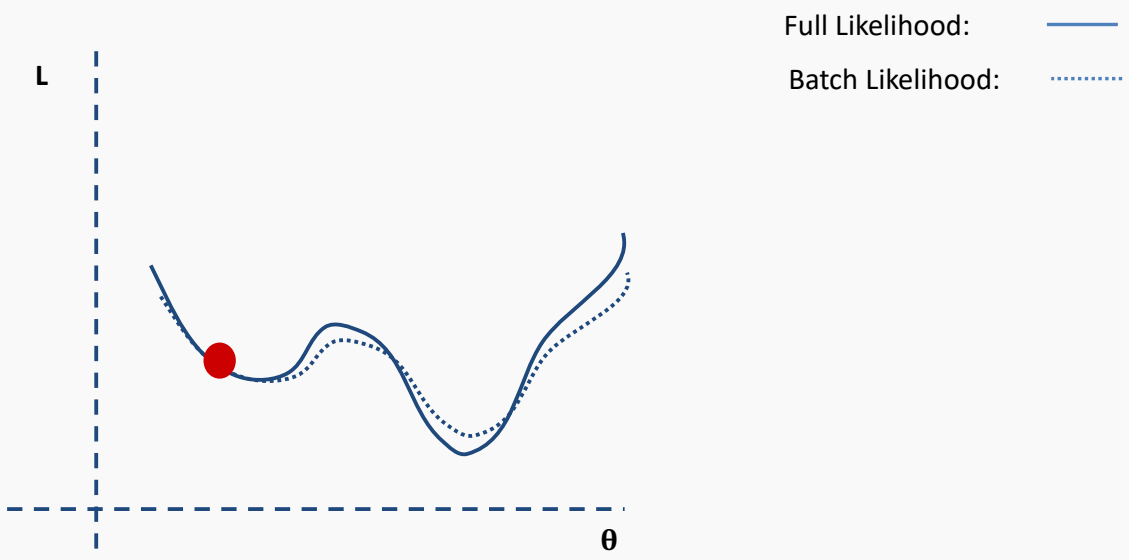
Full Likelihood:

Batch Likelihood:

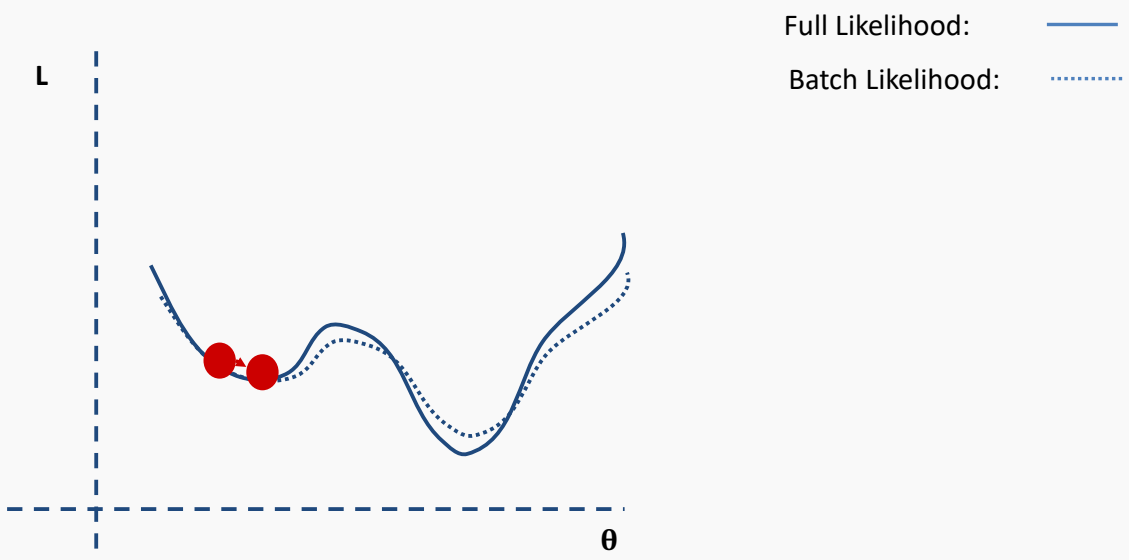
Batch and Stochastic Gradient Descent



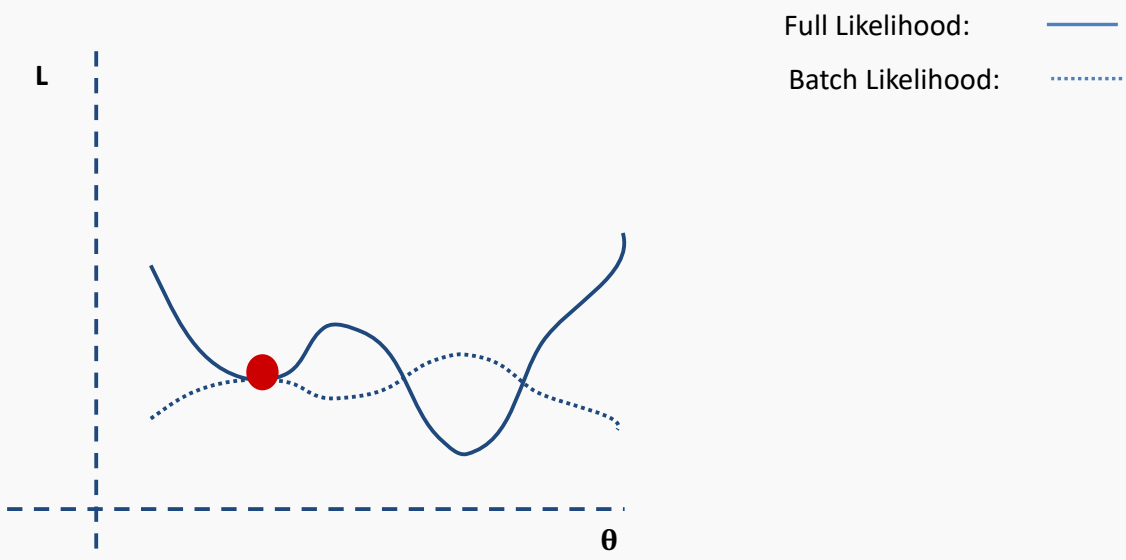
Batch and Stochastic Gradient Descent



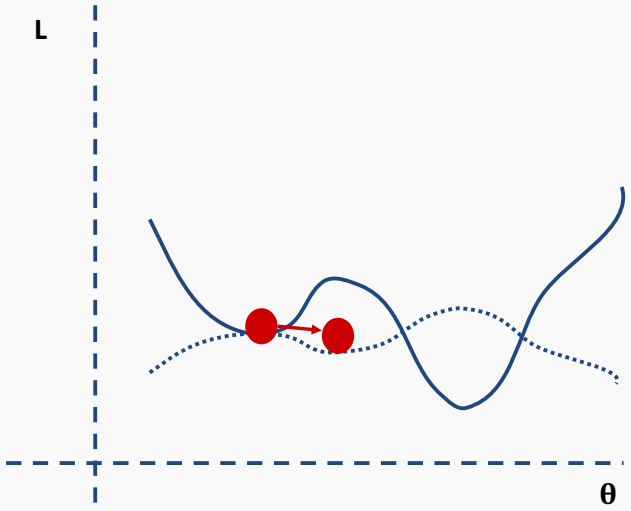
Batch and Stochastic Gradient Descent



Batch and Stochastic Gradient Descent



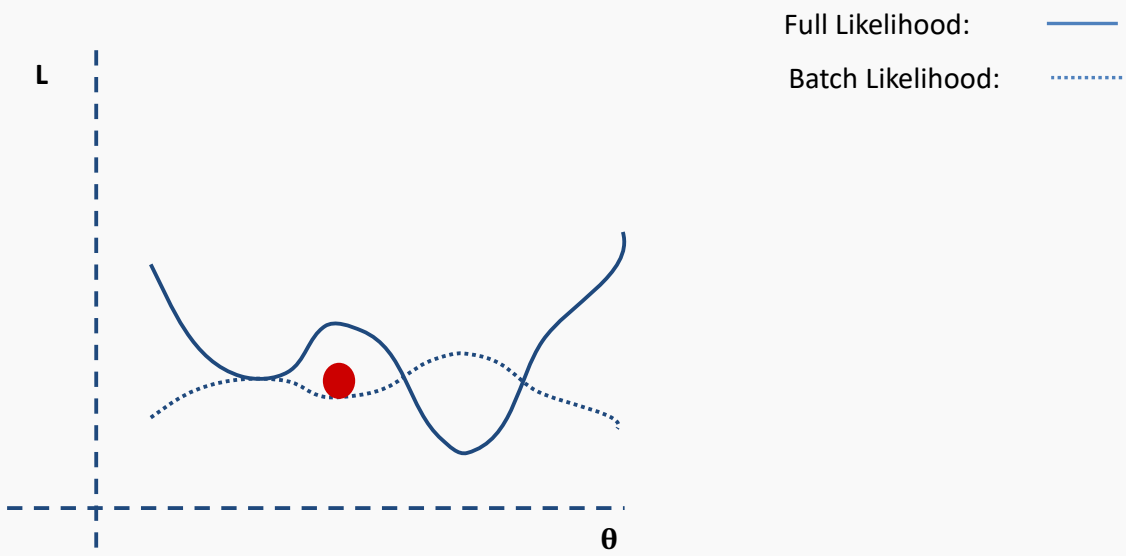
Batch and Stochastic Gradient Descent



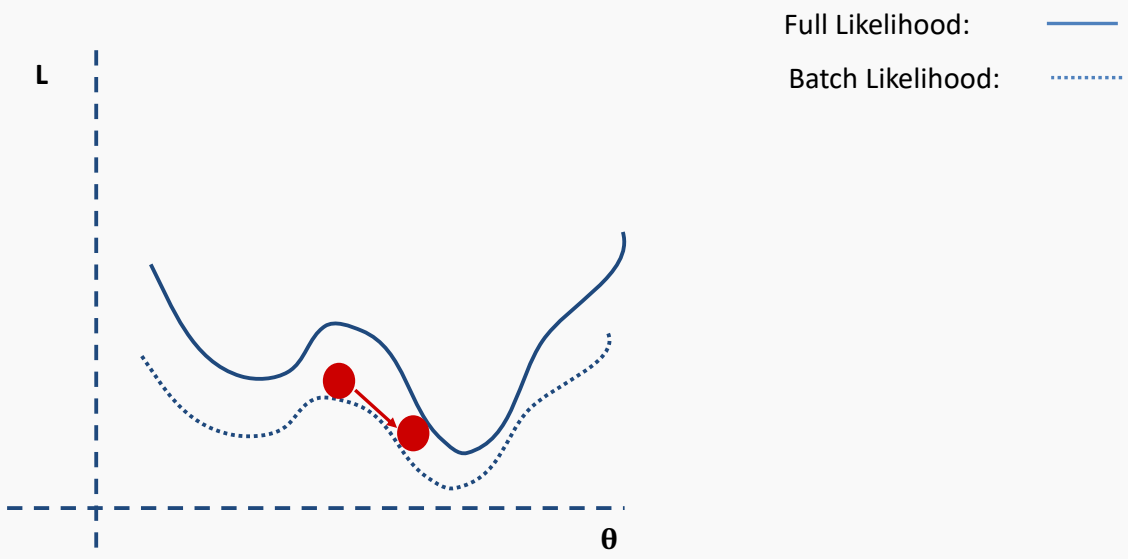
Full Likelihood:

Batch Likelihood:

Batch and Stochastic Gradient Descent



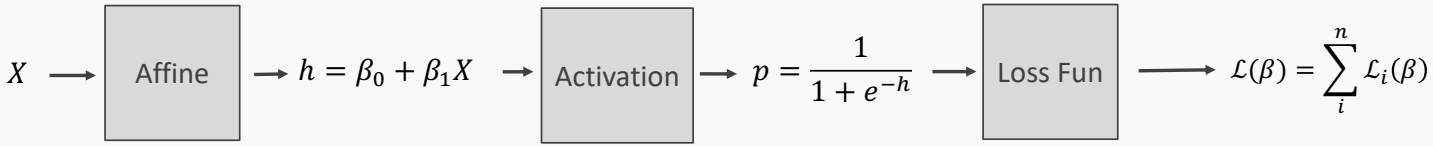
Batch and Stochastic Gradient Descent



Backpropagation

Backpropagation: Logistic Regression Revisited

$$\mathcal{L} = -\sum_i [y_i \log p_i + (1 - y_i) \log(1 - p_i)]$$



$$\frac{\partial \mathcal{L}}{\partial p} \quad \frac{\partial \mathcal{L}}{\partial p}\frac{\partial p}{\partial h} \quad \frac{\partial \mathcal{L}}{\partial p}\frac{\partial p}{\partial h}\frac{\partial h}{\partial \beta}$$

$$\frac{\partial h}{\partial \beta_1} = X, \frac{d\mathcal{L}}{d\beta_0} = 1 \qquad \frac{\partial p}{\partial h} = \sigma(h)(1 - \sigma(h)) \qquad \frac{\partial \mathcal{L}}{\partial p} = -y\frac{1}{p} - (1 - y)\frac{1}{1 - p}$$

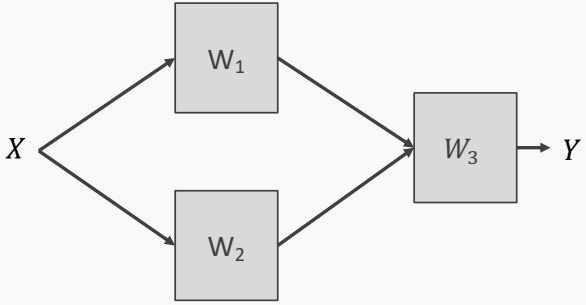
$$\frac{\partial \mathcal{L}}{\partial \beta_1} = -X\sigma(h)\big(1 - \sigma(h)\big)[y\frac{1}{p} + (1 - y)\frac{1}{1 - p}]$$

$$\frac{\partial \mathcal{L}}{\partial \beta_0} = -\sigma(h)\big(1 - \sigma(h)\big)[\, y\frac{1}{p} + (1 - y)\frac{1}{1 - p}]$$

Backpropagation

1. Derivatives need to be evaluated at some values of *X, y,* and *W*.
2. But since we have an expression, we can build a function that takes as input *X, y, W,* and returns the derivatives, and then we can use gradient descent to update.
3. This approach works well but it does not generalize. For example if the network is changed, we need to write a new function to evaluate the derivatives.

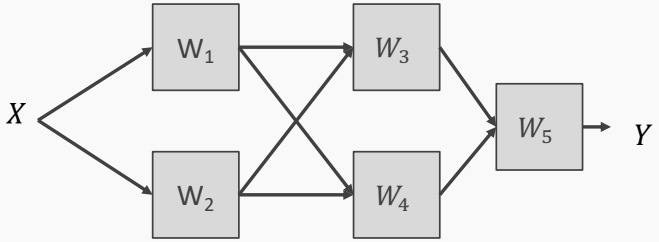
For example, this network will need a different function for the derivatives



Backpropagation

1. Derivatives need to be evaluated at some values of *X, y,* and *W*.
2. But since we have an expression, we can build a function that takes as input *X, y, W,* and returns the derivatives, and then we can use gradient descent to update.
3. This approach works well but it does not generalize. For example if the network is changed, we need to write a new function to evaluate the derivatives.

For example, this network will need a different function for the derivatives



Backpropagation

Need to find a formalism to calculate the derivatives of the loss w.r.t. weights that is:

1. Flexible enough that adding a node or a layer or changing something in the network will not require re-deriving the functional form from scratch.
2. It is exact.
3. It is computationally efficient.

Hints:

1. Remember we only need to evaluate the derivatives at X_i, y_i and $W^{(k)}$.
2. We should take advantage of the chain rule we learned before

Idea 1: Evaluate the derivative at: $X=\{3\}, y=1, W=3$

Variables	derivatives	Value of the variable	Value of the partial derivative	$\frac{d\xi_n}{dW}$
$\xi_1 = -W^T X$	$\frac{\partial \xi_1}{\partial W} = -X$	-9	-3	-3
$\xi_2 = e^{\xi_1} = e^{-W^T X}$	$\frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1}$	e^{-9}	e^{-9}	$-3e^{-9}$
$\xi_3 = 1 + \xi_2 = 1 + e^{-W^T X}$	$\frac{\partial \xi_3}{\partial \xi_2} = 1$	$1+e^{-9}$	1	$-3e^{-9}$
$\xi_4 = \frac{1}{\xi_3} = \frac{1}{1 + e^{-W^T X}} = p$	$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\xi_3^2}$	$\frac{1}{1 + e^{-9}}$	$\left(\frac{1}{1 + e^{-9}}\right)^2$	$-3e^{-9}\left(\frac{1}{1+e^{-9}}\right)^2$
$\xi_5 = \log \xi_4 = \log p = \log \frac{1}{1 + e^{-W^T}}$	$\frac{\partial \xi_5}{\partial \xi_4} = \frac{1}{\xi_4}$	$\log \frac{1}{1 + e^{-9}}$	$1 + e^{-9}$	$-3e^{-9}\left(\frac{1}{1+e^{-9}}\right)$
$\mathcal{L}_i^A = -y\xi_5$	$\frac{\partial \mathcal{L}}{\partial \xi_5} = -y$	$-\log \frac{1}{1 + e^{-9}}$	-1	$3e^{-9}\left(\frac{1}{1+e^{-9}}\right)$
$\frac{\partial \mathcal{L}_i^A}{\partial W} = \frac{\partial \mathcal{L}_i}{\partial \xi_5}\frac{\partial \xi_5}{\partial \xi_4}\frac{\partial \xi_4}{\partial \xi_3}\frac{\partial \xi_3}{\partial \xi_2}\frac{\partial \xi_2}{\partial \xi_1}\frac{\partial \xi_1}{\partial W}$			-3	0.00037018372

Basic functions

We still need to derive **derivatives** ☹

Variables	derivatives	Value of the variable	Value of the partial derivative	$\frac{d\xi_n}{dW}$
$\xi_1 = -W^T X$	$\frac{\partial \xi_1}{\partial W} = -X$	-9	-3	-3
$\xi_2 = e^{\xi_1} = e^{-W^T X}$	$\frac{\partial \xi_2}{d\partial \xi_1} = e^{\xi_1}$	e^{-9}	e^{-9}	$-3e^{-9}$
$\xi_3 = 1 + \xi_2 = 1 + e^{-W^T X}$	$\frac{\partial \xi_3}{\partial \xi_2} = 1$	$1+e^{-9}$	1	$-3e^{-9}$
$\xi_4 = \frac{1}{\xi_3} = \frac{1}{1 + e^{-W^T X}} = p$	$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\xi_3^2}$	$\frac{1}{1 + e^{-9}}$	$\left(\frac{1}{1 + e^{-9}}\right)^2$	$-3e^{-9}\left(\frac{1}{1+e^{-9}}\right)^2$
$\xi_5 = \log \xi_4 = \log p \ = \log \frac{1}{1 + e^{-W^T X}}$	$\frac{\partial \xi_5}{\partial \xi_4} = \frac{1}{\xi_4}$	$\log \frac{1}{1 + e^{-9}}$	$1 + e^{-9}$	$-3e^{-9}\left(\frac{1}{1+e^{-9}}\right)$
$\mathcal{L}_i^A = -y\xi_5$	$\frac{\partial \mathcal{L}}{\partial \xi_5} = -y$	$-\log \frac{1}{1 + e^{-9}}$	-1	$3e^{-9}\left(\frac{1}{1+e^{-9}}\right)$
$\frac{\partial \mathcal{L}_i^A}{\partial W} = \frac{\partial \mathcal{L}_i}{\partial \xi_5}\frac{\partial \xi_5}{\partial \xi_4}\frac{\partial \xi_4}{\partial \xi_3}\frac{\partial \xi_3}{\partial \xi_2}\frac{\partial \xi_2}{\partial \xi_1}\frac{\partial \xi_1}{\partial W}$			-3	0.00037018372

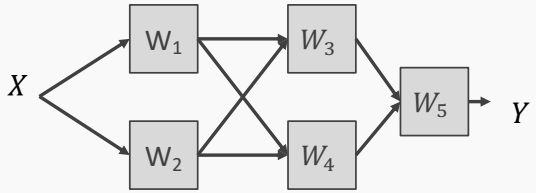
Basic functions

Notice though those are basic functions that everybody can do!

$\xi_0 = X$	$\frac{\partial \xi_0}{\partial X} = 1$	`def x0(x):` `  return X`	`def derx0():` `  return 1`
$\xi_1 = -W^T \xi_0$	$\frac{\partial \xi_1}{\partial W} = -X$	`def x1(a,x):` `  return -a*X`	`def derx1(a,x):` `  return -a`
$\xi_2 = \mathrm{e}^{\xi_1}$	$\frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1}$	`def x2(x):` `  return np.exp(x)`	`def derx2(x):` `  return np.exp(x)`
$\xi_3 = 1 + \xi_2$	$\frac{\partial \xi_3}{\partial \xi_2} = 1$	`def x3(x):` `  return 1+x`	`def derx3(x):` `  return 1`
$\xi_4 = \frac{1}{\xi_3}$	$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\xi_3^2}$	`def der1(x):` `  return 1/(x)`	`def derx4(x):` `  return  -(1/x)**(2)`
$\xi_5 = \log \xi_4$	$\frac{\partial \xi_5}{\partial \xi_4} = \frac{1}{\xi_4}$	`def der1(x):` `  return np.log(x)`	`def derx5(x)` `  return 1/x`
$\mathcal{L}_i^A = -y\xi_5$	$\frac{\partial \mathcal{L}}{\partial \xi_5} = -y$	`def der1(y,x):` `  return -y*x`	`def derL(y):` `  return -y`

Putting it altogether

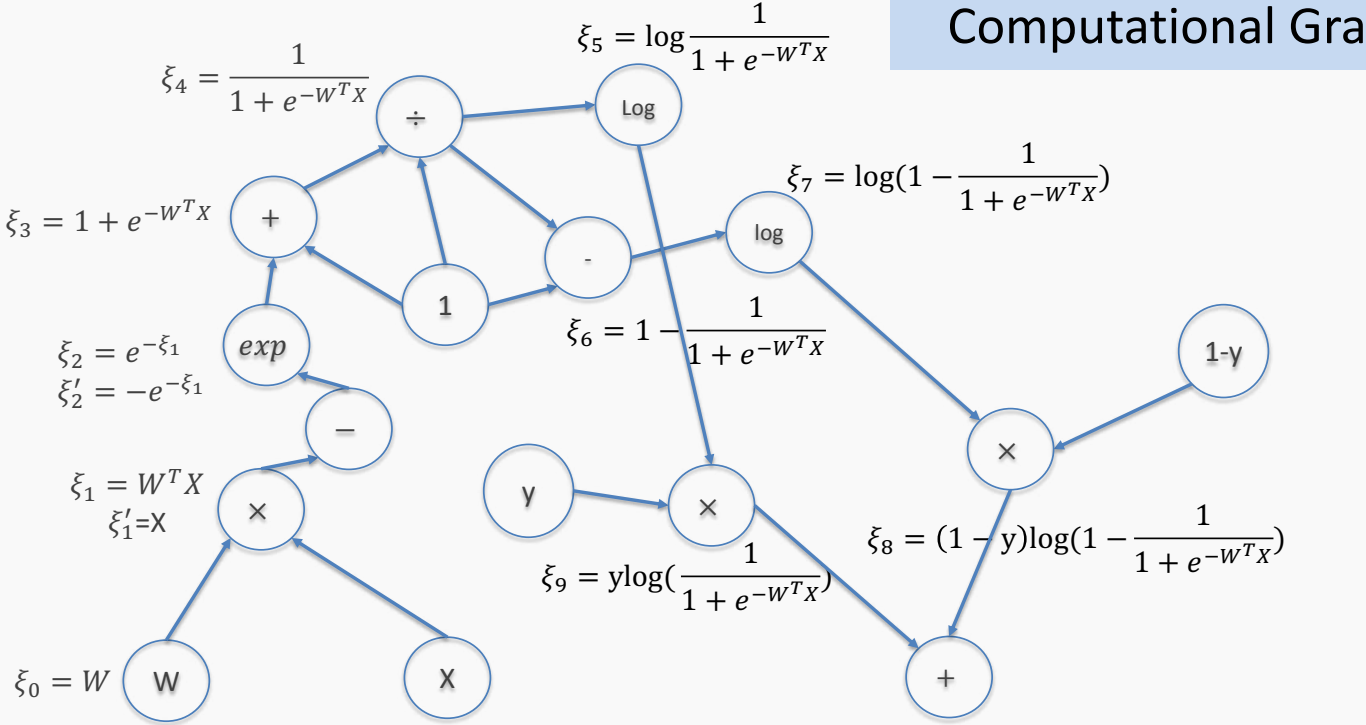
1. We specify the network structure



2. Following the computational graph …

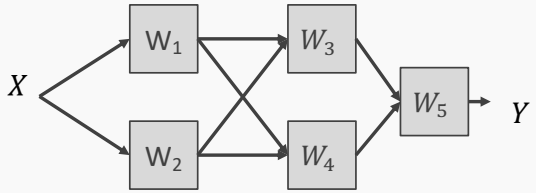
What is computational graph?

Computational Graph



Putting it altogether

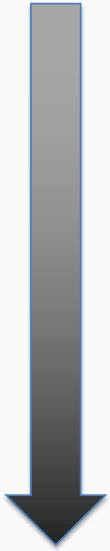
1. We specify the network structure



- Envision the computational graph (only needed for the reverse mode).
- At each node of the graph we build two functions: the evaluation of the variable and its partial derivative with respect to the previous variable (as shown in the table 3 slides back)
- Now we can either go forward or backward depending on the situation. In general, forward is easier to implement and to understand. The difference is clearer when there are multiple nodes per layer.

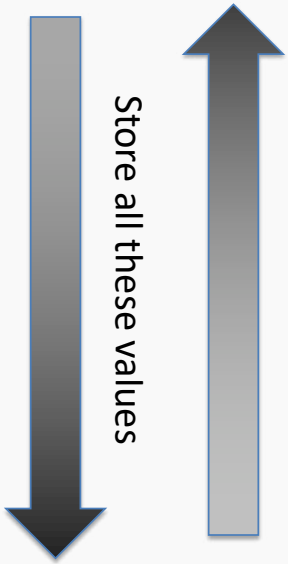
Forward mode: Evaluate the derivative at: $X=\{3\}, y=1, W=3$

Variables	derivatives	Value of the variable	Value of the partial derivative	$\frac{d\mathcal{L}}{d\xi_n}$
$\xi_1 = -W^T X$	$\frac{\partial \xi_1}{\partial W} = -X$	-9	-3	-3
$\xi_2 = e^{\xi_1} = e^{-W^T X}$	$\frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1}$	e^{-9}	e^{-9}	$-3e^{-9}$
$\xi_3 = 1 + \xi_2 = 1 + e^{-W^T X}$	$\frac{\partial \xi_3}{\partial \xi_2} = 1$	$1+e^{-9}$	1	$-3e^{-9}$
$\xi_4 = \frac{1}{\xi_3} = \frac{1}{1+e^{-W^T X}} = p$	$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\xi_3^2}$	$\frac{1}{1+e^{-9}}$	$\left(\frac{1}{1+e^{-9}}\right)^2$	$-3e^{-9}\left(\frac{1}{1+e^{-9}}\right)^2$
$\xi_5 = \log \xi_4 = \log p = \log \frac{1}{1+e^{-W^T X}}$	$\frac{\partial \xi_5}{\partial \xi_4} = \frac{1}{\xi_4}$	$\log \frac{1}{1+e^{-9}}$	$1+e^{-9}$	$-3e^{-9}\left(\frac{1}{1+e^{-9}}\right)$
$\mathcal{L}_i^A = -y\xi_5$	$\frac{\partial \mathcal{L}}{\partial \xi_5} = -y$	$-\log \frac{1}{1+e^{-9}}$	-1	$3e^{-9}\left(\frac{1}{1+e^{-9}}\right)$
$\frac{\partial \mathcal{L}_i^A}{\partial W} = \frac{\partial \mathcal{L}_i}{\partial \xi_5}\frac{\partial \xi_5}{\partial \xi_4}\frac{\partial \xi_4}{\partial \xi_3}\frac{\partial \xi_3}{\partial \xi_2}\frac{\partial \xi_2}{\partial \xi_1}\frac{\partial \xi_1}{\partial W}$			-3	0.00037018372



Backward mode: Evaluate the derivative at: $X=\{3\}, y=1, W=3$

Variables	derivatives	Value of the variable	Value of the partial derivative
$\xi_1 = -W^T X$	$\frac{\partial \xi_1}{\partial W} = -X$	-9	-3
$\xi_2 = e^{\xi_1} = e^{-W^T X}$	$\frac{\partial \xi_2}{\partial \xi_1} = e^{\xi_1}$	e^{-9}	e^{-9}
$\xi_3 = 1 + \xi_2 = 1 + e^{-W^T X}$	$\frac{\partial \xi_3}{\partial \xi_2} = 1$	$1+e^{-9}$	1
$\xi_4 = \frac{1}{\xi_3} = \frac{1}{1+e^{-W^T X}} = p$	$\frac{\partial \xi_4}{\partial \xi_3} = -\frac{1}{\xi_3^2}$	$\frac{1}{1+e^{-9}}$	$\left(\frac{1}{1+e^{-9}}\right)^2$
$\xi_5 = \log \xi_4 = \log p = \log \frac{1}{1+e^{-W^T X}}$	$\frac{\partial \xi_5}{\partial \xi_4} = \frac{1}{\xi_4}$	$\log \frac{1}{1+e^{-9}}$	$1+e^{-9}$
$\mathcal{L}_i^A = -y\xi_5$	$\frac{\partial \mathcal{L}}{\partial \xi_5} = -y$	$-\log \frac{1}{1+e^{-9}}$	-1
$\frac{\partial \mathcal{L}_i^A}{\partial W} = \frac{\partial \mathcal{L}_i}{\partial \xi_5}\frac{\partial \xi_5}{\partial \xi_4}\frac{\partial \xi_4}{\partial \xi_3}\frac{\partial \xi_3}{\partial \xi_2}\frac{\partial \xi_2}{\partial \xi_1}\frac{\partial \xi_1}{\partial W}$			Type equation here.



Check this: https://autoed.herokuapp.com

Outline

Regularization of NN

- Norm Penalties
- Early Stopping
- Data Augmentation
- Dropout

Regularization

Regularization is any modification we make to a learning algorithm that is intended to **reduce its generalization error** but not its training error.

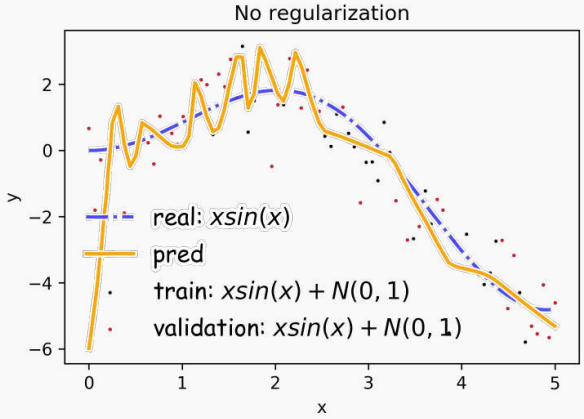
Outline

Regularization of NN

- **Norm Penalties**
- Early Stopping
- Data Augmentation
- Dropout

Overfitting

Fitting a deep neural network with 5 layers and 100 neurons per layer can lead to a very good prediction on the training set but poor prediction on validations set.



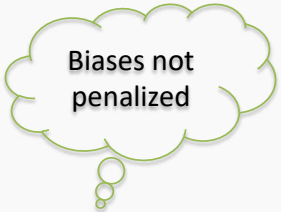
Norm Penalties

We used to optimize:

$$L(W; X, y)$$

Change to …

$$L_R(W; X, y) = L(W; X, y) + \alpha\Omega(W)$$



L_2 regularization:

- Weights decay
- MAP estimation with Gaussian prior

$$\Omega(W) = \frac{1}{2} \| W \|_2^2$$

L_1 regularization:

- encourages sparsity
- MAP estimation with Laplacian prior

$$\Omega(W) = \frac{1}{2} \| W \|_1$$

Norm Penalties

We used to optimize:

$$W^{(i+1)} = W^{(i)} - \lambda \frac{\partial L}{\partial W}$$

Change to:

$$L_R(W; X, y) = L(W; X, y) + \frac{1}{2}\alpha \parallel W \parallel_2^2$$

$$W^{(i+1)} = W^{(i)} - \lambda \frac{\partial L}{\partial W} - \lambda\alpha\, W^{(i)}$$

Weights decay in proportion to size

Biases not penalized

L_2 regularization:

- **Decay of weights**
- MAP estimation with Gaussian prior

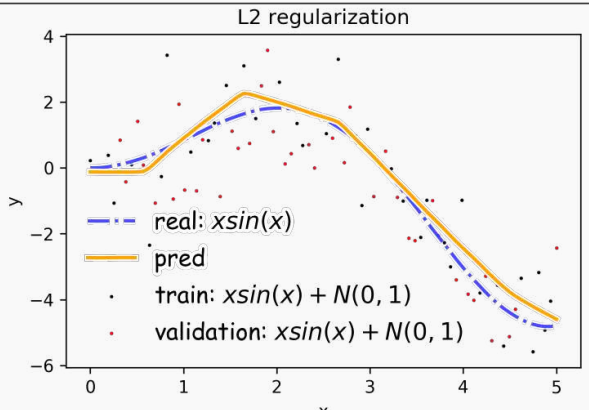
$$\Omega(W) = \frac{1}{2} \parallel W \parallel_2^2$$

L_1 regularization:

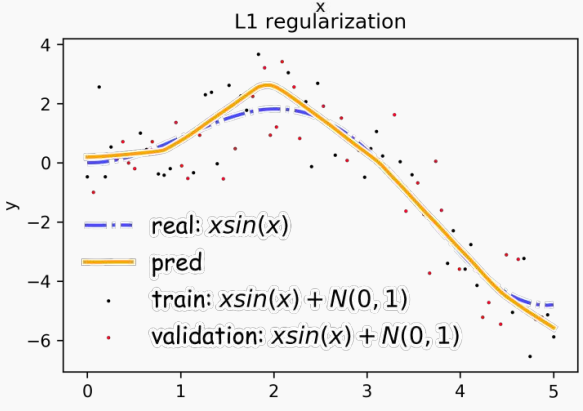
- encourages sparsity
- MAP estimation with Laplacian prior

$$\Omega(W) = \frac{1}{2} \parallel W \parallel_1$$

Norm Penalties



$$\Omega(W) = \frac{1}{2} \| W \|_2^2$$



$$\Omega(W) = \frac{1}{2} \| W \|_1$$

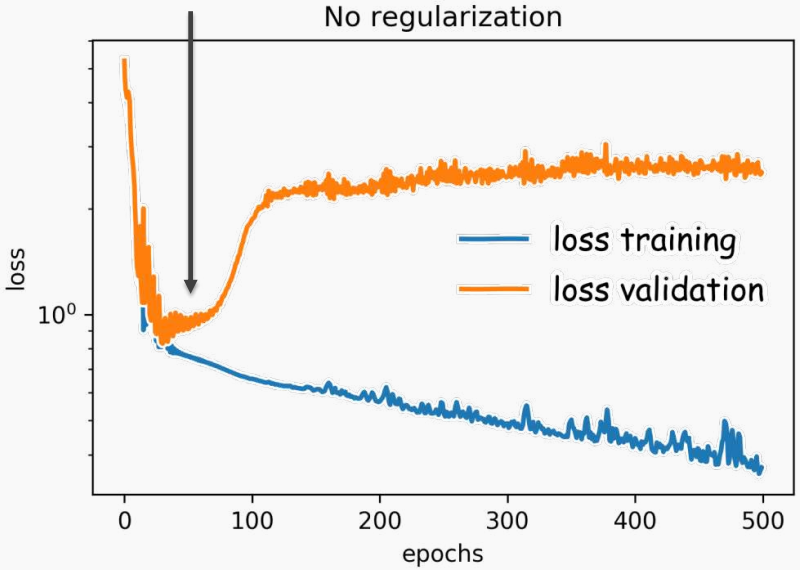
Outline

Regularization of NN

- Norm Penalties
- **Early Stopping**
- Data Augmentation
- Dropout

Early Stopping

Early stopping: terminate while validation set performance is better. Sometimes is worth waiting a little before stopping. This is called `patience`.



Patience is defined as the number of epochs to wait before early stop if no progress on the validation set.

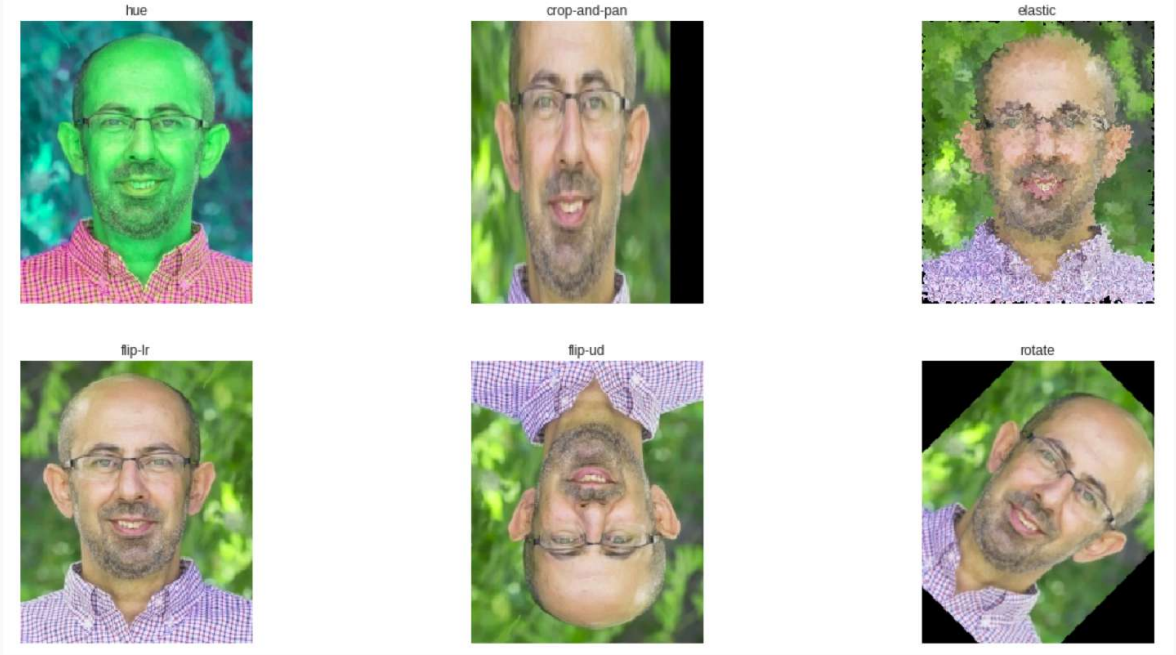
The patience is often set somewhere between 10 and 100 (10 or 20 is more common), but it really depends on the dataset and network.

Outline

Regularization of NN

- Norm Penalties
- Early Stopping
- **Data Augmentation**
- Dropout

Data Augmentation



Data Augmentation: dos and don'ts

We use ImageDataGenerator to augment the dataset

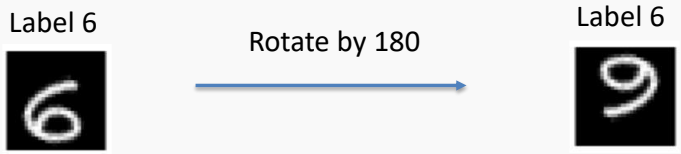
```
def get_generator():
    # create duplicate images
    BATCHES_PER_EPOCH = 300//BATCH_SIZE
    classes = ['pavlos', 'not-pavlos']
    for img_class in classes:
        img = Image.open((f'{DATA_DIR}/{img_class}.jpeg'))
        for i in range(1, BATCH_SIZE*BATCHES_PER_EPOCH//2+1):
            img.thumbnail(TARGET_SIZE, Image.ANTIALIAS)
            img.save(f'{DATA_DIR}/{img_class}/{img_class}{i:0>3}.jpeg', "JPEG")

    data_gen = ImageDataGenerator(
            rescale=1./255,
            height_shift_range=0.5,
            width_shift_range=0.5)

    img_generator = data_gen.flow_from_directory(
            DATA_DIR,
            target_size=(TARGET_SIZE),
            batch_size=BATCH_SIZE,
            classes=classes,
            class_mode='binary')
    return img_generator
```

Data Augmentation: dos and don'ts

Carefully choose your transformations. Not all transformations are valid.



Data Augmentation does not work for tabular data and not as nicely for time series.

Outline

Regularization of NN

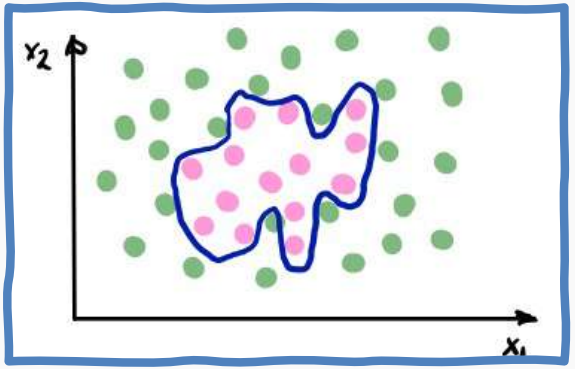
- Norm Penalties
- Early Stopping
- Data Augmentation
- **Dropout**

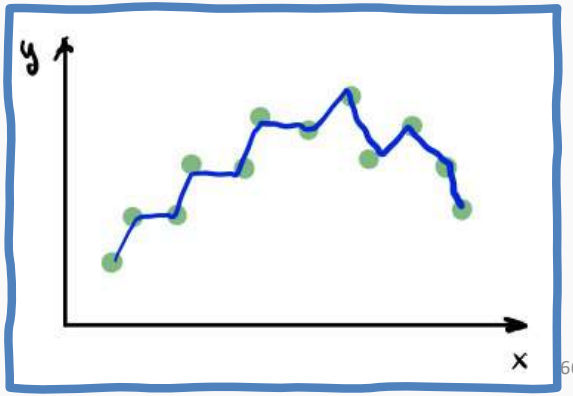
Co-adaptation

Overfitting occurs when the model is sensitive to slight variations on the input and therefore it fits the noise.

L1 and L2 regularizations ‘shrink’ the weights to avoid this problem.

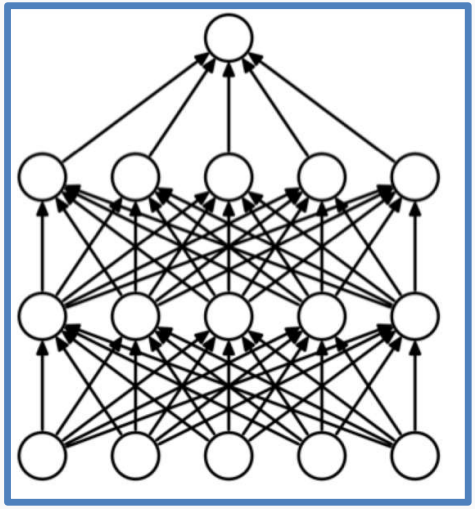
However in a large network many units can collaborate to respond to the input while the weights can remain relatively small. This is called co-adaptation.



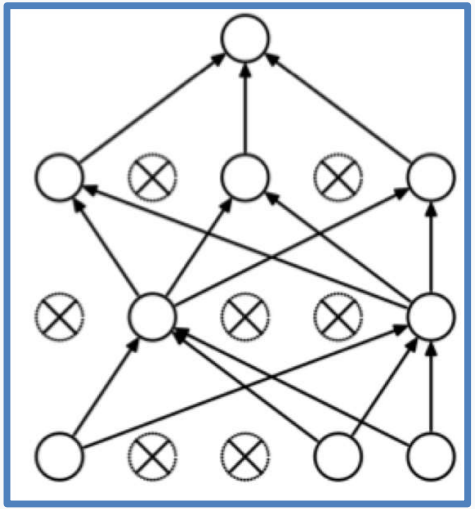


Dropout

- Randomly set some neurons and their connections to zero (i.e. “dropped”)
- Prevent overfitting by reducing co-adaptation of neurons
- Like training many random sub-networks



Standard Neural Network



After applying dropout

https://arxiv.org/pdf/1207.0580.pdf

Dropout: Training

For each new example in a mini-batch (could be for one mini-batch depending on the implementation):

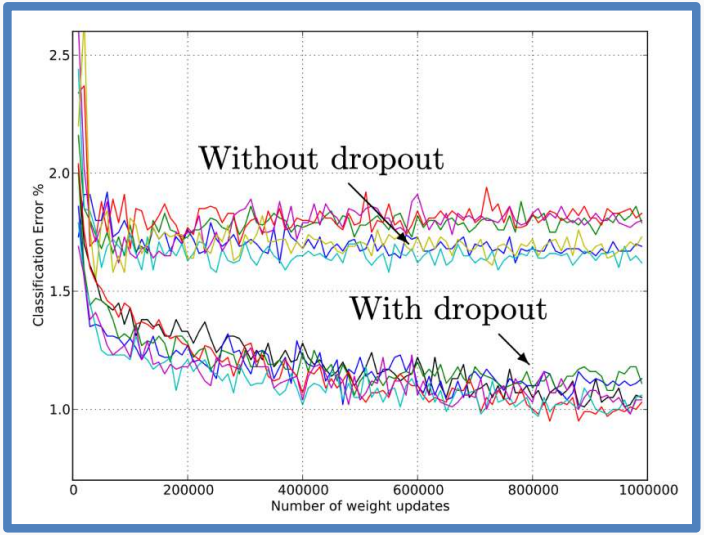
- Randomly sample a binary mask μ independently, where μ_i indicates if input/hidden node i is included
- Multiply output of node i with μ_i, and perform gradient update

Typically:

- Input nodes are included with prob=0.8 (as per original paper, but rarely used)
- Hidden nodes are included with prob=0.5

Dropout

- Widely used and highly effective



Test error for different architectures with and without dropout.

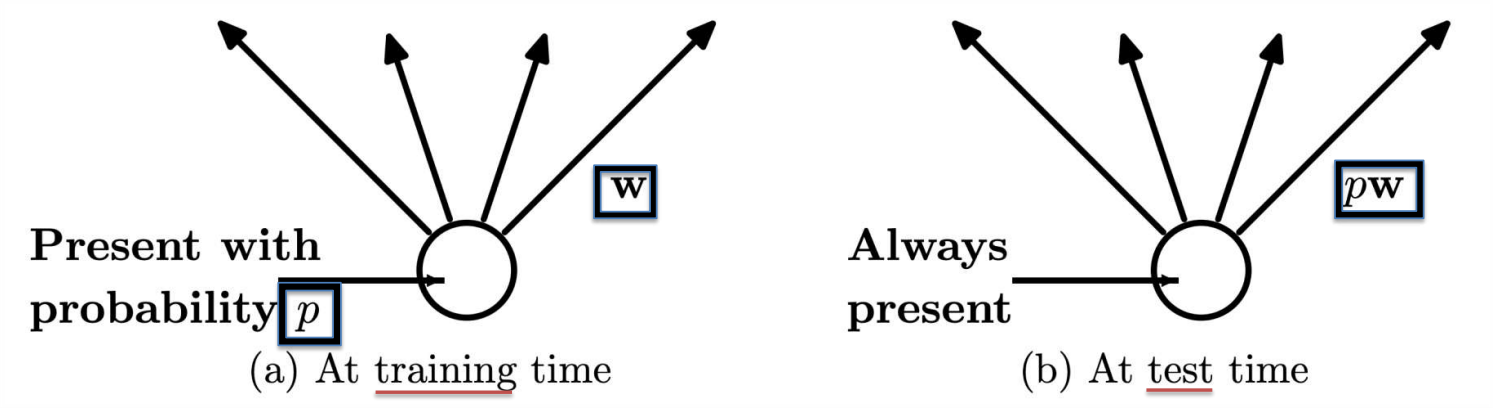
The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

- Proposed as an alternative to ensemble methods, which is too expensive for neural nets

http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

Dropout: Prediction

- We can think of dropout as training many of sub-networks
- At **test time**, we can "aggregate" over these sub-networks by **reducing connection weights in proportion to dropout probability, *p***



NOTE: Dropouts can be used for **neural network inference** by dropping during predictions and predicting multiple times to get a distribution

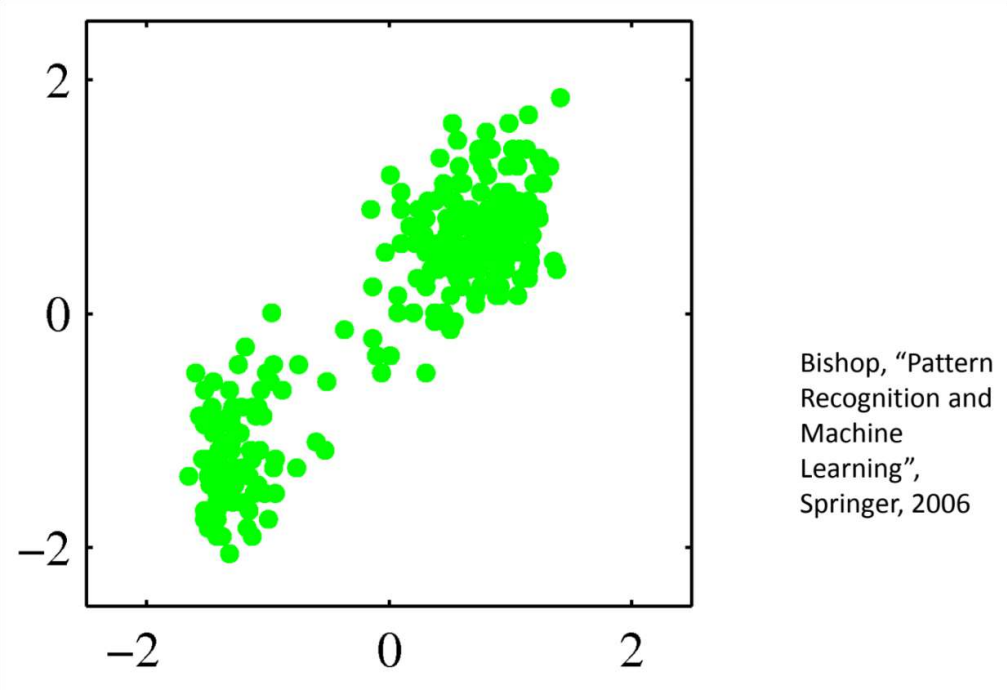
Unsupervised Learning

Unsupervised Learning

- K-means
- Mean-shift
- Hierarchical Clustering
- DBSCAN

- Applications

Unsupervised Setting



Bishop, “Pattern Recognition and Machine Learning”, Springer, 2006

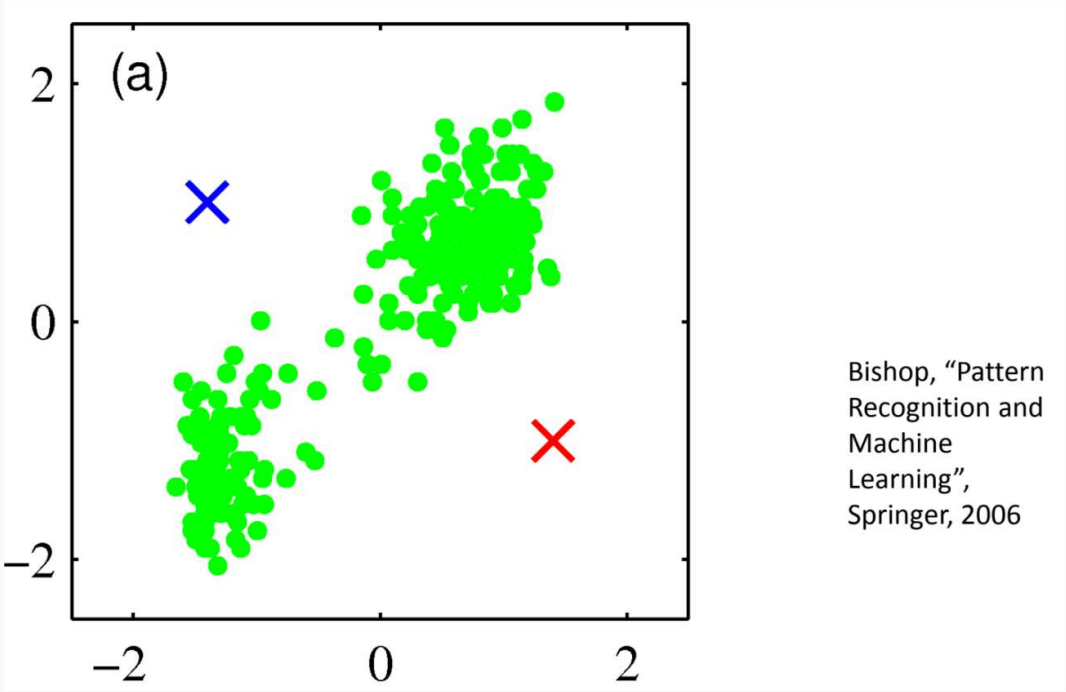
K-means – Algorithm

Initialization:

– choose K random positions

– assign cluster centers μ^j to these positions

K-means



Bishop, “Pattern Recognition and Machine Learning”, Springer, 2006

K-means

Until Convergence:

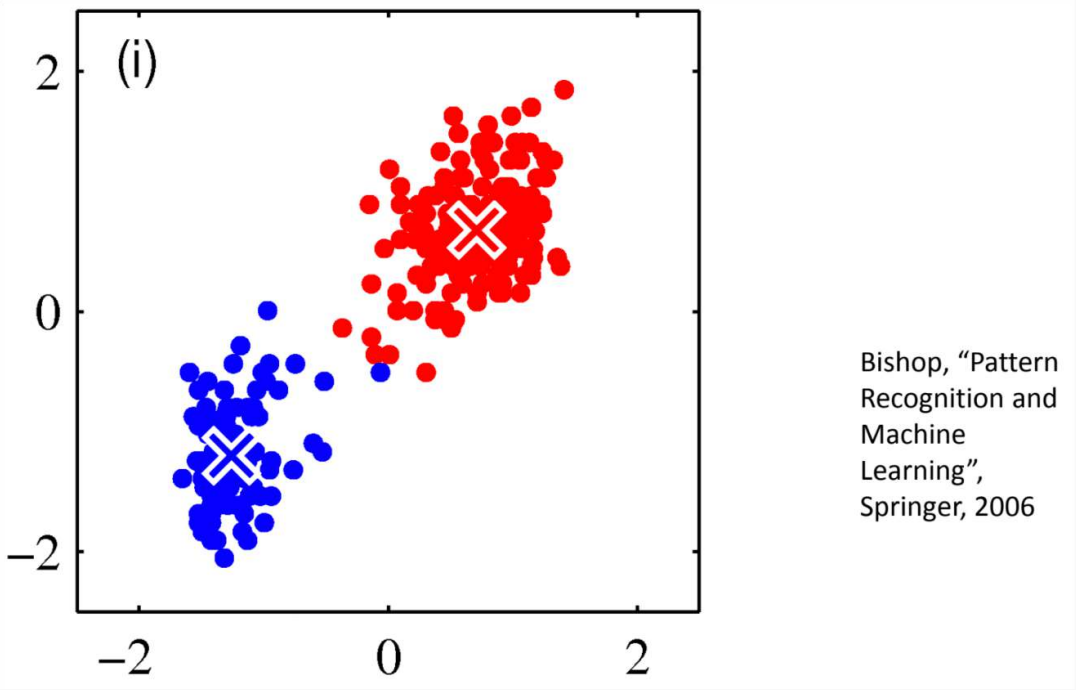
– Compute distances $\left\| x^{(i)} - \mu^{(i)} \right\|$

– Assign points to nearest cluster center

– Update Cluster centers:

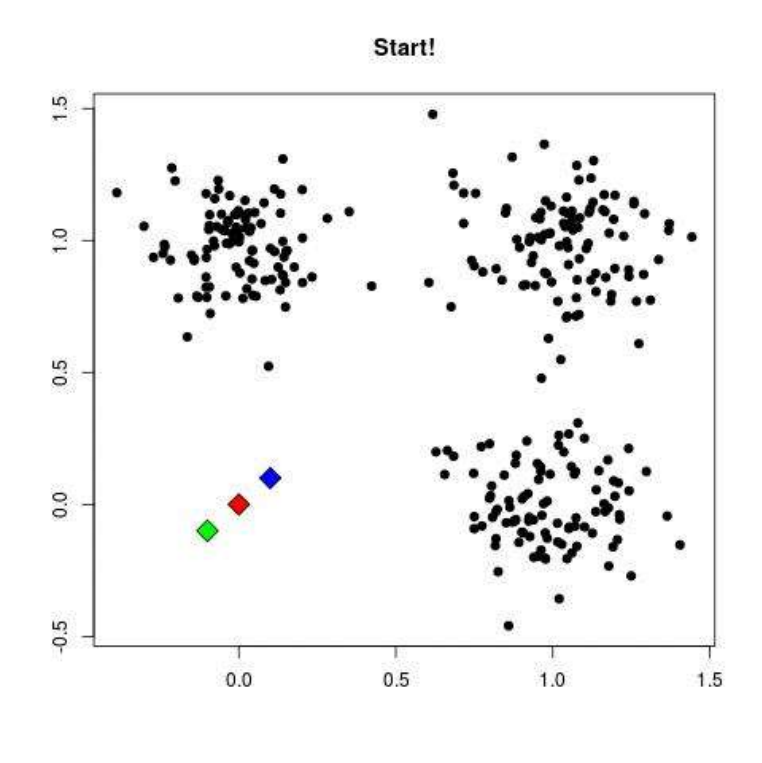
$$\mu^{(j)} = \frac{1}{N_j} \sum_{x_i \in C_j} x_i$$

K-means

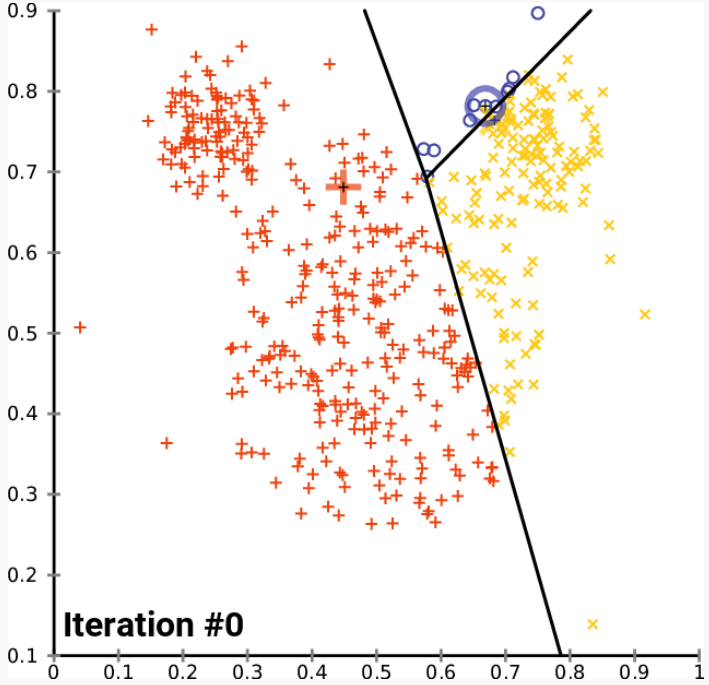


Bishop, “Pattern Recognition and Machine Learning”, Springer, 2006

K-means



K-means



K-means Example

R

G

B

K-means Example

K-means Example

K-means Summary

- Guaranteed to converge
- Result depends on initialization
- Number of clusters is important
- Sensitive to outliers
  - Use median instead of mean for updates

Initialization Methods

- Random Positions
- Random data points as Centers
- Random Cluster assignment to data points

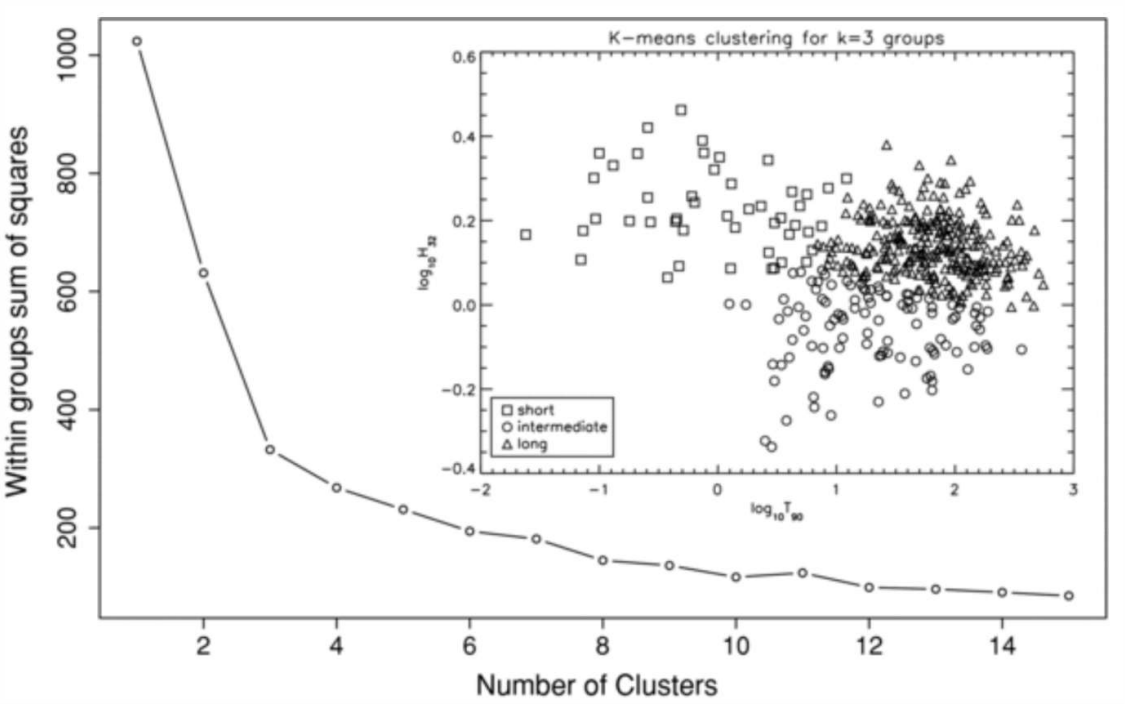
- Start several times

How to find K

- Extreme cases:
  - K=1
  - K=N

- Choose K such that increasing it does not model the data much better.

“Knee” or “Elbow” method



Cross Validation

- Use this if you want to apply your clustering solution to new unseen data

- Partition data into n folds
- Cluster on n-1 folds
- Compute sum of squared distances to centroids for validation set

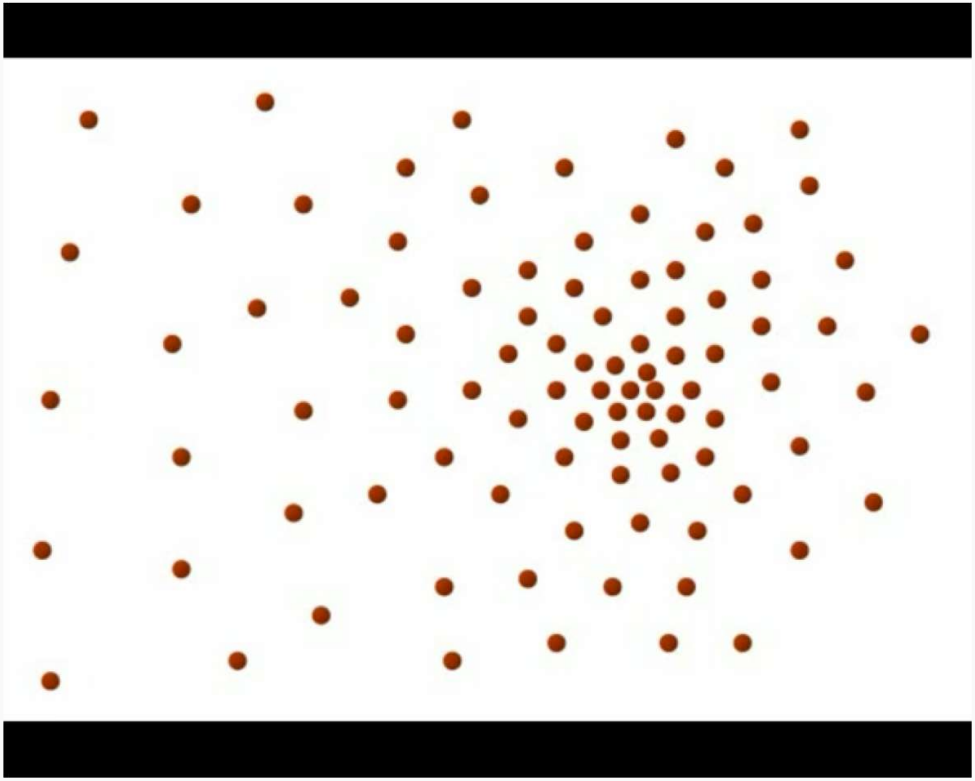
Getting Rid of K

- Having to specify K is annoying
- Can we do without?

Mean Shift

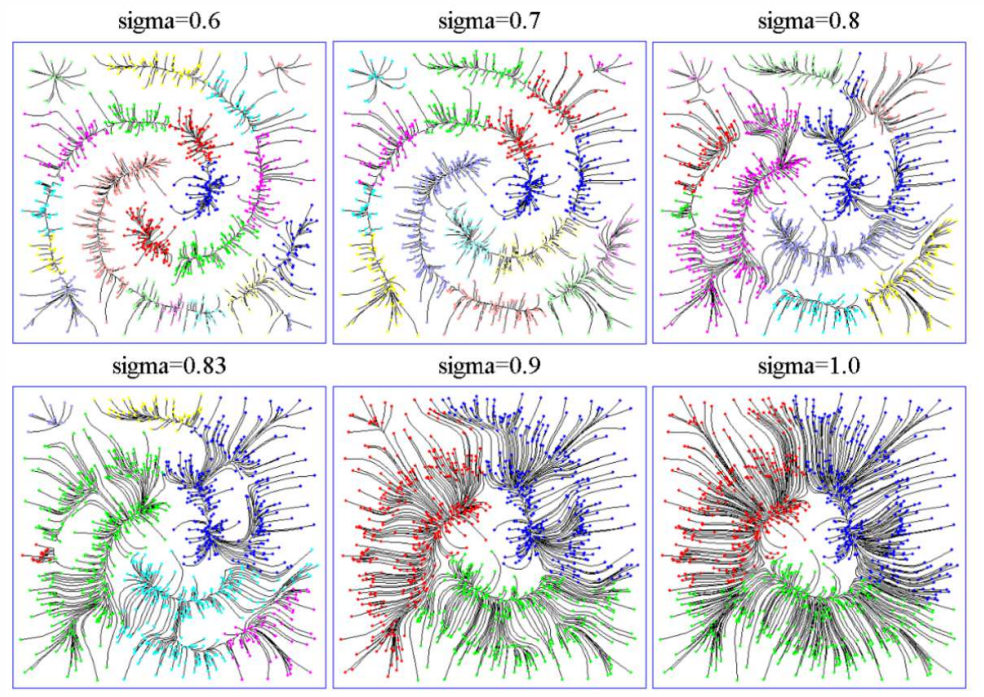
1. Put a window around each point
2. Compute mean of points in the frame.
3. Shift the window to the mean
4. Repeat until convergence

Mean Shift



https://www.youtube.com/watch?v=kmaQAsotT9s

Mean Shift



Fischer et al., “Clustering with the Connectivity Kernel“, NIPS (2003)

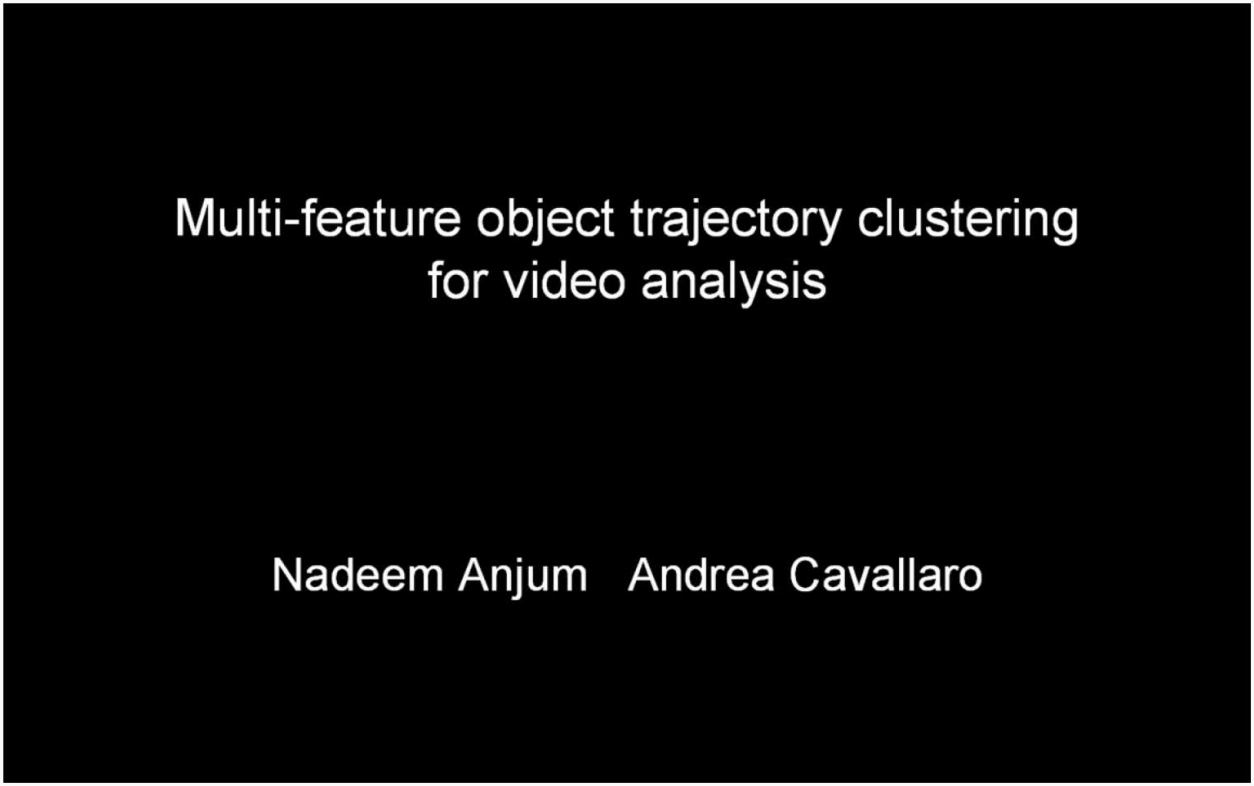
Mean Shift Summary

- Does not need to know number of clusters
- Can handle arbitrary shaped clusters
- Robust to initialization
- Needs bandwidth parameter (window size)
- Computationally expensive

- Very good article:

http://saravananthirumuruganathan.wordpress.com/2010/04/01/introduction-to-mean-shift-algorithm/

Multi-feature trajectory clustering using mean shift

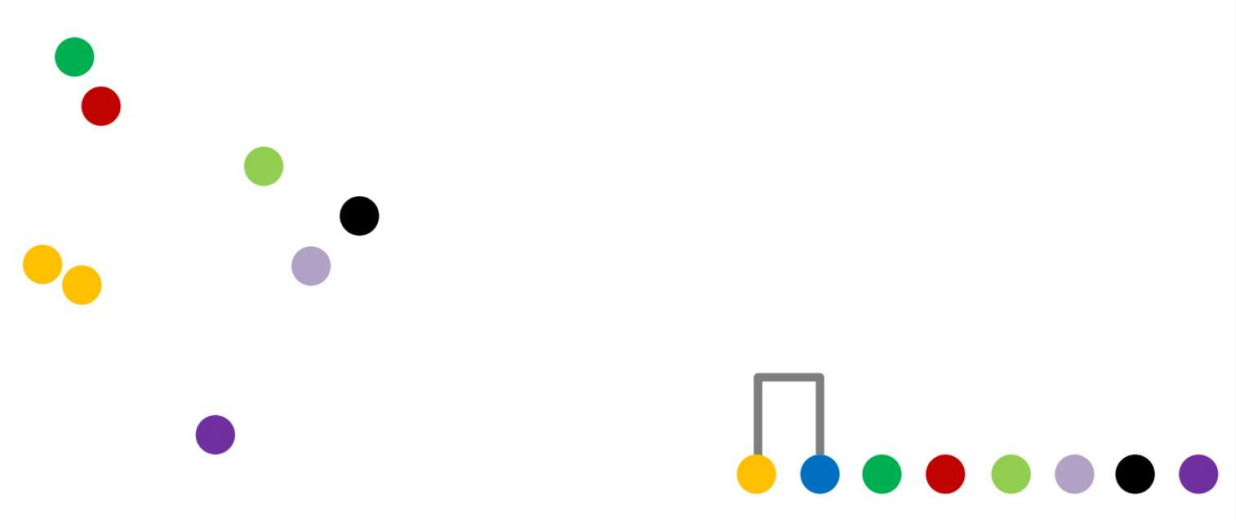


Parameters parameters

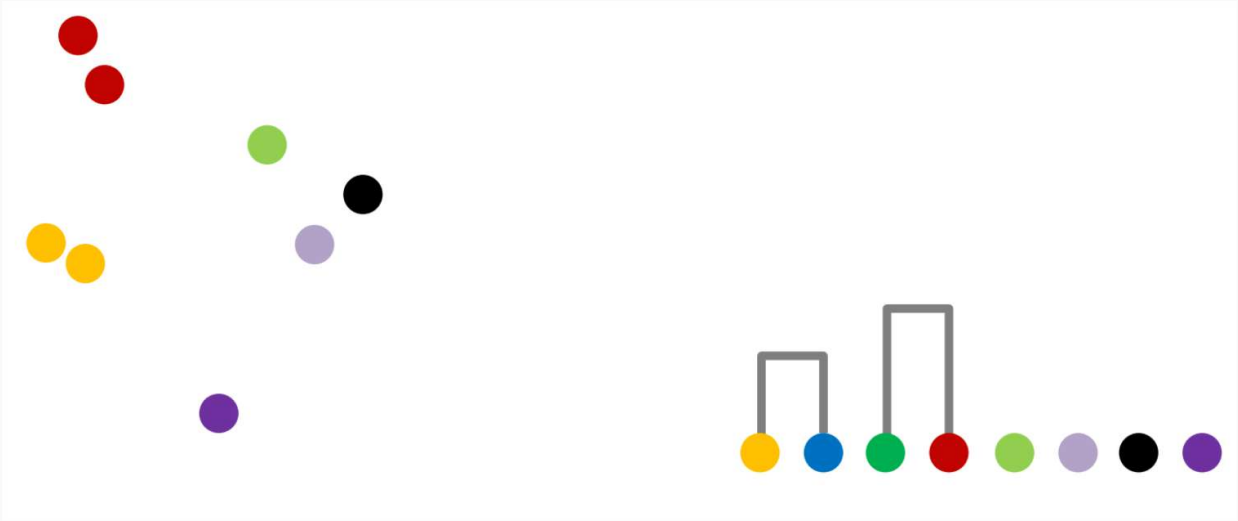
- For K means we need K and result depends on initialization
- For mean shift we need the window size and a lot of computation
- Hierarchical Clustering keeps a history of all possible cluster assignments

Hierarchical Clustering

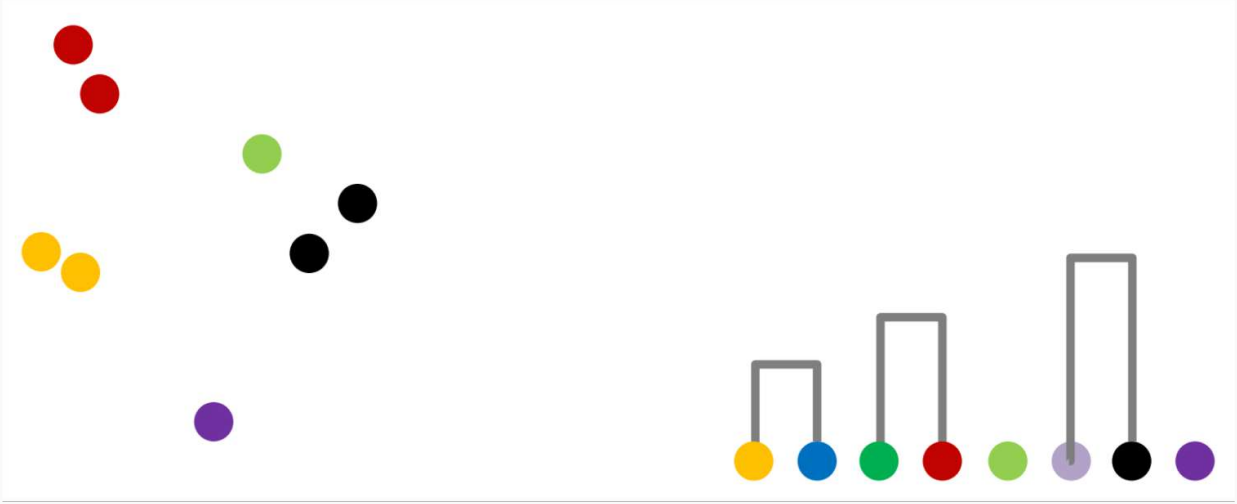
Hierarchical Clustering



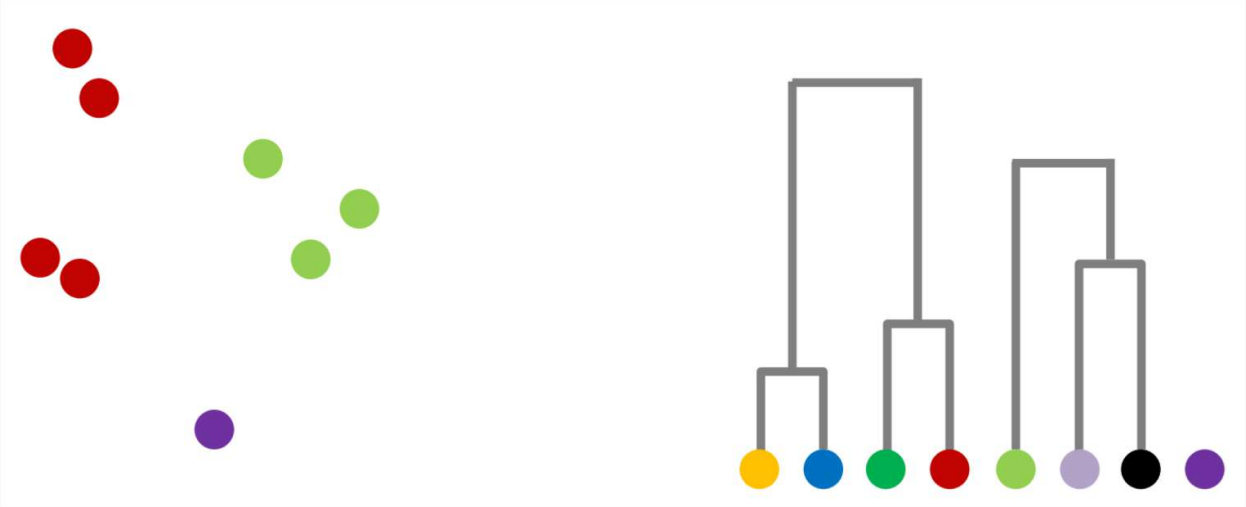
Hierarchical Clustering



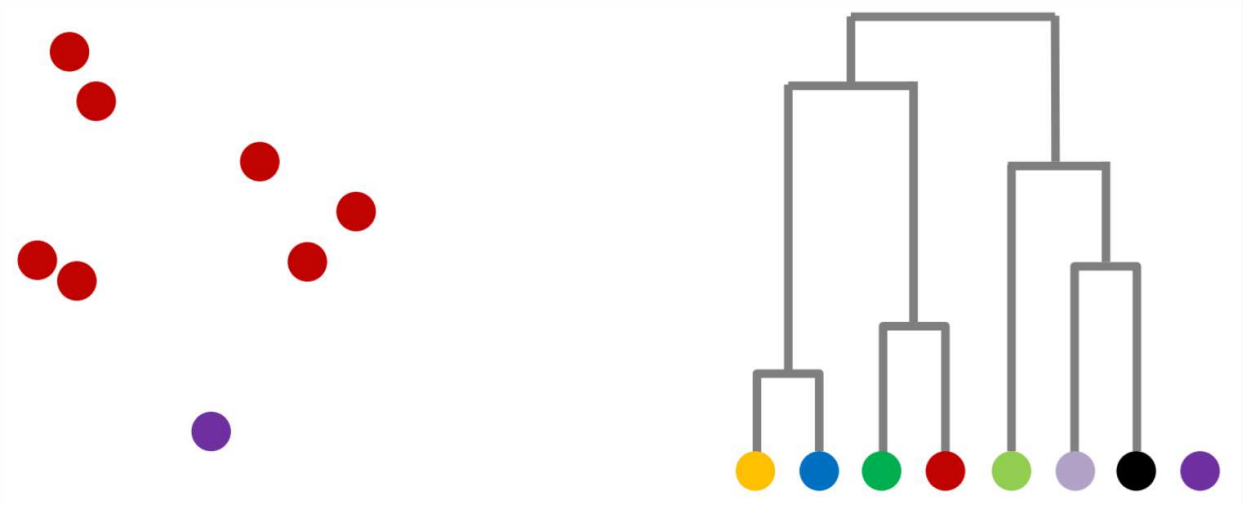
Hierarchical Clustering



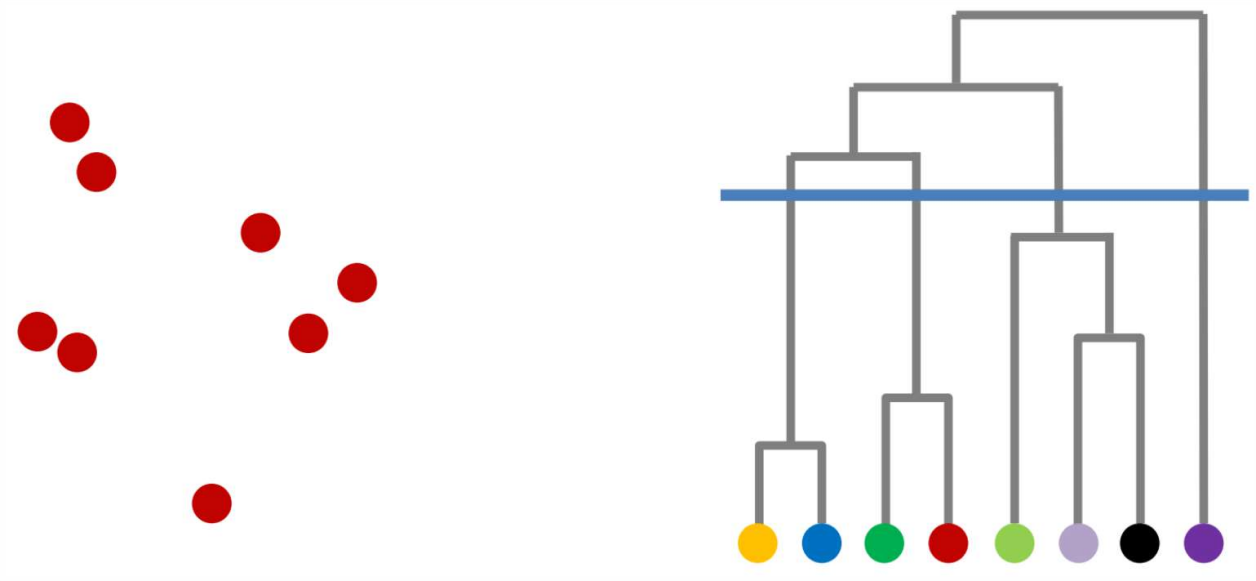
Hierarchical Clustering



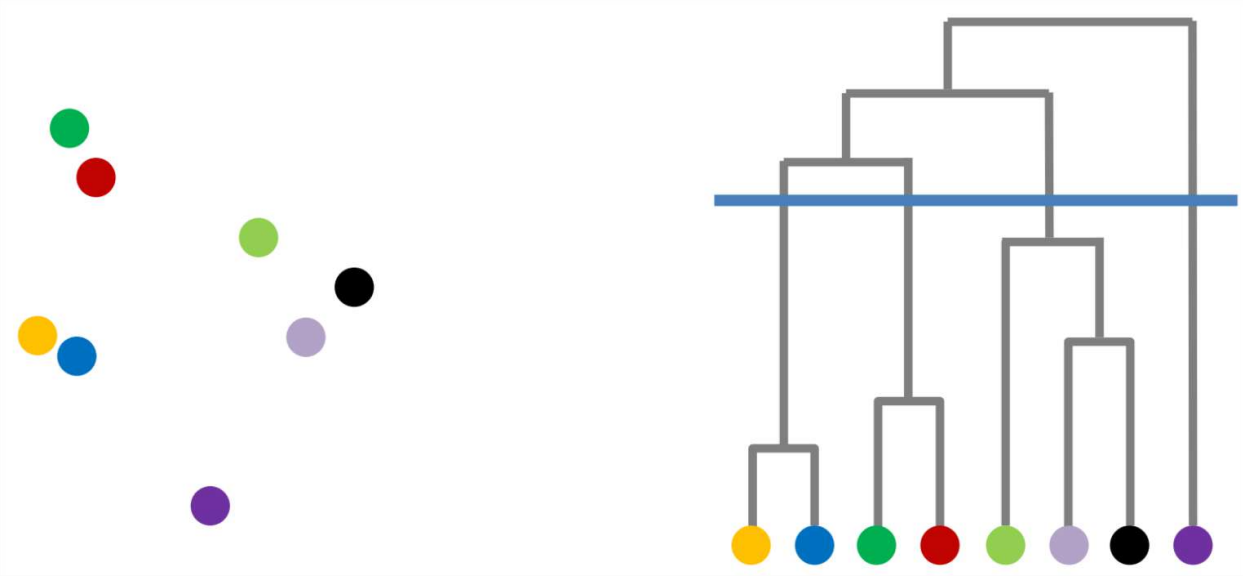
Hierarchical Clustering



Hierarchical Clustering



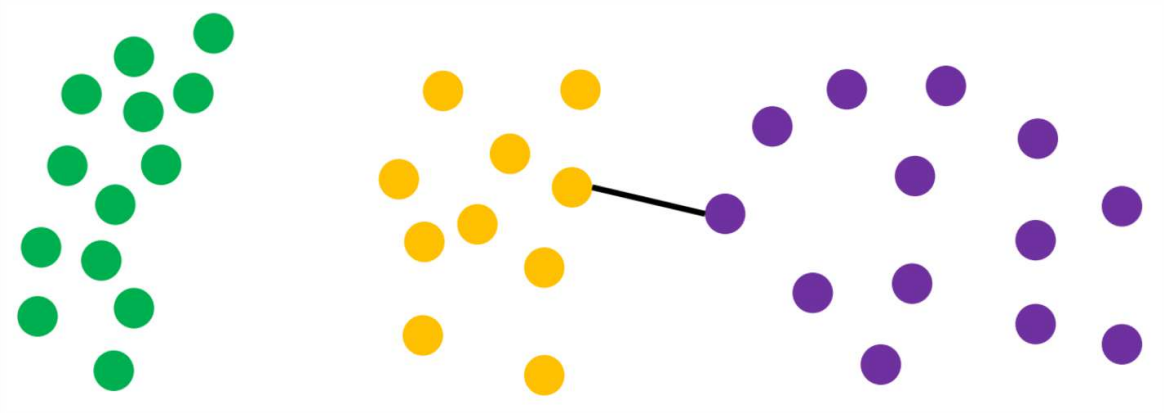
Hierarchical Clustering



Hierarchical Clustering

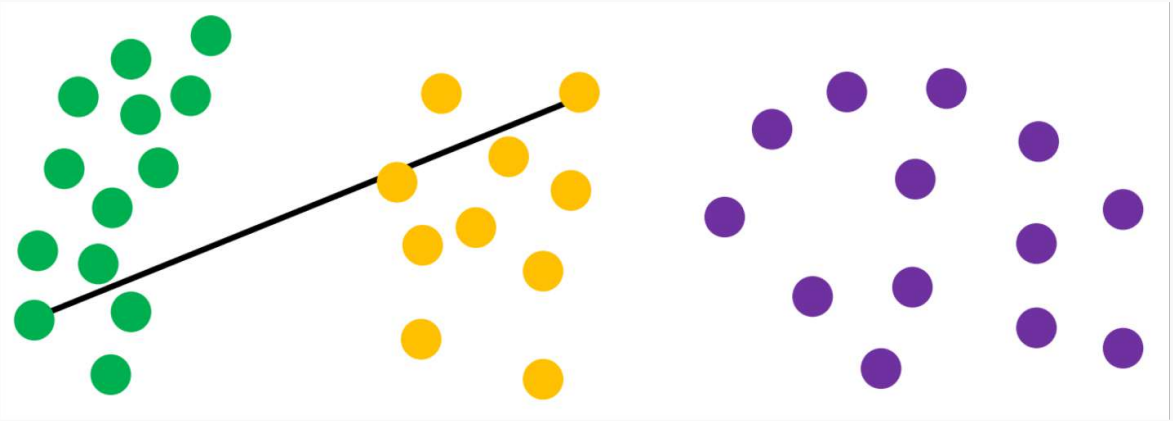
- Produces complete structure
- No predefined number of clusters
- Similarity between clusters:
  - single-linkage: $\min\{d(x,y) : x \in \mathcal{A}, y \in \mathcal{B}\}$
  - complete-linkage: $\max\{d(x,y) : x \in \mathcal{A}, y \in \mathcal{B}\}$
  - average linkage: $\frac{1}{|\mathcal{A}| \cdot |\mathcal{B}|} \sum_{x \in \mathcal{A}} \sum_{y \in \mathcal{B}} d(x,y)$

Single Linkage



$$\min\{d(x,y) : x \in \mathcal{A}, y \in \mathcal{B}\}$$

Complete Linkage

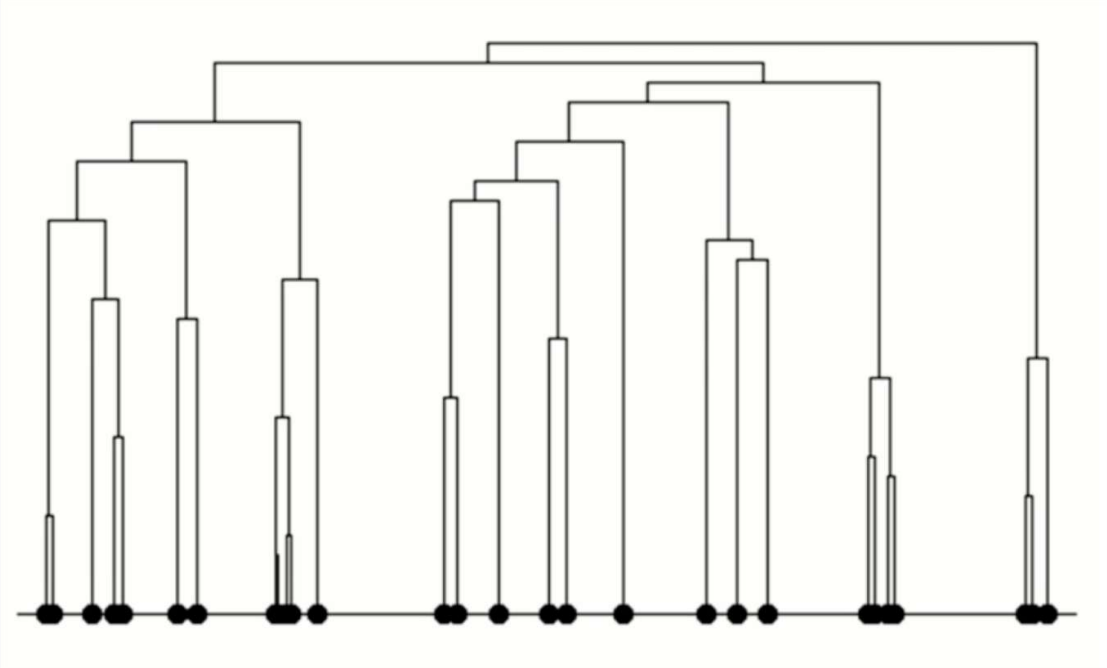


$$\max\{d(x,y) : x \in \mathcal{A}, y \in \mathcal{B}\}$$

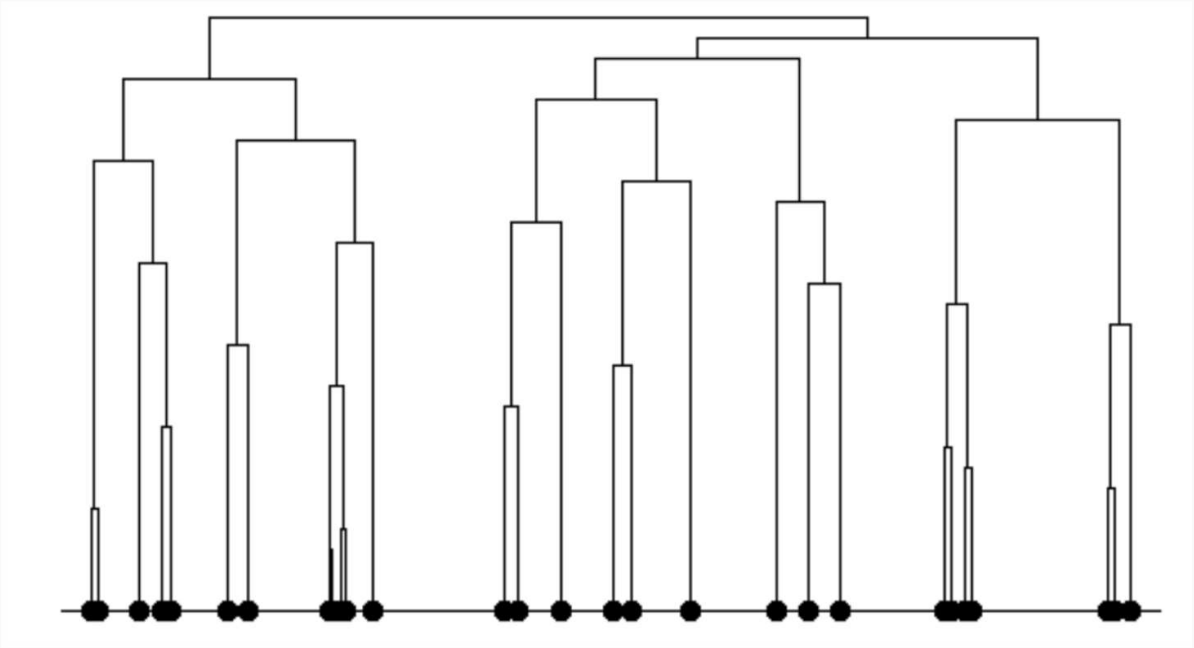
Linkage Matters

- Single linkage: tendency to form long chains
- Complete linkage: Sensitive to outliers
- Average-link: Trying to compromise between the two

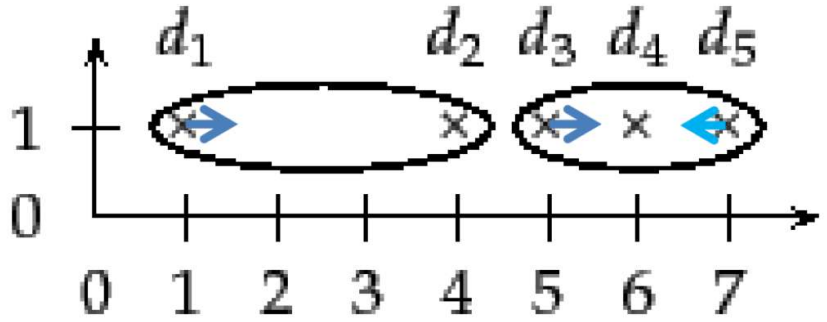
Chaining Phenomenon: Single linkage



Chaining Phenomenon: Complete linkage



Outlier Sensitivity

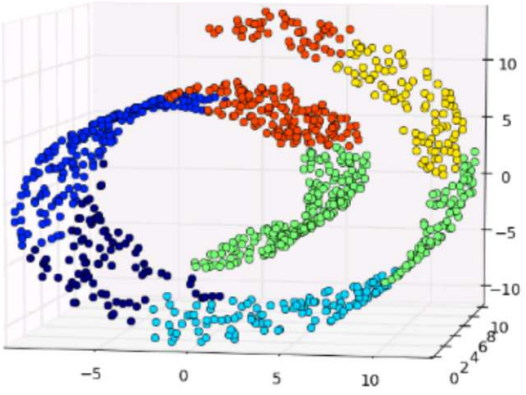


+ 2*epsilon

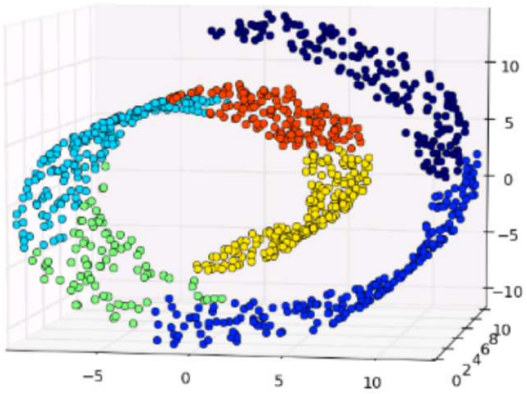
- 1*epsilon

Outliers in complete-link clustering. The five documents have the x-coordinates $1+2\epsilon, 4, 5+2\epsilon, 6$ and $7-\epsilon$. Complete-link clustering creates the two clusters shown as ellipses. The most intuitive two-cluster clustering is $\{\{d_1\}, \{d_2, d_3, d_4, d_5\}\}$, but in complete-link clustering, the outlier d_1 splits $\{d_2, d_3, d_4, d_5\}$ as shown.

Swiss Role Problem



without connectivity constraints

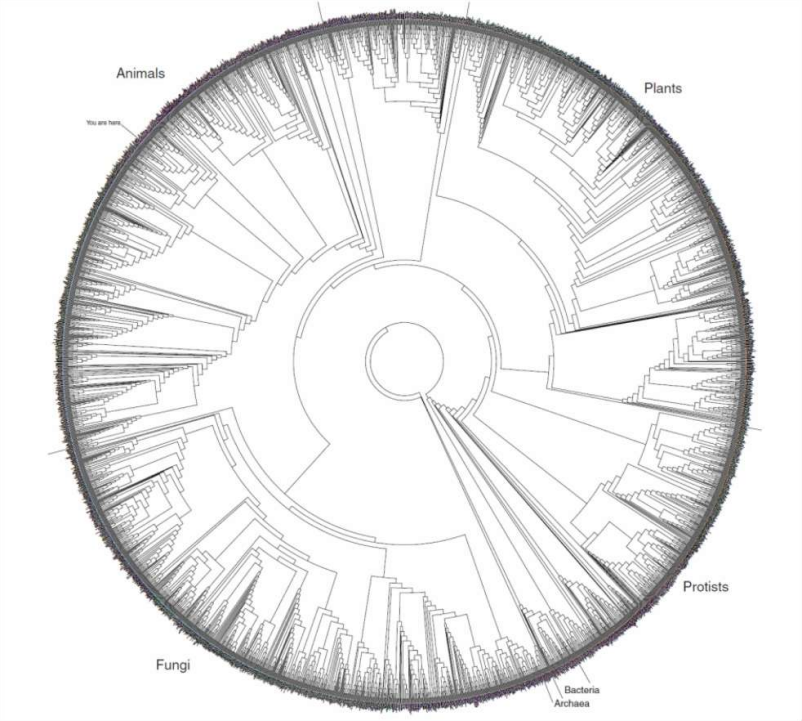


with connectivity constraints

only adjacent clusters can be merged together

http://scikit-learn.sourceforge.net/modules/clustering.html

Tree of Life



http://www.zo.utexas.edu/faculty/antisense/DownloadfilesToL.html

Efficient Hierarchical Graph-Based Video Segmentation

Matthias Grundmann[1,2], Vivek Kwatra[2],
Mei Han[2] and Irfan Essa[1]

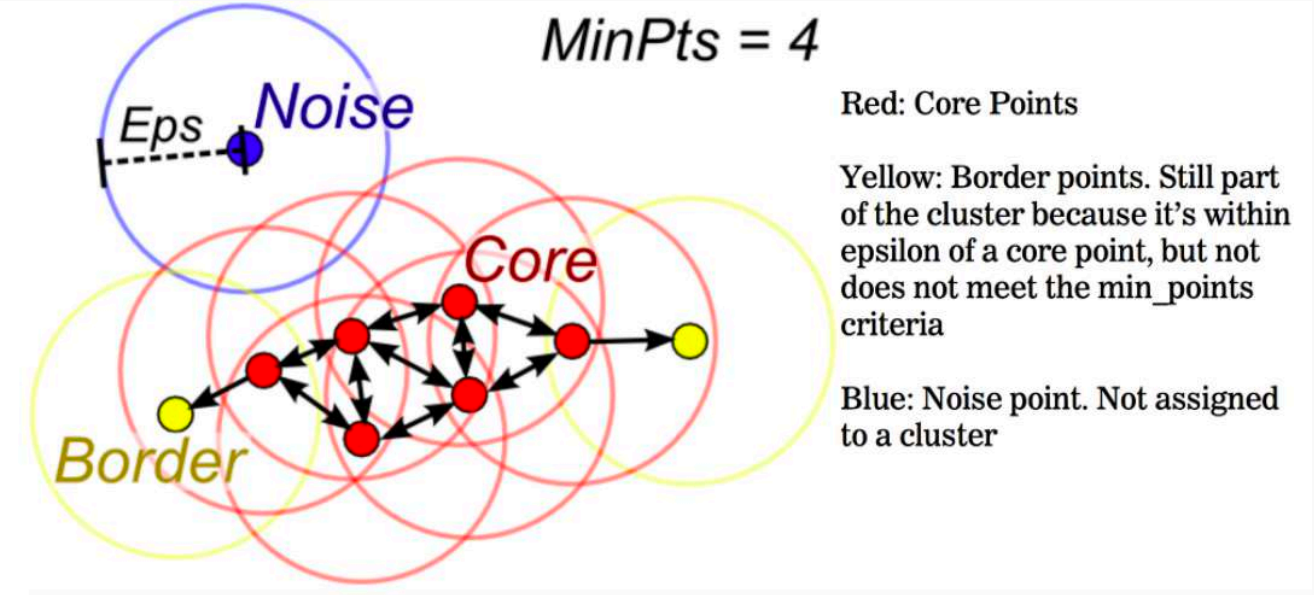
[1]Georgia Tech [2]Google Research

IEEE CVPR, San Francisco, USA, June 2010

DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN)

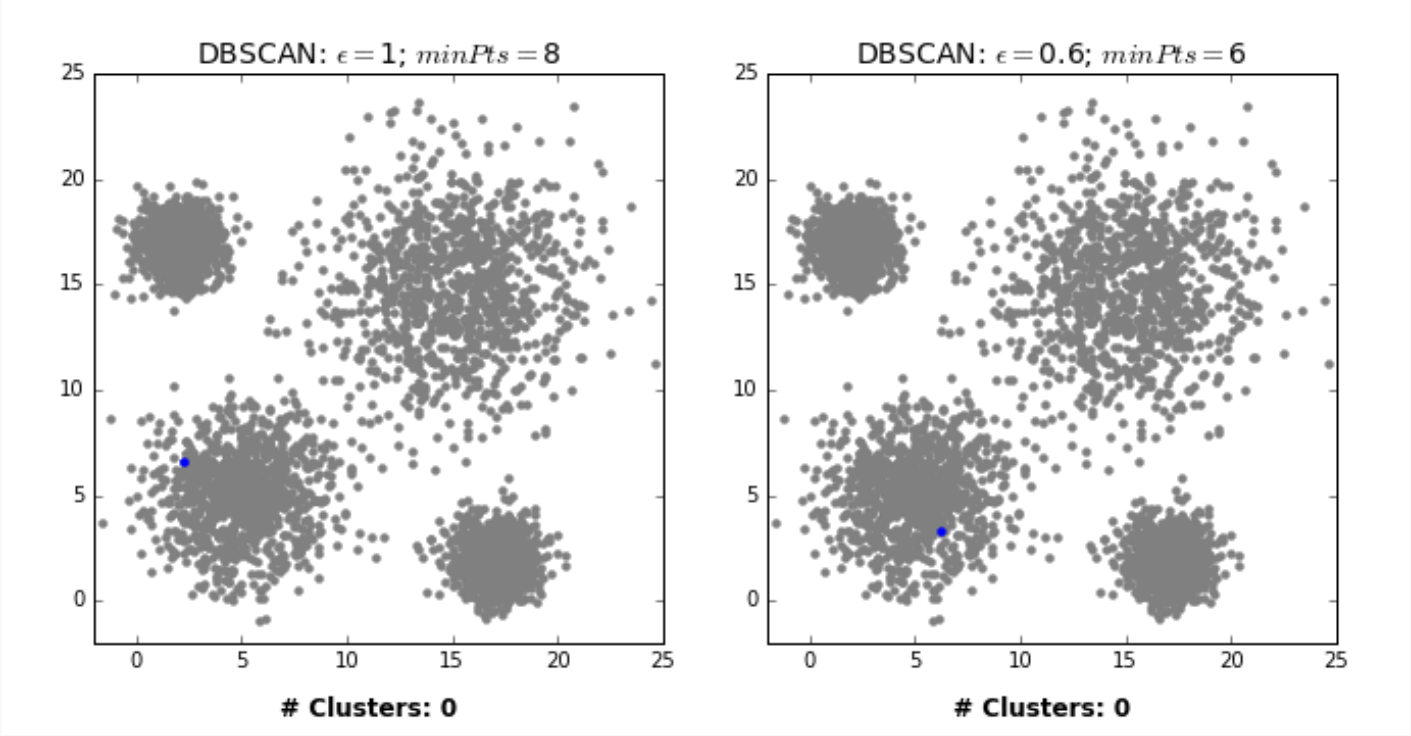
- groups together points that are closely packed together
- marking as outliers points that lie alone in low-density regions



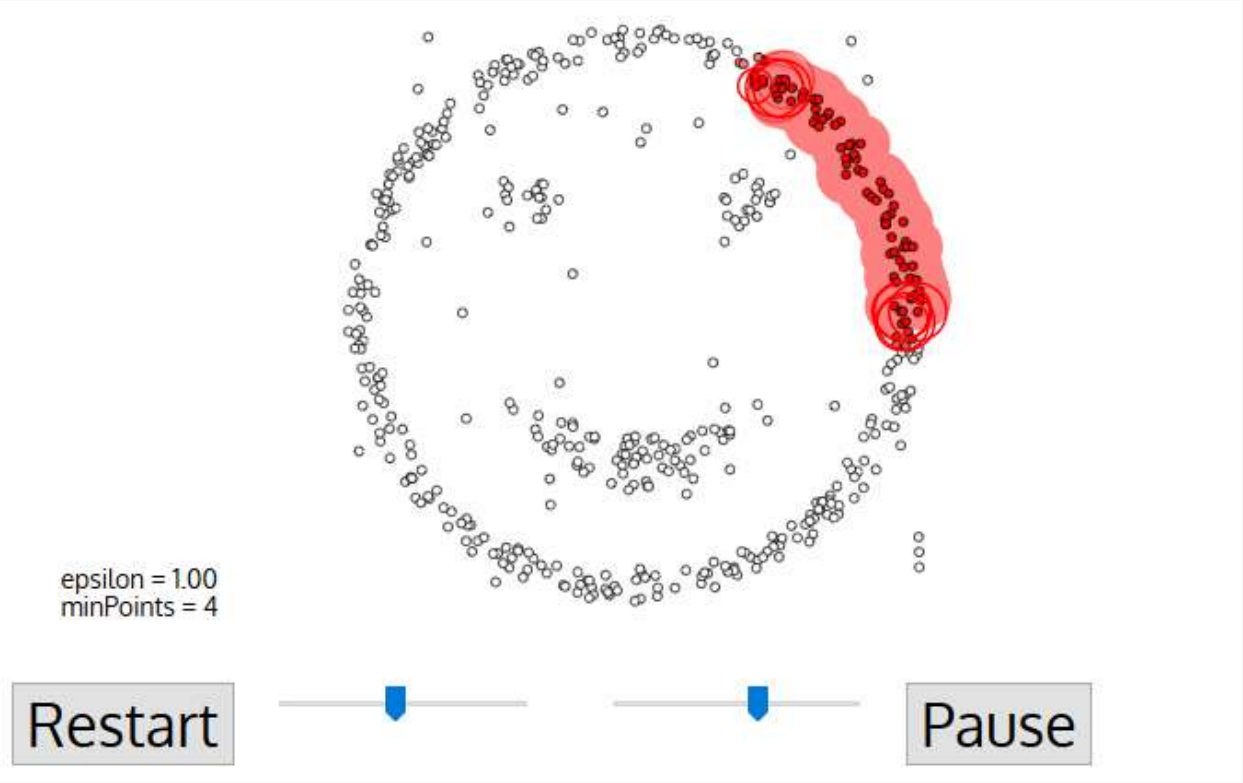
DBSCAN

- A point *p* is a *core point* if at least minPts points are within distance *ε* of it (including *p*).
- A point *q* is *directly reachable* from *p* if point *q* is within distance *ε* from core point *p*.
- A point *q* is *reachable* from *p* if there is a path $p_1,\ldots, p_n$ with $p_1 = p$ and $p_n = q$, where each p_{i+1} is directly reachable from p_i.
- All points not reachable from any other point are *outliers* or *noise points*.

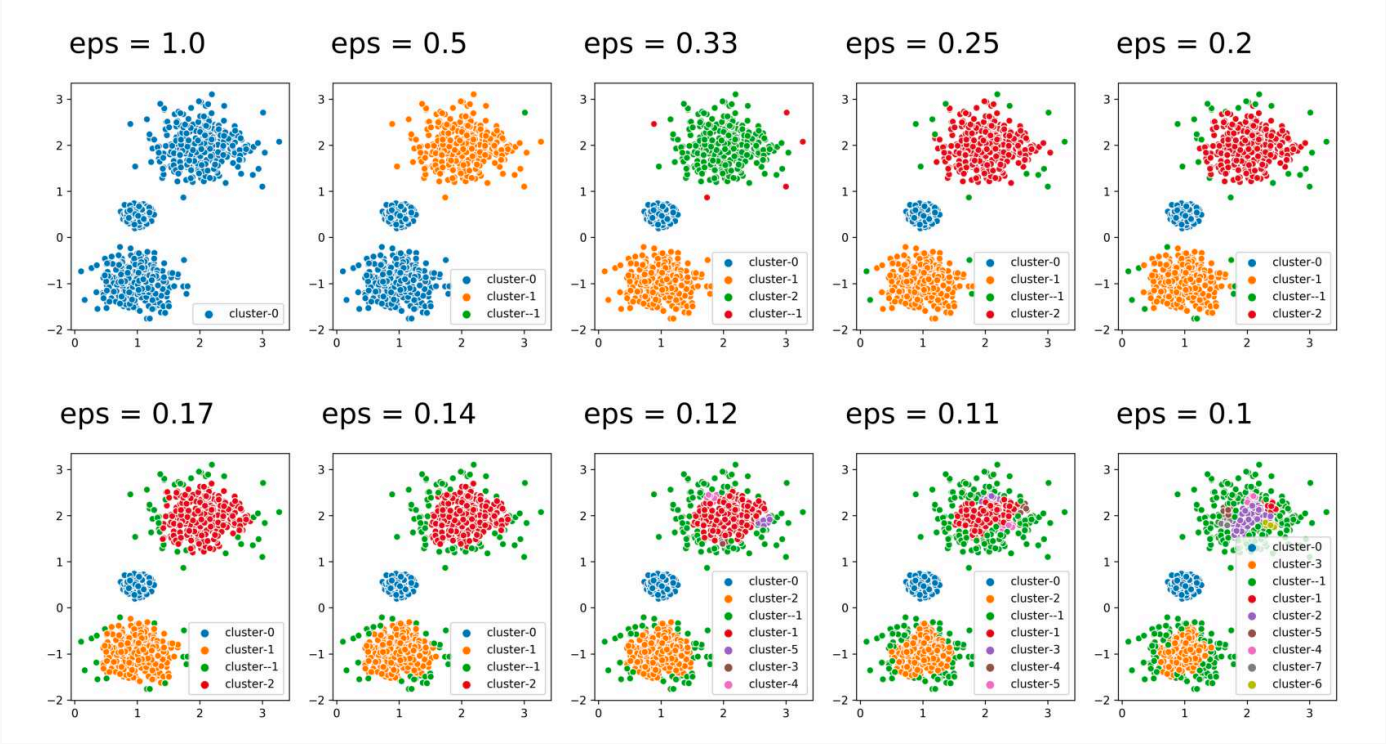
DBSCAN


DBSCAN: $\epsilon = 1$; $minPts = 8$
Clusters: 0
DBSCAN: $\epsilon = 0.6$; $minPts = 6$
Clusters: 0

DBSCAN



DBSCAN


eps = 1.0
cluster-0
eps = 0.5
cluster-0
cluster-1
cluster--1
eps = 0.33
cluster-0
cluster-1
cluster-2
cluster--1
eps = 0.25
cluster-0
cluster-1
cluster--1
cluster-2
eps = 0.2
cluster-0
cluster-1
cluster--1
cluster-2
eps = 0.17
cluster-0
cluster-1
cluster--1
cluster-2
eps = 0.14
cluster-0
cluster-1
cluster--1
cluster-2
eps = 0.12
cluster-0
cluster-2
cluster--1
cluster-1
cluster-5
cluster-3
cluster-4
eps = 0.11
cluster-0
cluster-2
cluster--1
cluster-1
cluster-3
cluster-4
cluster-5
eps = 0.1
cluster-0
cluster-3
cluster--1
cluster-1
cluster-2
cluster-5
cluster-4
cluster-7
cluster-6

K-means vs DBSCAN