ASSIGNMENT 1: OPTIMIZERS

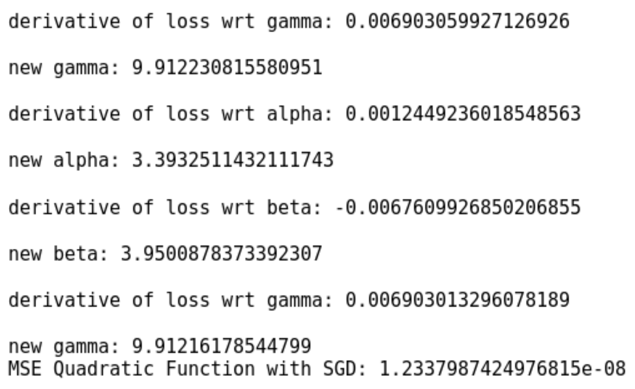
Rubén Martínez Sánchez de la Torre

1. **Replace the linear function with a quadratic function alpha +_beta*x + gamma*x**2. Feel free to name your variables in any way you like. Suggested value of gamma_true is 8.0, but feel free to use any other value. Create the corresponding animation.**

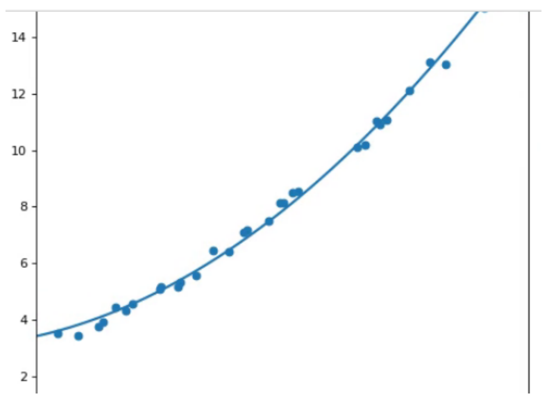
- In this case, we're going to update the parameters (alpha, beta and gamma) using the following rule:
- **param = param – learning_rate * gradient_wrt_param**
- The intuition behind this is that we're moving in opposite direction to the gradient of the loss function with respect to each parameter.
- To reach that goal we're going to repeat the SGD algorithm several iterations and we're going to update the parameter on each step.
- If we choose a very high value of the learning rate we can skip the global minimum, oscillating between the loss surface without reaching the convergence.
- If we choose a very low value of the learning rate we're going to move towards the global minimum very slow
- In the experiments showed in the Jupyter notebook we can see how the Mean Squared Error (MSE) is going down not too aggressively between steps until reaching an approximate value of 3e-09.
- I've included in the Jupyter notebook text messages showing the evolution of the gradients with respect to each parameter and the corresponding MSE.

2. **Implement the following optimizers:**

- **SGD:** It has been explained in the first section.

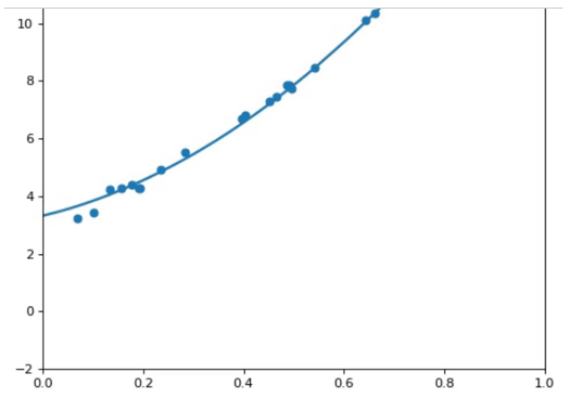


```
derivative of loss wrt gamma: 0.006903059927126926

new gamma: 9.912230815580951

derivative of loss wrt alpha: 0.0012449236018548563

new alpha: 3.3932511432111743

derivative of loss wrt beta: -0.0067609926850206855

new beta: 3.9500878373392307

derivative of loss wrt gamma: 0.006903013296078189

new gamma: 9.91216178544799
MSE Quadratic Function with SGD: 1.2337987424976815e-08
```



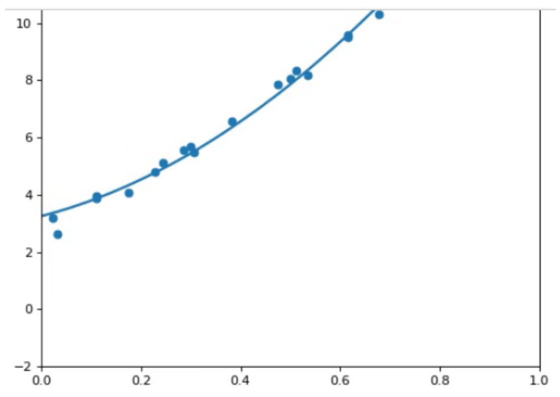
- **Gradient Descent with Momentum:** In this technique, we're going to accumulate the gradients of the past steps to guide the search of the minimum value of the loss function. To compute this momentum we're going to use a **velocity variable 'v'** that will be responsible of retain gradients of previous iterations and a **coefficient of momentum** (we're going to call it **'m'** which takes values like 0.5 or 0.9) which is the percentage of the gradient retained each step: **v_param = m*v_param – learning_rate * gradient_wrt_param**. We can see that the weight of the most recent gradients is more than the oldest. Finally, we're going to update each parameter by adding its corresponding v_param: **param = param + v_param**. This method can help to avoid local minimums using that momentum. At the end of the experiment we have an approximate MSE of 4e-08.

```
derivative of loss wrt gamma: 0.006512367846138404

new gamma: 9.729334410390825

derivative of loss wrt alpha: 0.001330192649176857

new alpha: 3.3095879554210987

derivative of loss wrt beta: -0.006909710196339911

new beta: 4.225245013582185

derivative of loss wrt gamma: 0.006511835191568532

new gamma: 9.72920416303303
MSE Quadratic Function using SGD with Momentum: 4.1962612002726015e-08
```

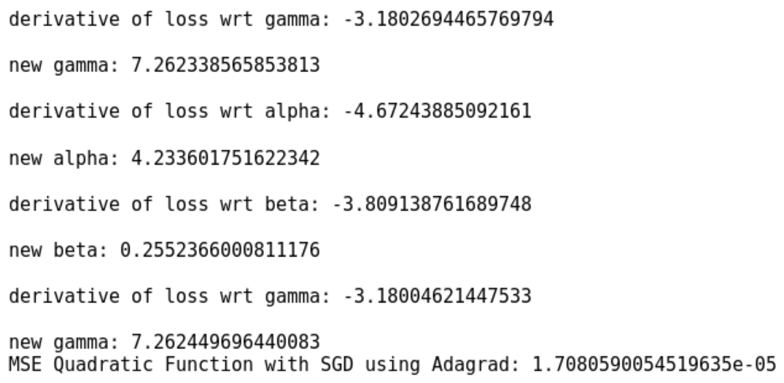


- **Gradient Descent with Nesterov Momentum:** This optimization algorithm adds an **interim update** so we first jump to the direction of previous accumulated gradients according to this formula: **param = m*v – learning_rate * gradient_wrt_param**. From that point we’re going to compute the gradient with respect to each parameter and after that we’re going to repeat the process of the SGD with Momentum, so we’re making a correction in the trajectory applying this formula: **v_param = m*v_param – learning_rate * gradient_wrt_param**. Once we’ve done that we will update the weights using the following equation: **param = param + v_param**. This method could be useful, for example, when the velocity added to the parameters causes exploding gradients giving you a high loss. In this case the SGD with Momentum method will have large oscillations but using Nesterov Momentum if the velocity guides you towards a bad loss then the gradient from that interim point will direct the update back towards the direction of the parameters before applying the interim update acting like a correction factor for the Momentum algorithm. In the following images we can see the last values of the gradients, the MSE and the graphic with the learned model (drew using a line) trying to approximate the real function (dots).

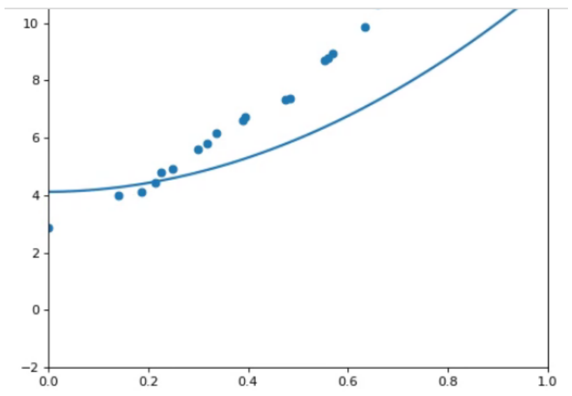
```
derivative of loss wrt gamma interim: 0.008308395791793462

new gamma: 9.393986077698738

derivative of loss wrt alpha interim: 0.0015828174204547628

new alpha: 3.2423640793087274

derivative of loss wrt beta interim: -0.008728846917270407

new beta: 4.531844244999807

derivative of loss wrt gamma interim: 0.008307495371414864

new gamma: 9.39381990978095
MSE Quadratic Function using SGD with Nesterov Momentum: 6.639705798314134e-08
```



- **Adagrad:** This method maintains and updates a separate learning rate for each parameter. If the gradients of some components are small this algorithm can help accumulating in each iteration the squared gradient of this step with the squared gradients added in previous steps using this equation: **r = r + g *g** (element-wise operations). The update operation for each parameter will add to the current parameter’s value a term composed by $\frac{-learning\,rate}{delta\cdot\sqrt{r}}\cdot g$ being delta a factor (e.g. 10e-7) for numerical stability. The effect will be that parameters associated with features that happen frequently (high partial derivative of the loss) will also suffer a high decrease in their learning rate meanwhile parameters linked to features that don't happen so frequently (small partial derivate of the loss) will have a small decrease in their learning rate. The results of the experiments can be observed in the following images.

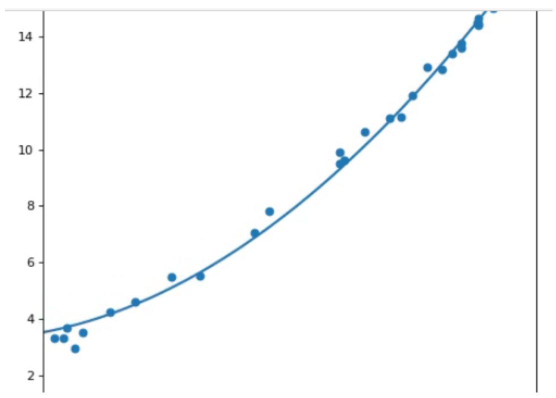


```
derivative of loss wrt gamma: -3.1802694465769794

new gamma: 7.262338565853813

derivative of loss wrt alpha: -4.67243885092161

new alpha: 4.233601751622342

derivative of loss wrt beta: -3.809138761689748

new beta: 0.2552366000811176

derivative of loss wrt gamma: -3.18004621447533

new gamma: 7.262449696440083
MSE Quadratic Function with SGD using Adagrad: 1.7080590054519635e-05
```



- **Adam:** It combines Momentum and RMSProp algorithms so it defines a variable 's' which acts as the first moment estimate and is similar to the velocity (used in SGD with Momentum) for each parameter to accumulate the current gradient with previous gradients according with $s = \rho_1 \cdot s + (1-\rho_1) \cdot g$. A typical value of ρ_1 (**decay rate for the first moment estimate 's'**)is 0.9 . Adam also uses a variable 'r' that reminds the trick of the RMSProp method taking into account the sum of the previous squared gradients with the current squared gradient. The precise formula used in this part of the algorithm is $r = \rho_2 \cdot r + (1-\rho_2) \cdot g \cdot g$ being ρ_2 (with suggested value of 0.999) the **decay rate for the second moment estimate 'r'**. As 's' and 'r' are biased to 0, Adam applies the following corrections for the bias in both terms: $\hat{s} = \frac{s}{1-\rho_1^t}$ and $\hat{r} = \frac{r}{1-\rho_2^t}$ being t the number of the current iteration. After that, each parameter is updated according the following rule: $param = param + \frac{-learning\ rate \cdot \hat{s}}{delta \cdot \sqrt{\hat{r}}}$ with a learning rate's suggested value of 0.001 and delta = 10e-8.

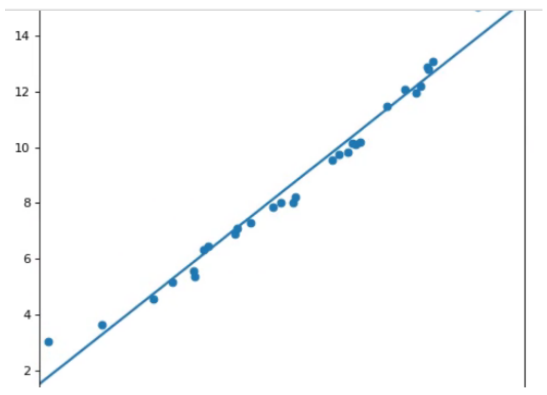
```
derivative of loss wrt gamma: 0.007298482354185904

new gamma: 10.397336484665566

derivative of loss wrt alpha: 0.010578734870700049

new alpha: 3.503800675302693

derivative of loss wrt beta: -0.021240902955432142

new beta: 3.3715116176643383

derivative of loss wrt gamma: 0.007311561832792261

new gamma: 10.397270206523093
MSE Quadratic Function with SGD using Adam: 4.806617477872448e-07
```



3. **Replace alpha +_beta*x + gamma*x**2 with alpha +_beta*x + 1e-6*gamma*x**2 and simultaneously make the value of gamma_true million times larger than previously. Then check again the performance of the same optimizers as in step 2.**

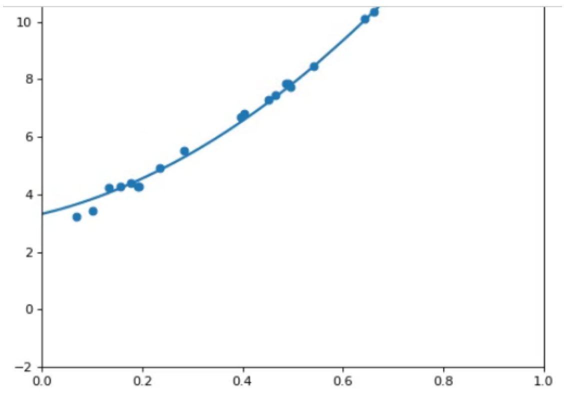
- **Modified Quadratic Function using SGD:**

```
derivative of loss wrt gamma: -0.06901354730746498

new gamma: 23.77975279418138

derivative of loss wrt alpha: 0.009951346721551705

new alpha: 1.4823872652253125

derivative of loss wrt beta: -0.01754628619323674

new beta: 13.798960560463655

derivative of loss wrt gamma: -0.0689977332115548

new gamma: 23.780442771513496
MSE Quadratic Function with SGD: 1.1193708418409947e-06
```



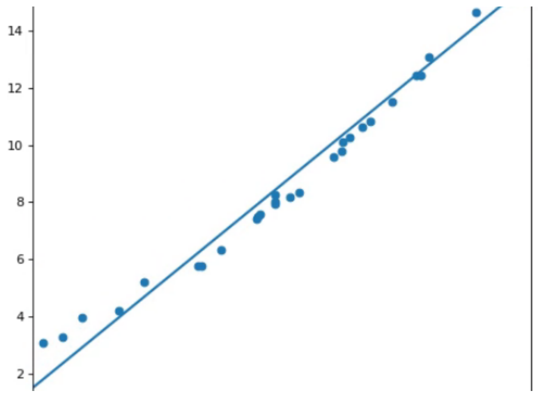
- **Modified Quadratic Function using SGD with Momentum:**

```
derivative of loss wrt gamma: 0.006512367846138404

new gamma: 9.729334410390825

derivative of loss wrt alpha: 0.001330192649176857

new alpha: 3.3095879554210987

derivative of loss wrt beta: -0.006909710196339911

new beta: 4.225245013582185

derivative of loss wrt gamma: 0.006511835191568532

new gamma: 9.72920416303303
MSE Quadratic Function using SGD with Momentum: 4.1962612002726015e-08
```



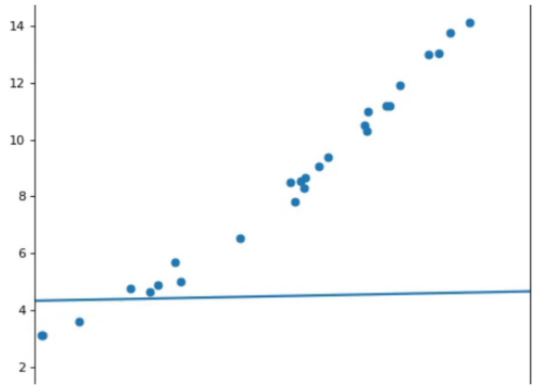
- **Modified Quadratic Function using SGD with Nesterov Momentum:**

```
derivative of loss wrt gamma interim: -0.09388042178640828

new gamma: 31.23182236657724

derivative of loss wrt alpha interim: 2.6311767415225045e-05

new alpha: 1.4856519779049635

derivative of loss wrt beta interim: -4.63881680597041e-05

new beta: 14.252349549046997

derivative of loss wrt gamma interim: -0.09388031534076159

new gamma: 31.233699975017302
MSE Quadratic Function using SGD with Nesterov Momentum: 3.226486683954177e-11
```



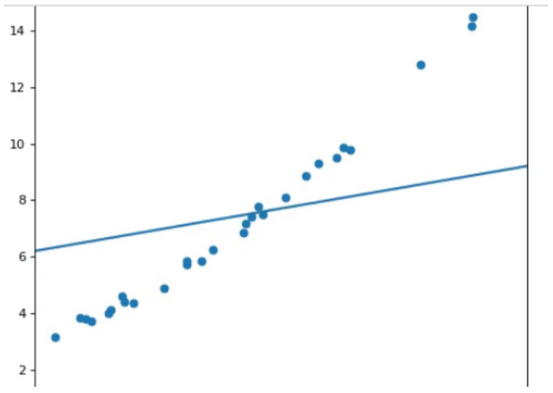
- **Modified Quadratic Function with Adagrad:**

```
derivative of loss wrt gamma: -6.353693718920605

new gamma: 7.3408460479544715

derivative of loss wrt alpha: -10.03094631748055

new alpha: 4.320460924802181

derivative of loss wrt beta: -7.865222098896018

new beta: 0.3355440559469559

derivative of loss wrt gamma: -6.353516353375317

new gamma: 7.340974495857899
MSE Quadratic Function with SGD using Adagrad: 1.4644468467689294e-05
```



- **Modified Quadratic Function with Adam:**

```
derivative of loss wrt gamma: -1.5835526738810939

new gamma: 10.260430155090225

derivative of loss wrt alpha: -0.23803153965699633

new alpha: 6.201110862228772

derivative of loss wrt beta: -1.5811536945315317

new beta: 3.0485515965201

derivative of loss wrt gamma: -1.58325344463588

new gamma: 10.261200719837035
MSE Quadratic Function with SGD using Adam: 9.822580236235465e-05
```


14
12
10
8
6
4
2