Standard Configuration:

* 1 hidden layer with 20 nodes
* Cross entropy loss function
* ReLu activation function

Activation Functions:

**Softplus**

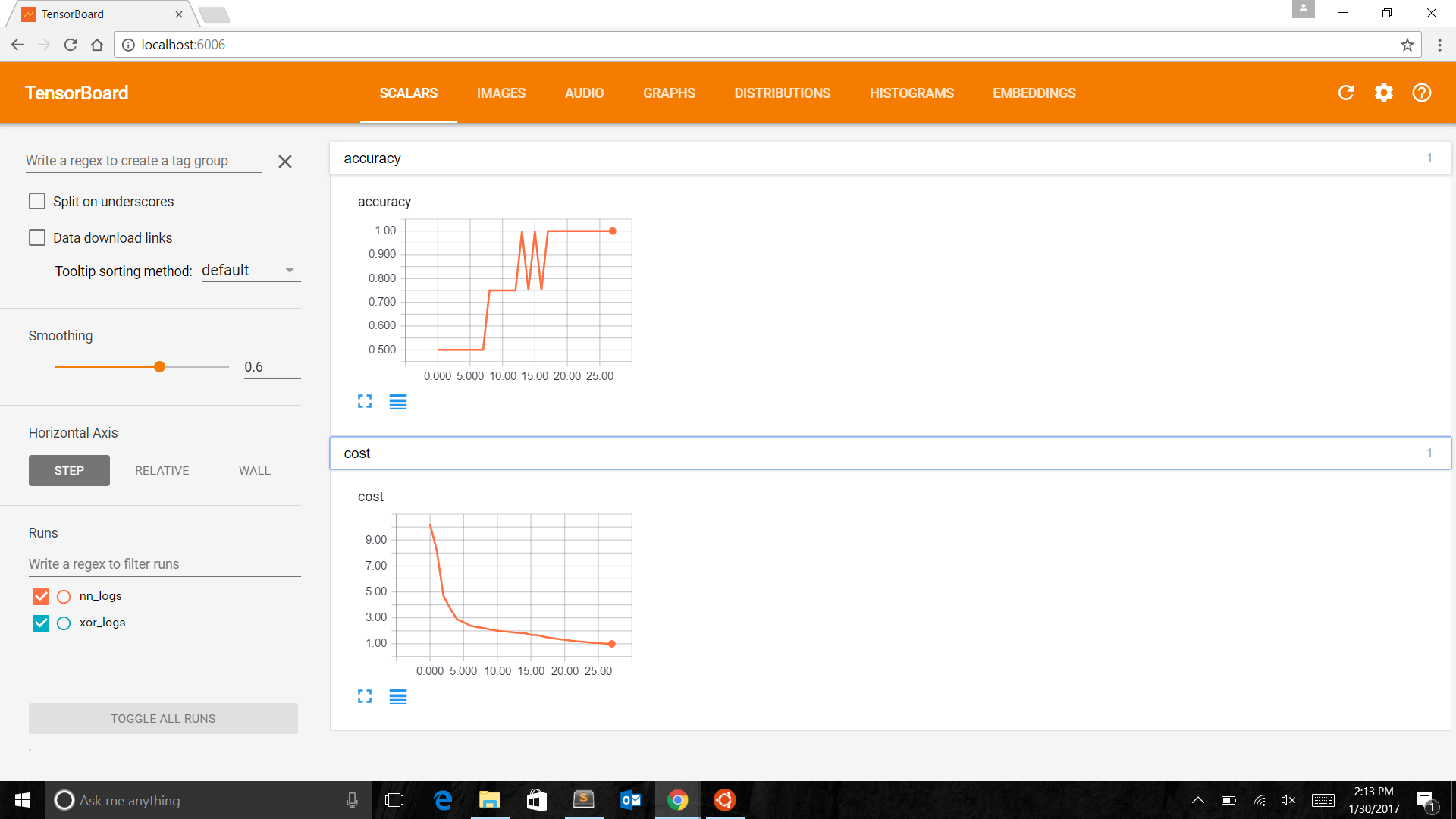
20 hidden units: within 528 steps accuracy 100%.

**Softsign**

20 hidden units: within 270 steps accuracy 100%.

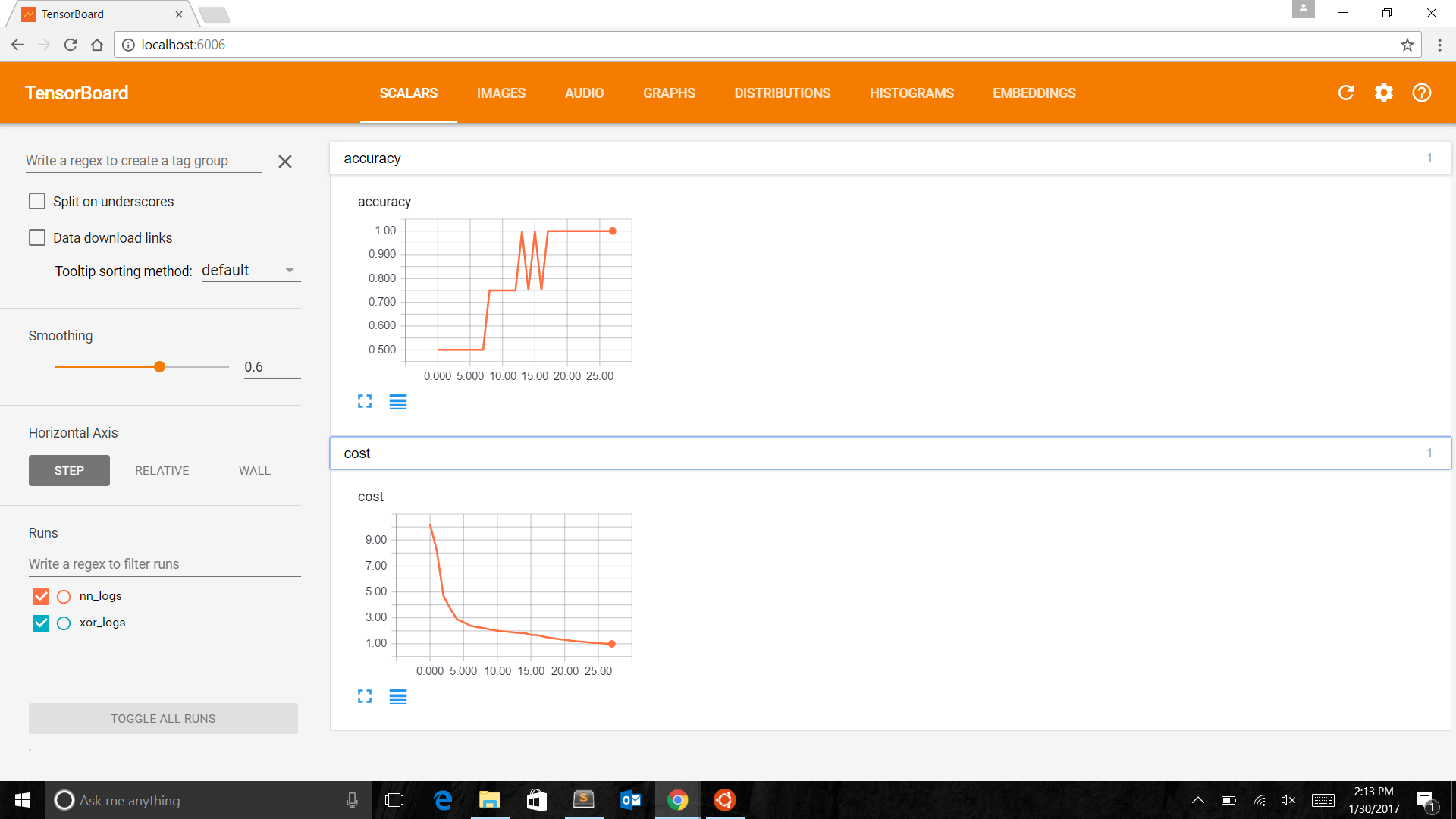
**Sigmoid**

20 hidden units: within 270 steps accuracy 100%.

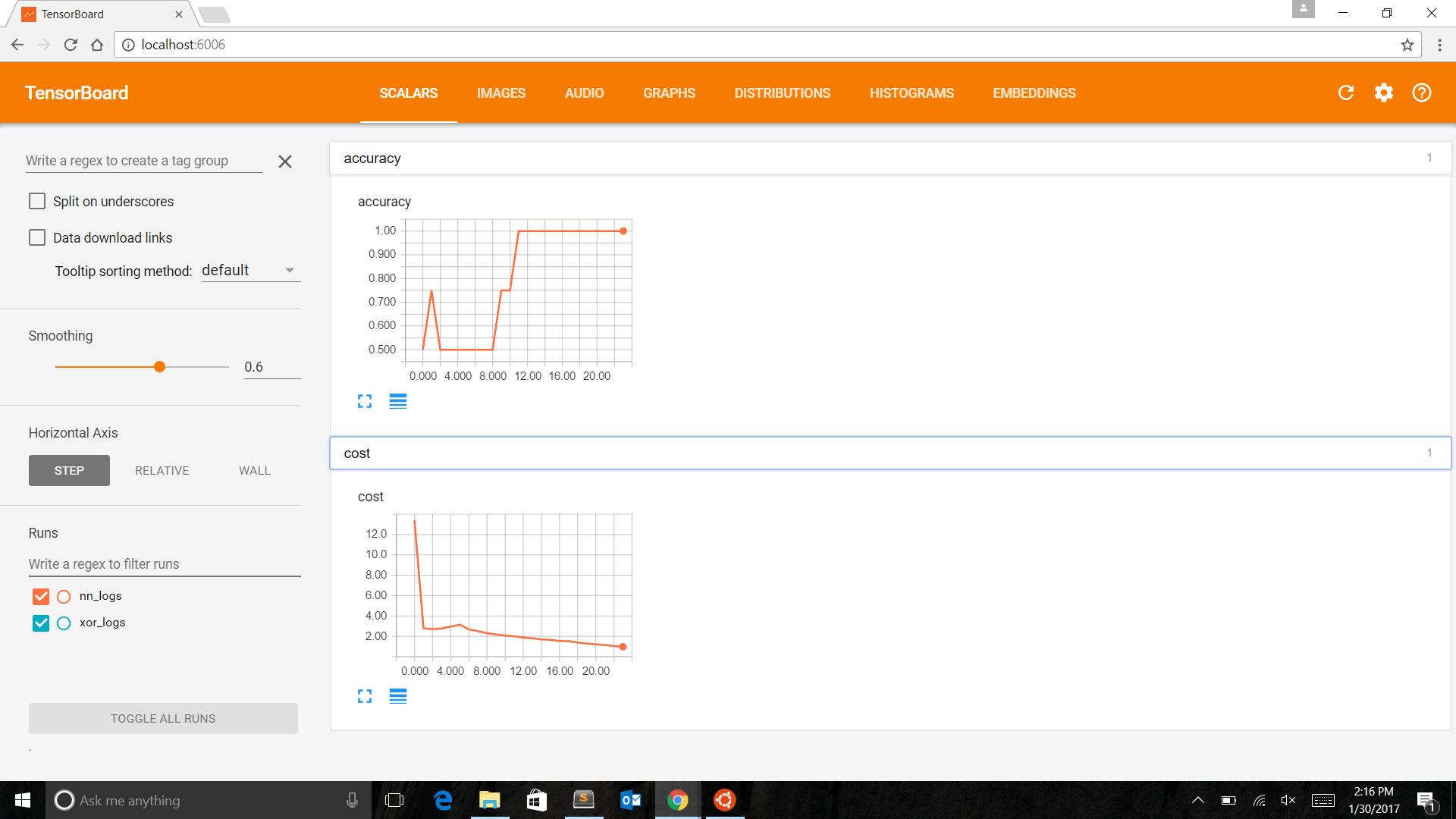


Optimizer:

1. Gradient Descent Optimizer with step size 0.2:100% accuracy.



**2.** Adagrad Optimizer with step size 0.2:100% accuracy. This took less time to converge.



3. Adam Optimizer with step size 0.2:75% accuracy. It could not converge in 1000 epochs.

Hidden Layers:

It is a well known fact that XOR gate needs at least one hidden layer. We tried running the model with one and two hidden layers with a different number of nodes in each layer.

**1 hidden layer:**

1. Hidden Units: 5 : We found that most of the time the loss converges within 80 iterations, and we have an accuracy of 75%
2. Hidden Units: 10: We found that most of the time the loss converges within 80 iterations, and we have an accuracy of 100% most of the time
3. Hidden Units: 20: We found that most of the time the loss converges within 50 - 70 iterations, and we have an accuracy of 100%
4. Hidden Units: 50: We found that most of the time the loss converges within 50 iterations, and we have an accuracy of 100%

**2 hidden layers:**

1. Training with 2 hidden layers, increases the number of iterations needed for convergence for all hidden units vs the model with 1 hidden layer.
2. Hidden Units: 5 : We found that most of the time the loss converges within 80 iterations, and we have an accuracy of 75%
3. Hidden Units: 10: We found that most of the time the loss converges within 80 iterations, and we have an accuracy of 100% most of the time
4. Hidden Units: 20: We found that most of the time the loss converges within 50 - 70 iterations, and we have an accuracy of 100%
5. Hidden Units: 50: We found that most of the time the loss converges within 50 iterations, and we have an accuracy of 100%

**Weight Initialization:**

Initialized weights and bias sampled from U(-1,1).

Then initialized weights as U(-(6/m+n)^0.5, (6/m+n)^0.5) and biases as 0.

In second case, the network converged very quickly. Initialization did not affect the accuracy though.

In standard configuration, the first case took roughly 60 epochs to converge, while the second one took about 15 epochs.