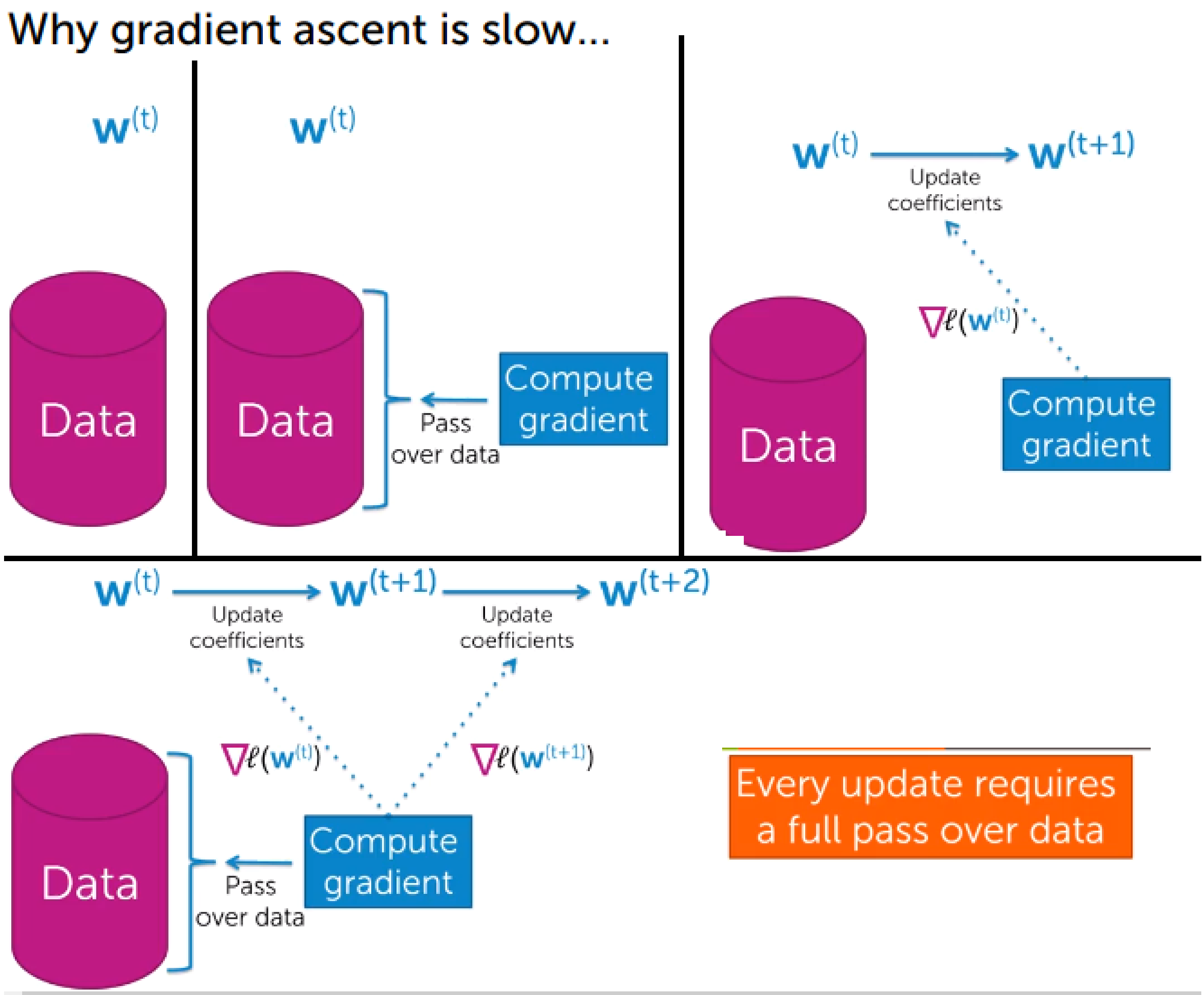
Scaling ML to huge datasets

Gradient Ascent will not scale to todays' huge datasets

Gradient Ascent

- Consider we have a large dataset (DATA) and a set of coefficients (**w(t)**) to be updated.
- Using gradient ascent compute the gradient on the dataset. This requires to make a pass or a scan over the data, computing the contribution of each of these data points to the gradient.
- Then compute the gradient and update the coefficients / parameters and get (**w(t+1)**).
- Then return to the dataset and make another pass over each data-points and compute a new gradient and update the parameters and coefficients ((**w(t+2)**)).

> ***There every time a coefficients needs to be updated, then we need to perform a full scan or a full pass over the entire dataset. This process can be 'very slow' if 'dataset is very big'.***



Datasets are getting huge, ad we need them!

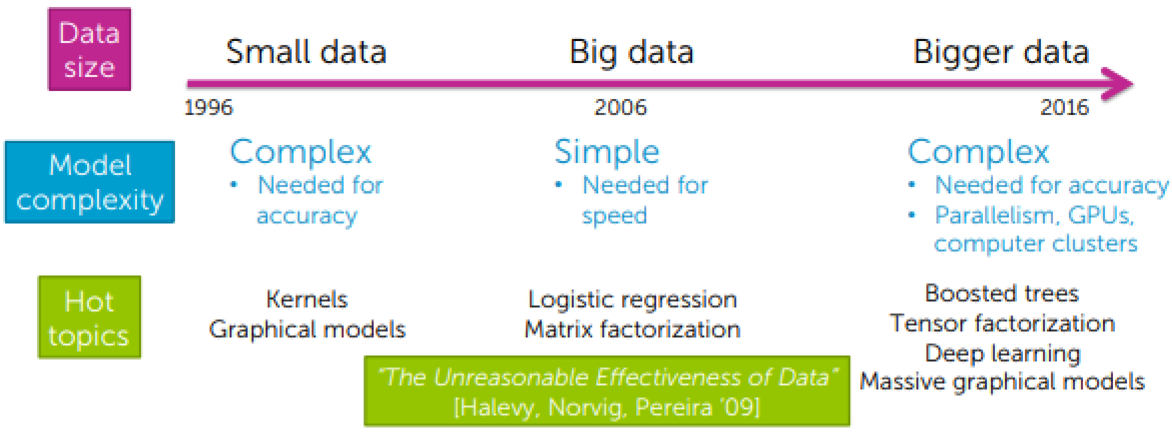
- Internet -> WWW - 4.8B webpages
- twitter -> 500M Tweets/day
- Internet of Things -> Sensors everywhere
- Youtube -> 300 hours uploaded/min

- 1B users
- Ad revenue
- 5B views/day
- **(Need ML algorithm to learn from billions of video views every day, & to recommend ads within milliseconds)**.

Timeline of scalabale machine learning and 'Stochastic Gradient'

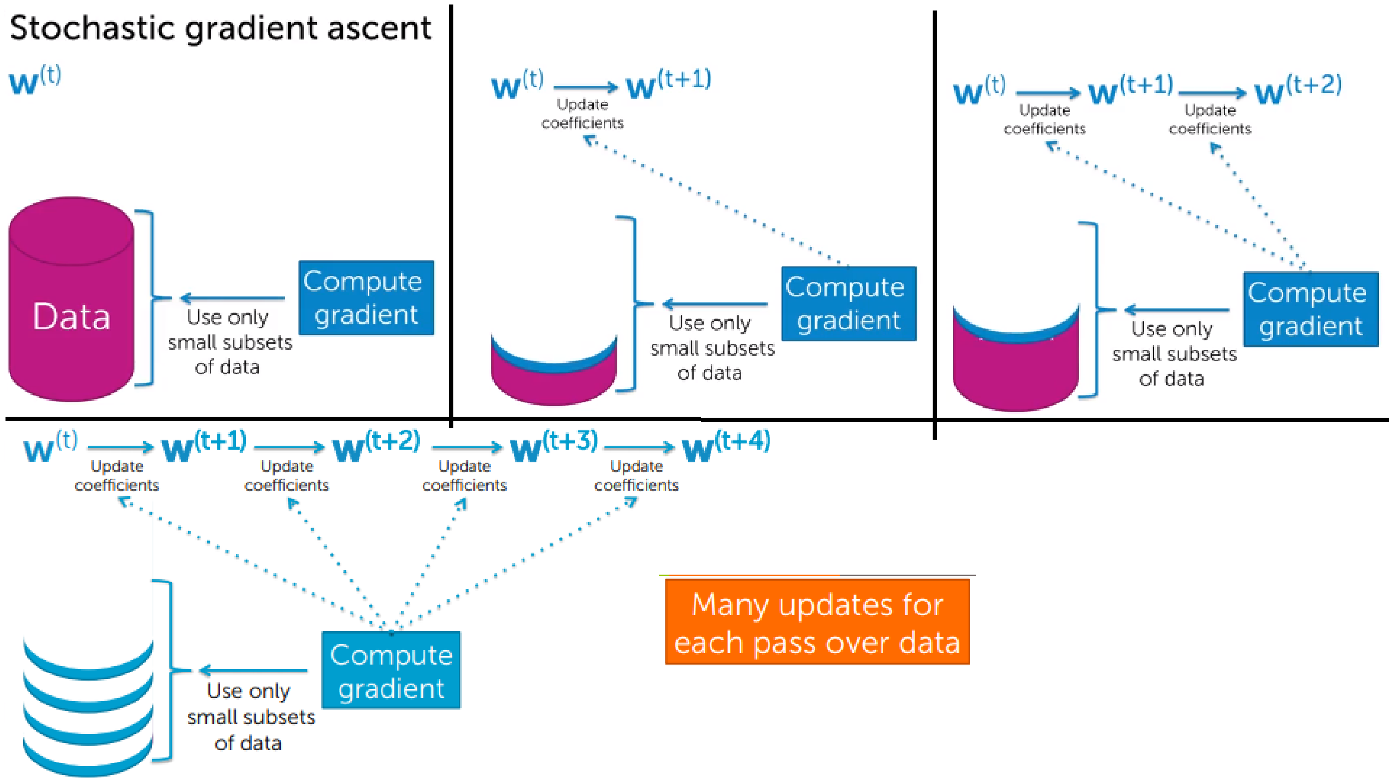
Machine Learning inproves significantly with bigger datasets

- ML has eveolved with the size of the datasets over the decades.



Stochastic Gradient Ascent

- Small change to the Gradient ascent algorithm.
- Takes a massive datatset and current parameters (**w(t)**).
- Computes the gadrient for only a small subset of data rather then the entire dataset. Then it updates the coefficients.



Scaling ML with stochastic gradient

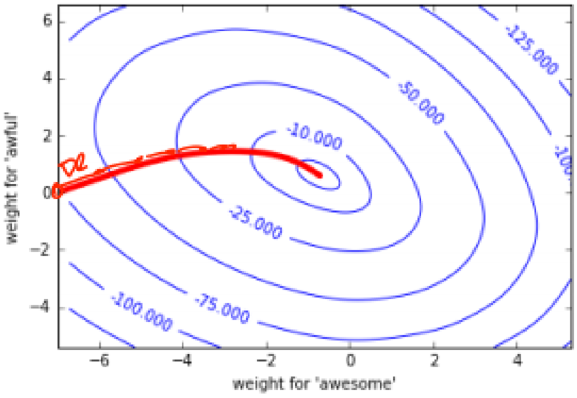
Why gradient ascent won't scale?

- Learning one data point at a time.
- Start at some point, and compute the gradient descent of the likelihood function and take a step in that space.
- Then compute the gradient again until the convergence is met, or stop condition.

Gradient descent expenses

- Sum over datapoint the contribution of each datapoint xiyi to gradient.

Gradient ascent



Algorithm:

while not converged
$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \eta \nabla \ell(\mathbf{w}^{(t)})$

How expensive is gradient ascent?

Sum over data points

$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} = \sum_{i=1}^{N} \underbrace{h_j(\mathbf{x}_i)\Big(\mathbb{1}[y_i = +1] - P(y = +1 \mid \mathbf{x}_i, \mathbf{w})\Big)}_{\text{Contribution of data point } \mathbf{x}_i, y_i \text{ to gradient}}$$

$$\boxed{\frac{\partial \ell_i(\mathbf{w})}{\partial \mathbf{w}_j}}$$

$$\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} = \sum_{i=1}^{N} \frac{\partial \ell_i(\mathbf{w})}{\partial \mathbf{w}_j}$$

Time to compute contribution of $\mathbf{x}_i, y_i$	# of data points (N)	Total time to compute 1 step of gradient ascent
1 millisecond	1000	1 sec
1 second	1000	16.7 min
1 millisecond	10 million	2.8 hours
1 millisecond	10 billion	115.7 days

Stochastic gradient

- Learning one data point at a time.
- Use one data-point to compute the gradient.
- Each time , while computing the gradient use a new data point i.
- Therefore. it will be computationally cheaper to perform stochastic gradient ascent when compared to gradient ascent.

gradient ascent for logistic regression

init $\mathbf{w}^{(1)}$=0, t=1
until converged
  for j=0,...,D
    partial[j] = $\boxed{\sum_{i=1}^{N}} h_j(\mathbf{x}_i)\Big(\mathbb{1}[y_i = +1] - P(y = +1 \mid \mathbf{x}_i, \mathbf{w}^{(t)})\Big)$ (Sum over data points)
    $w_j^{(t+1)} \leftarrow w_j^{(t)} + \eta$ partial[j]
  t ← t + 1

Stochastic gradient ascent for logistic regression

init $\mathbf{w}^{(1)}$=0, t=1
until converged
  $\boxed{\textbf{for } i=1,\dots,N}$
  for j=0,...,D
    partial[j] = $h_j(\mathbf{x}_i)\Big(\mathbb{1}[y_i = +1] - P(y = +1 \mid \mathbf{x}_i, \mathbf{w}^{(t)})\Big)$ (Each time, pick different data point i)
    $w_j^{(t+1)} \leftarrow w_j^{(t)} + \eta$ partial[j]
  t ← t + 1

Comparing computational time per step

Gradient ascent: $\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} = \sum_{i=1}^{N} \frac{\partial \ell_i(\mathbf{w})}{\partial \mathbf{w}_j}$

Stochastic gradient ascent: $\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} \approx \frac{\partial \ell_i(\mathbf{w})}{\partial \mathbf{w}_j}$

Time to compute contribution of $\mathbf{x}_i, y_i$	# of data points (N)	Total time for 1 step of gradient	Total time for 1 step of stochastic gradient
1 millisecond	1000	1 second	1 millisecond
1 second	1000	16.7 minutes	1 sec
1 millisecond	10 million	2.8 hours	1 millisec
1 millisecond	10 billion	115.7 days	1 millisec

Instead of all data points for gradient, use 1 data point only

Gradient ascent: $\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} = \sum_{i=1}^{N} \frac{\partial \ell_i(\mathbf{w})}{\partial \mathbf{w}_j}$ (Sum over data points)

Stochastic gradient ascent: $\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}_j} \approx \frac{\partial \ell_i(\mathbf{w})}{\partial \mathbf{w}_j}$ (Each time, pick different data point i)

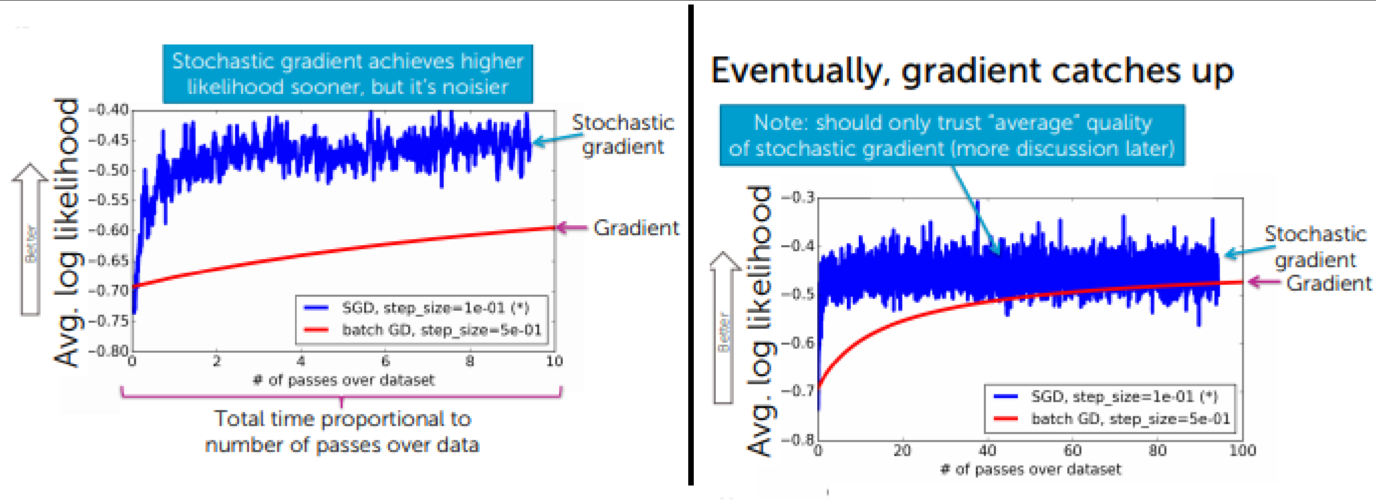
Compare gradient to stochastic gradient

- Stochatic gradient is computationally cheaper for large datasets.
- In practice stochastic gradient is often faster than gradient.
- But stochastic gradient is **'very sensitive to parameters'**. Choice of step size, etc.
- A graph is drawn between x-axis (# passes over dataset) & y-axis (average likelihood);
  - higher the likelihood, better it is.

- From the graph, after 10 passes, gradient has much lower likehood when compared to stochastic gradient.
- stochastic gradient achieves higher likelihood sooner, but it is noiser.
- stochastic gradient converges faster but oscillates more.
- eventually the gradient descent catches up. It is smooth.

Comparing gradient to stochastic gradient

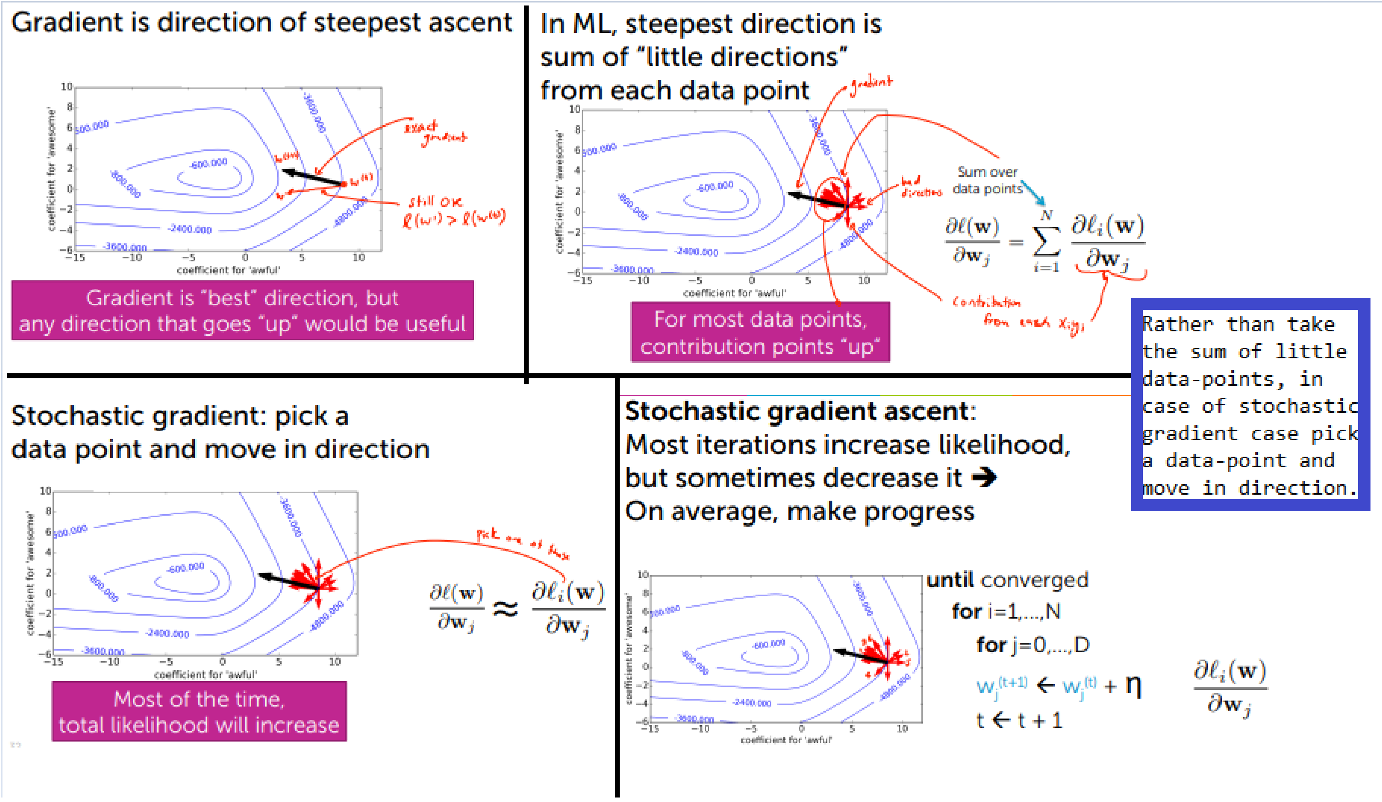
Algorithm	Time per iteration	Total time to convergence for large data: In theory	Total time to convergence for large data: In practice	Sensitivity to parameters
Gradient	Slow for large data	Slower	Often slower	Moderate
Stochastic gradient	Always fast	Faster	Often faster	Very high



Understanding why stochastic gradient works

Why would stochastic gradient ever work?

- The Gradient is the "best" direction, but any direction that goes "up" would be useful.
- In machine learning the steepest direction is the sum of "little directions" from each data-point. For most data-points contributuons points "up" - ascent.
- In case of the gradient - the grdient is the likelihood of sum over all data-points.
- In case of the stochastic gradient - pick a data-point and move in direction.
- In case stochastic gradient - total likelihood will increase.

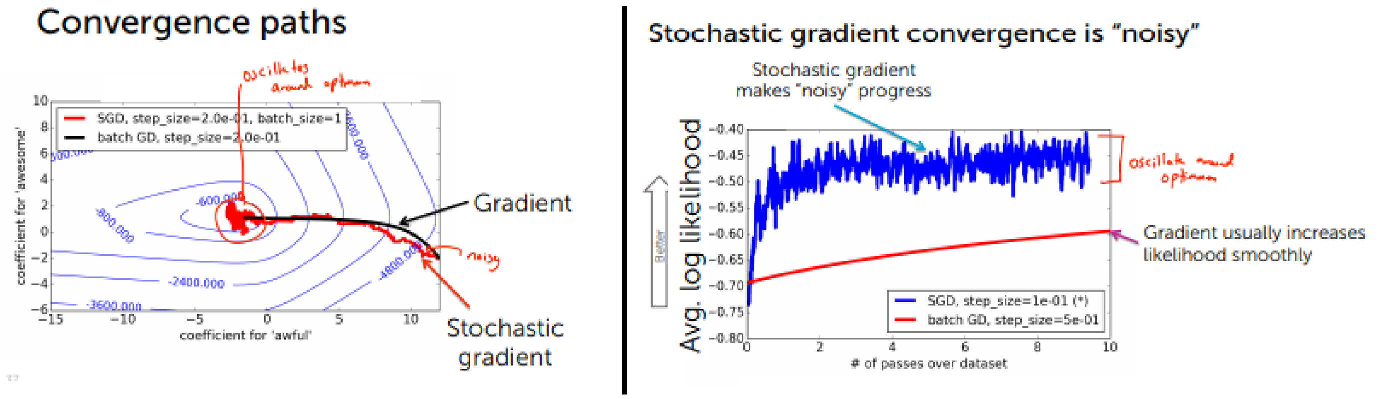


Rather than take the sum of little data-points, in case of stochastic gradient case pick a data-point and move in direction.

Convergence path

- Gradient finds the direction of steeps ascent. In converges smoothly.
- Gradient is the sum of contributions from each data-point.

- Stochastic gradient is the sum of contributions from each data-point.
- On an avrage increase likelihood and sometimes decrease.
- Stochastic gradient has a nosiy convergence. It oscillates around the optimum. (Issue);



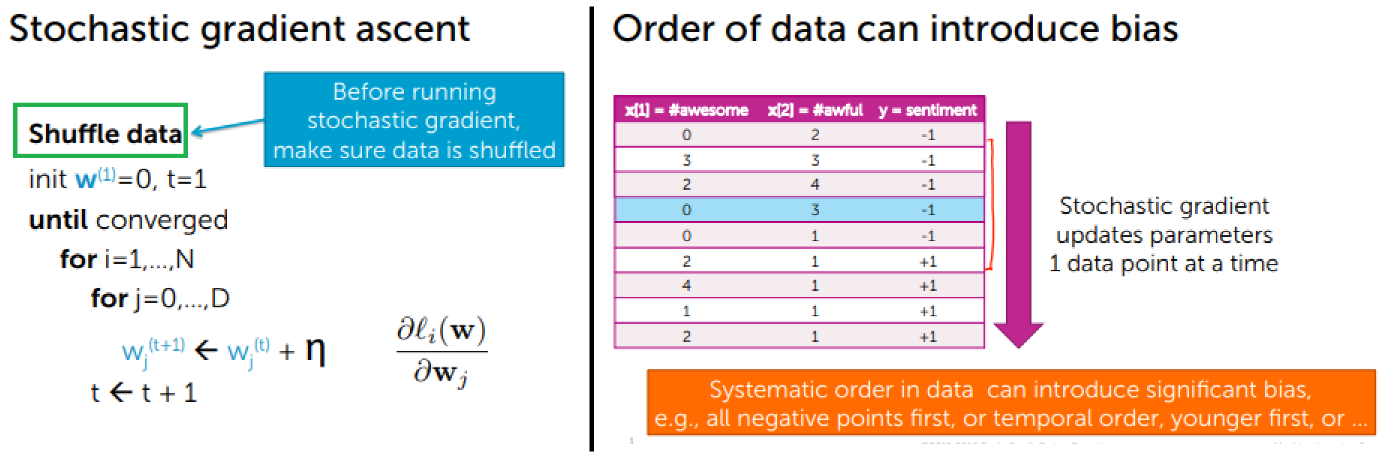
Stochastic Gradient : Practical Tricks

Shuffle data before running stochastic gradient

- If the data is implictly sorted , it can cause issues.
- Systematic order of data (like y is -1 comes before all the y is +1) can introduce bias.
- Therefore shuffle the dataset. No long regions in the dataset.

Algorithm:

1. Shuffle the data
2. initialize the coefficients/parameters = 0 at t=1
3. until convergence
    - for any dataset i
        - for the features j
            - compute the new coefficient -> wj(t+1) <- wj(t) + (step-size) * gradient of data-point;
4. optimum



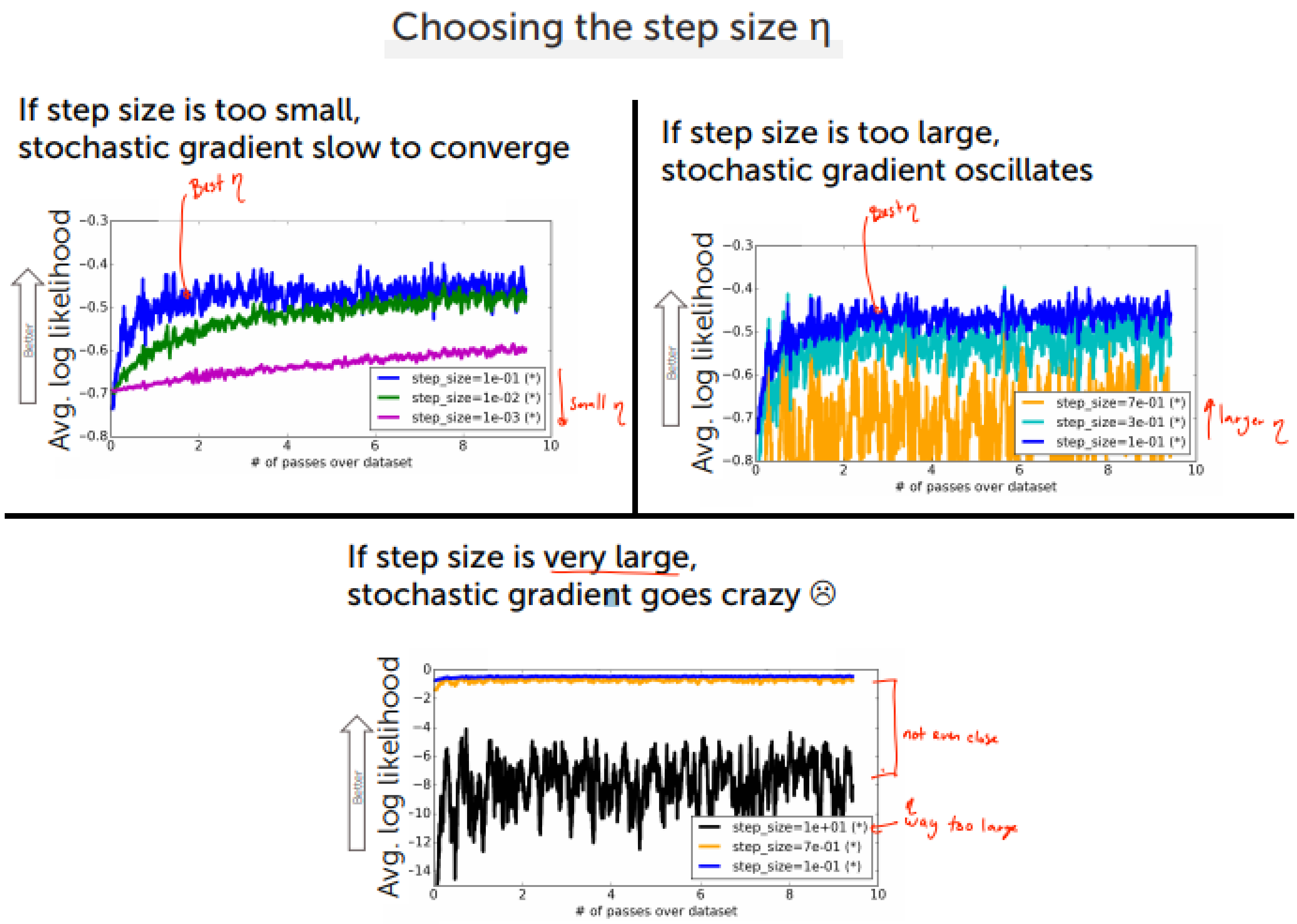
Choosing a step-size

- Picking step-size for **stochastic gradient** is very similar to picking a step-size for gradient.
- But since stochastic gradient is more noiser and fluctuates more frequently, it is a lot more unstable.
- If step-size is too small, stochastic gradient slows to converge.
- If step-size is too large, stochastic gradient oscillates a lot.
- If step-size is very large, stochastic gradient goes crazy.

Simple rule of thumb for picking step size η similar to gradient

- It requires a lot of trial and error.
- Try several values exponentially spaced.
    - Goal: plot learning curves to
        - find one η that is too small.
        - find one η that is too large.
- Advanced tip : step-size that decreases with iterations is very important for stochastic gradient.

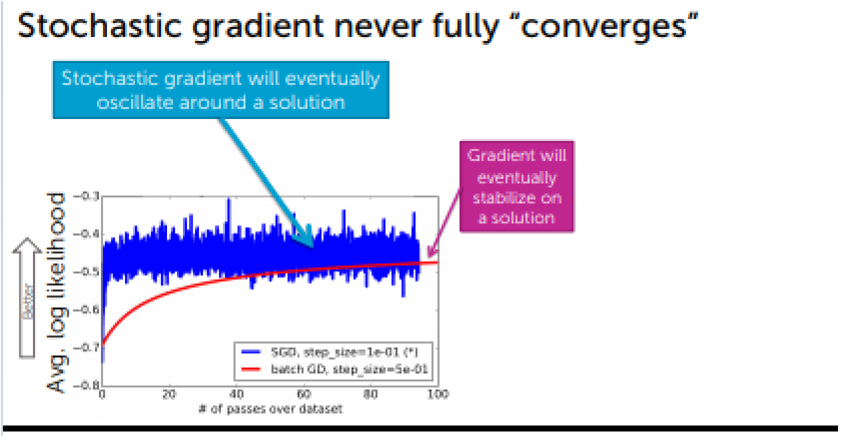
η(t) = η0 / t; (η0 -> constant; t -> iteration #; η(t) -> step-size for that iteration);

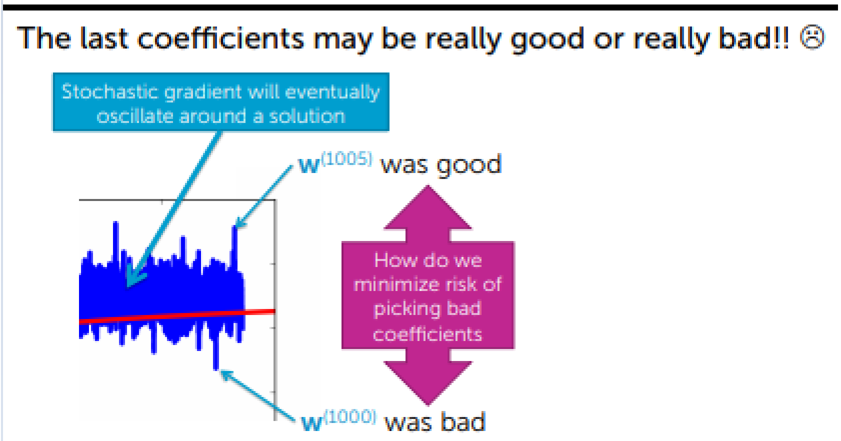


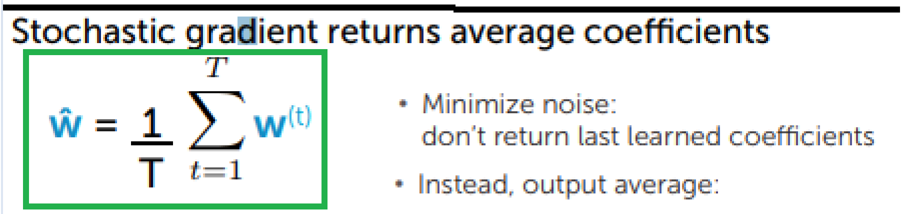
Stochastic Gradient convergence

- Stochastic gradient oscillates vastly over the minimum. Therefore cannot trust the last parameter/coefficient.
- The gradient will eventually stabilize must the stochastic gradient oscillats around the solution.
- Stochastic gradient return the average coeffients.

w-hat = (1/T) Σ (t=1,..,T) * w(t);

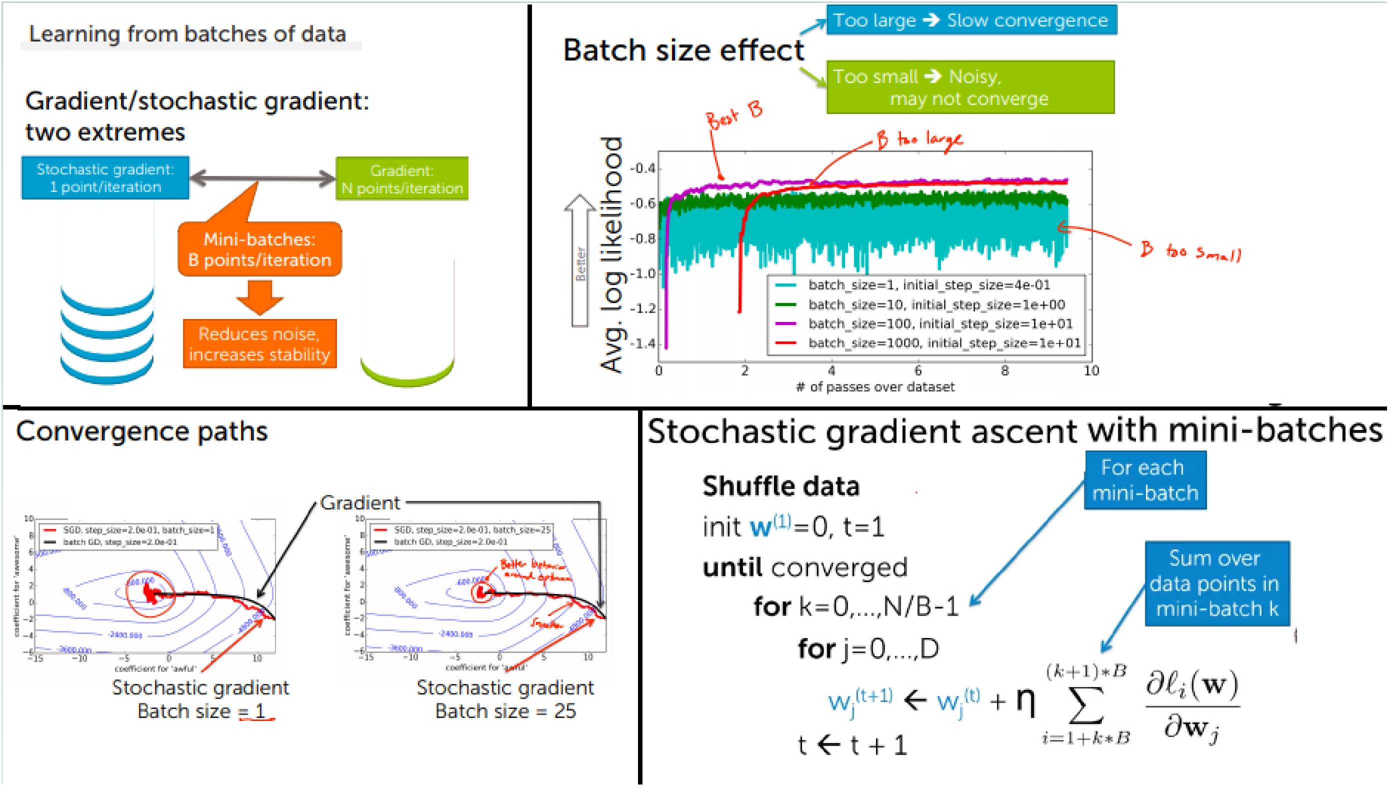






Learning from batches of data

- Learning from just one data-point is too noisy, therefore prefer mini-batches.
- The gradient descent -> pass over all data-points and stochastic gradient -> uses only one data-point are at the extremes. Require something is between.
- **Mini-batches**: B points/iteration -> reduces noise, increases stability.
- Stochastoc gradient with batch-size of 25 when compare to a batch-size of 1 has a smoother curve towards convergence and oscillates less at the optimum.
- Batch-size must not be too-large (gradient - slow to converge) or too small (batch-size -1 -> noisy);



Measuring Convergence

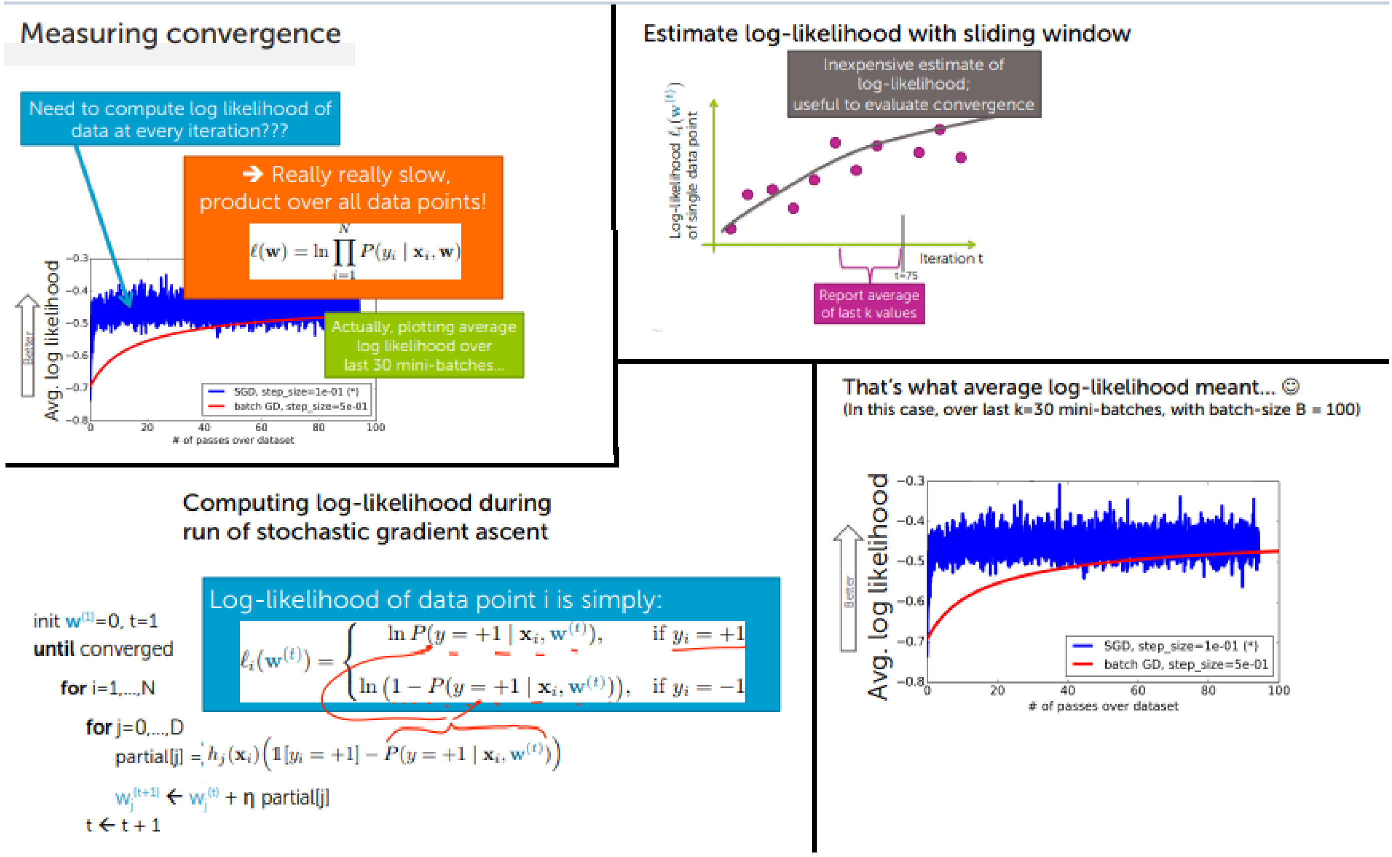
- Stochastic gradient gets to the optimum before a single pass over the data. While the gradient takes 100 or more passses over the data.
- Inorder to get one point on the plot (#passes over dataset v/s avg likelihood) then if we had to compute the whole likelihood over the entire dataset, then it would require to compute the product over all the dataset , makes the process much slower.
- Therefore measure the log-likelihood based on the probability.

Estimating the log-likelihood with sliding window

- For every iteration t can compute the likelihood of a particular data point.
- We can't use this value to measure the conversion because it would do well on one data point classified perfectly but not for others -> leading to noise.
- In order to compute the progress after certain iterations. Eg t = 75, report the average of the last k values (Take the likelihood for the last few datapoints, average it and create a smoother curve).

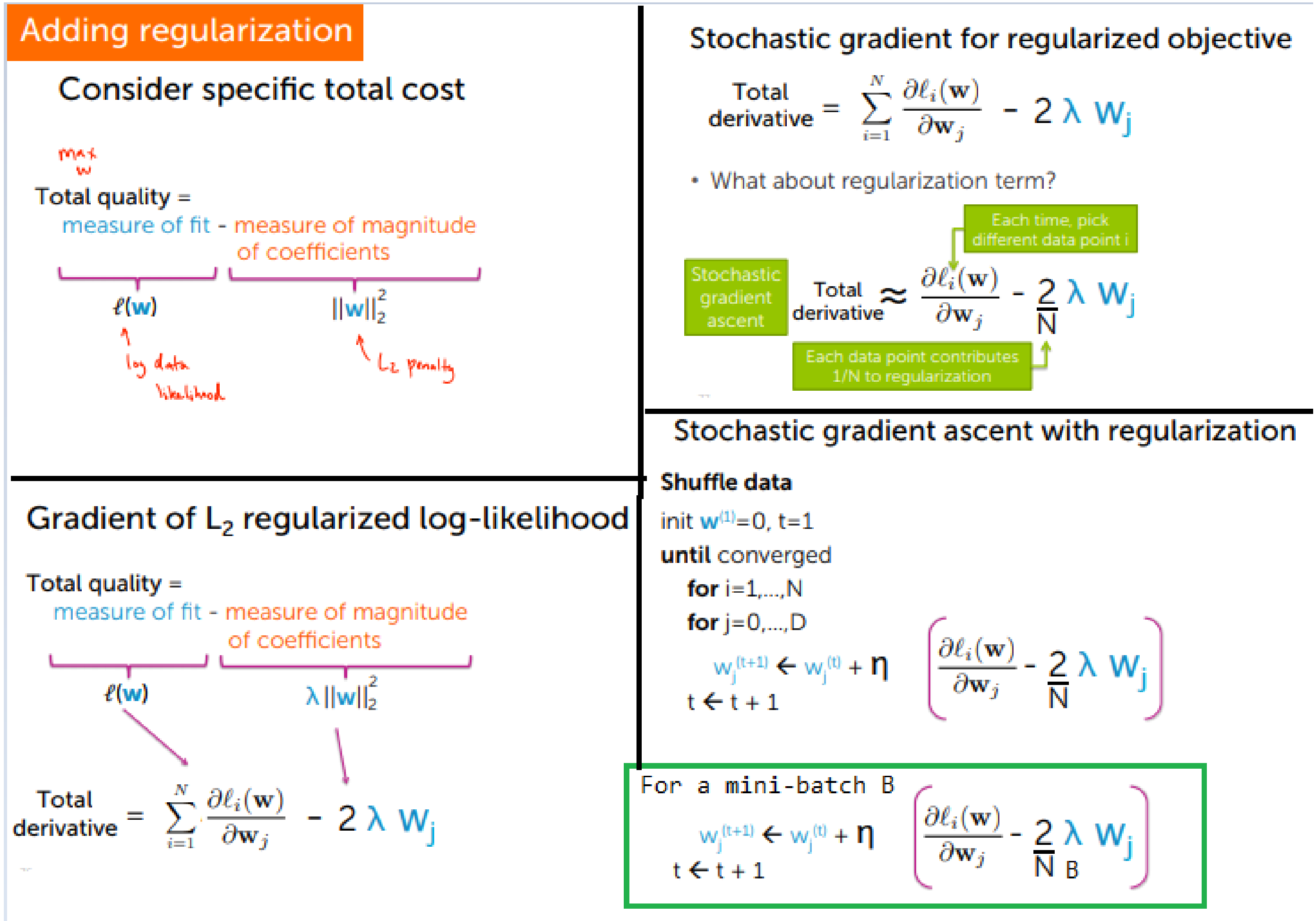
For every timestamp need to keep an average of the last few likelihoods in order to measure the convergence.

- Minibatches on size = 100.Converges faster than gradient.
- Inorder to draw the blue line -> need to average the likelihood over the last 30 data-points.



Adding regularization

- Stochastic gradient -> take a contribution for a single data point and add the contributions we get the gradient. Sum of the stochastic gradients is equal to gradient.
- Total cost for the algorithm
  - Total quality = measure of fit - measure of magnitude of coefficients
  - Total quality = log data likelihood - L2 penalty
  - Total quality = l(w) + (lambda) ||w||2 ^2
  - lamba -> tuning/regularization parameter;
- Gradient - Derivative of the Total cost
  - Total derivative = sum of the derivative of likelihood - 2 *lambda* Wj;
- Stochastic gradient -> gradient over a data-point or a batch
  - Need to introduce the regularization parameter while computing stochastic gradient.
  - Consider each datapoint contributes 1/N to regularization.
  - Adding all the stochastic gradients will give the gradient descent.



Online Learning

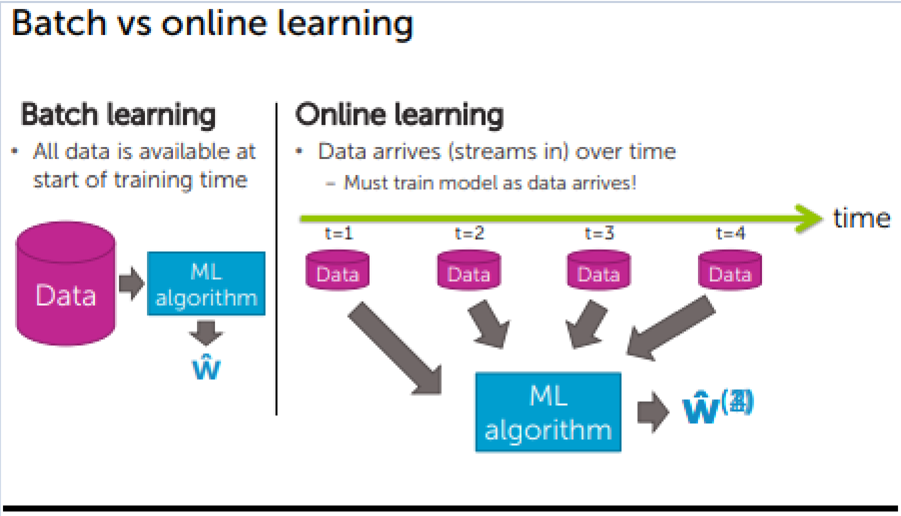
The online learning task

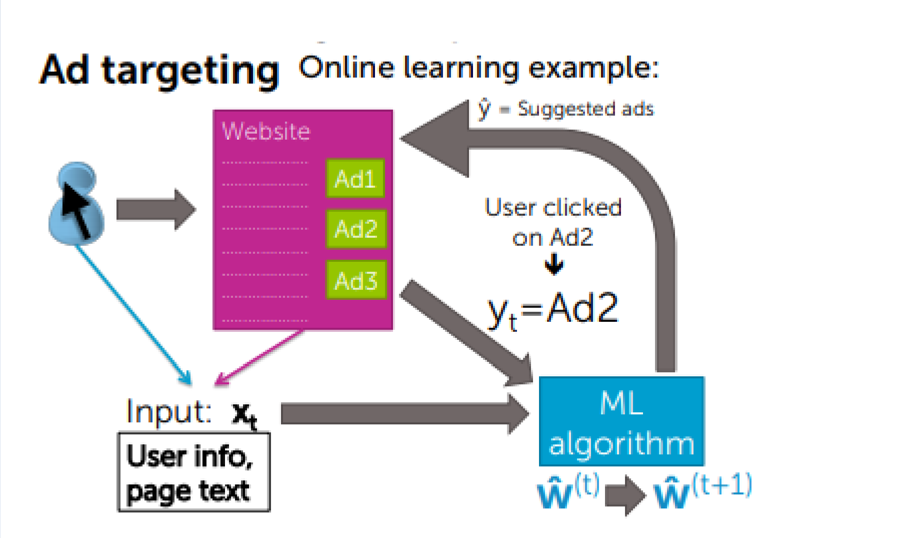
- Batch leaning -> All data is available at the start of training time.
- Online learning -> Data arrives (streams in) over time.
  - Must train the model as data arrives.
  - Compute the coefficients as data is feed to the model.

Online Learning Example:

Ad targeting

- Consider visiting a websites, that shows a lot of ads.
  - The input -> user info -age, previous websites, etc -> who visits the pages is fed to the machine learning algorithm.
  - This machine learning algorithm is gonna use some set of coefficients w-hat(t) and predict the best ads to show.
  - The ads shown are -> **y-hat (suggested ads)**.
  - In case the user clicks any ad shown, then machine learning algorithm figure that the user chose a particular ad.
  - The ML assign yt, true label for the particular chosen ad in the website.
  - The ML algoithm updates the coefficient w(t) -> w(t+1);





Using stochastic-gradient for online learning

Online learning problem

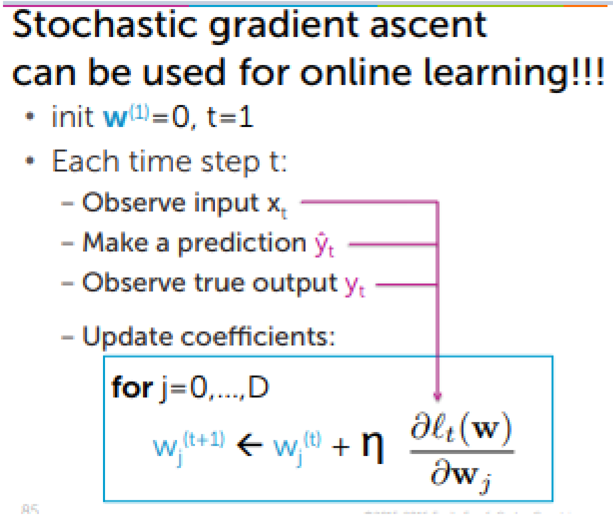
- Data arrives over each time step t:
  - **Observe input Xt**
    - Info of user,text of webpage, etc;
  - **Make a prediction y-hat(t)**
    - Which ad to show
  - **Observe true output y(t)**
    - Which ad use clicked on
  - Feed the information **ML Algorithm** to update coefficients each time.

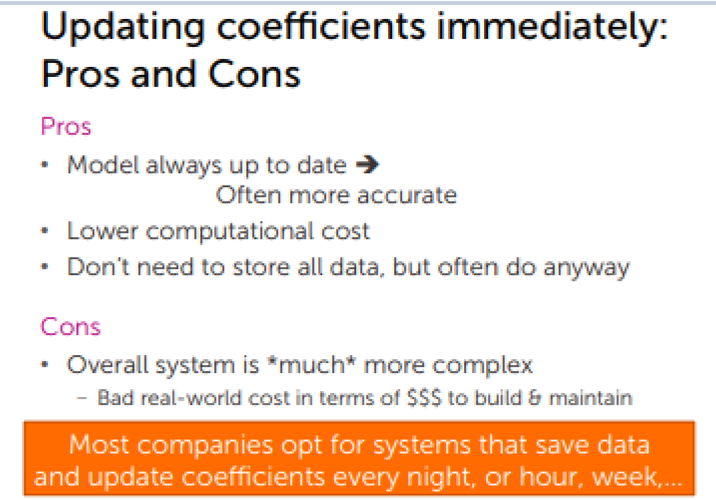
Stochastic gradient ascent can be used for online learning.

As Stochastic gradient uses a small number of data-points to compute coefficients it can be used for online learning.

- init w(1) = 0, t = 1
- Each time step t:
  - Observe input x(t)
  - Make a prediction y-hat(t)
  - Observe true output y(t)
  - Update coefficients

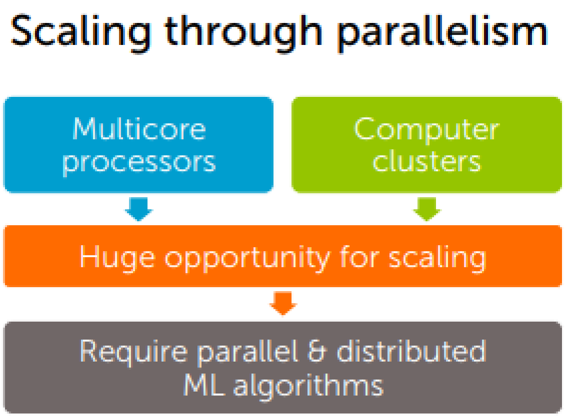
- for j = 0,...,D wj(t+1) <- wj(t) + η * (gradient likelihood);





Scaling huge datasets & Online learning

- Stochastic gradient is a means to scale large huge datasets and online learning.
- Scaling through parallelism



Quiz

1. (True/False) Stochastic gradient ascent often requires fewer passes over the dataset than batch gradient ascent to achieve a similar log likelihood.

   - (•) True
   - ( ) False

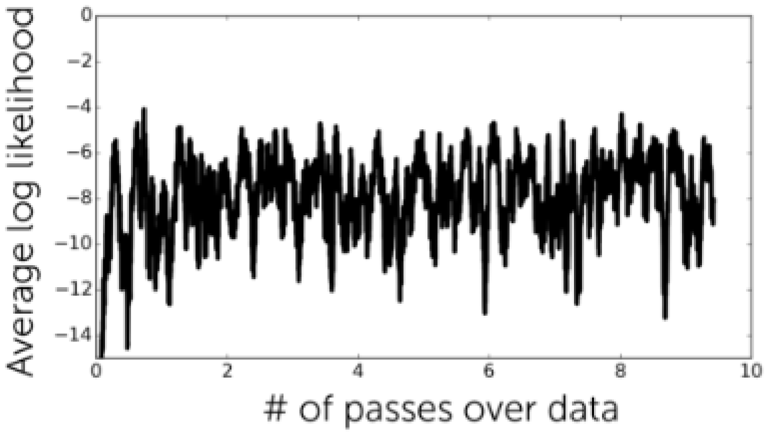
2. (True/False) Choosing a large batch size results in less noisy gradients

   - (•) True
   - ( ) False

3. (True/False) The set of coefficients obtained at the last iteration represents the best coefficients found so far.

- ○ True
- ◉ False

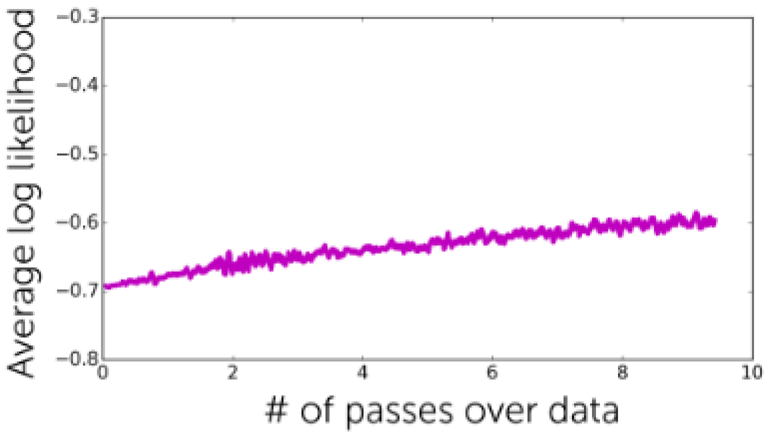
4. Suppose you obtained the plot of log likelihood below after running stochastic gradient ascent.



Which of the following actions would help the most to improve the rate of convergence?

- ○ Increase step size
- ◉ Decrease step size
- ○ Decrease batch size

5. Suppose you obtained the plot of log likelihood below after running stochastic gradient ascent.



Which of the following actions would help to improve the rate of convergence?

- ○ Increase batch size
- ◉ Increase step size
- ○ Decrease step size

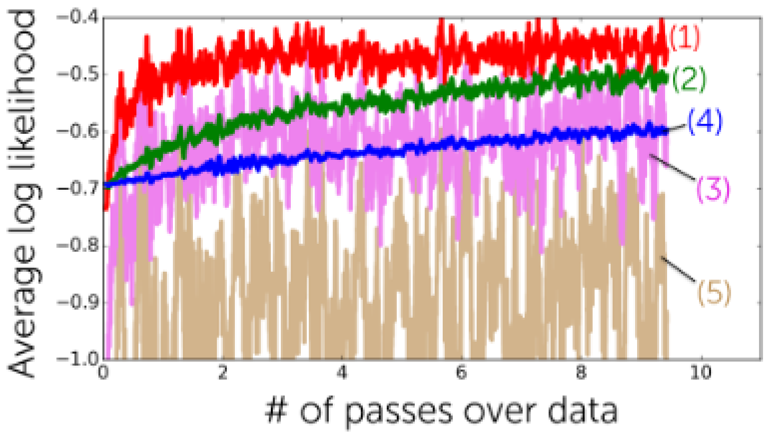
- purple line = step size is too small, takes too long to converge.
- to improve rate of convergance : increase step size.

6. Suppose it takes about 1 milliseconds to compute a gradient for a single example. You run an online advertising company and would like to do online learning via mini-batch stochastic gradient ascent. If you aim to update the coefficients once every 5 minutes, how many examples can you cover in each update? Overhead and other operations take up 2 minutes, so you only have 3 minutes for the coefficient update.

180000

- 3 minutes for coefficient update x 60 seconds x 1 millisecond to compute gradient for a single example = 180,000 gradient examples can be calculated.
- t=time to contribute contribution of x, N data points.
- gradient requires N x t time to calc 1 step of gradient
- stochastic gradient requires t time to calc 1 step of gradient. (ie: not affected by number of data points)
- ANSWER = 180,000

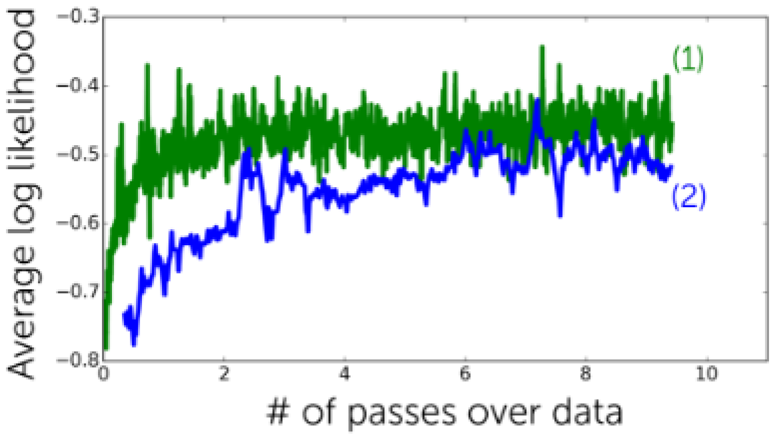
7. In search for an optimal step size, you experiment with multiple step sizes and obtain the following convergence plot.



Which line corresponds to the best step size?

- (1)
- (2)
- (3)
- (4)
- (5)

8. Suppose you run stochastic gradient ascent with two different batch sizes. Which of the two lines below corresponds to the smaller batch size (assuming both use the same step size)?



- (x) (1)
- ( ) (2)

- too small batch size will be noisy and may not converge.

9. Which of the following is NOT a benefit of stochastic gradient ascent over batch gradient ascent? Choose all that apply.

- [ ] Each coefficient step is very fast.
- [x] Log likelihood of data improves monotonically.
- [ ] Stochastic gradient ascent can be used for online learning.
- [ ] Stochastic gradient ascent can achieve higher likelihood than batch gradient ascent for the same amount of running time.
- [x] Stochastic gradient ascent is highly robust with respect to parameter choices.

10. Suppose we run the stochastic gradient ascent algorithm described in the lecture with batch size of 100. To make 10 passes over a dataset consisting of 15400 examples, how many iterations does it need to run?

1540

- 15400/100 = 154 batches.
- 10 passes x 154 batches = 1540

In [ ]: