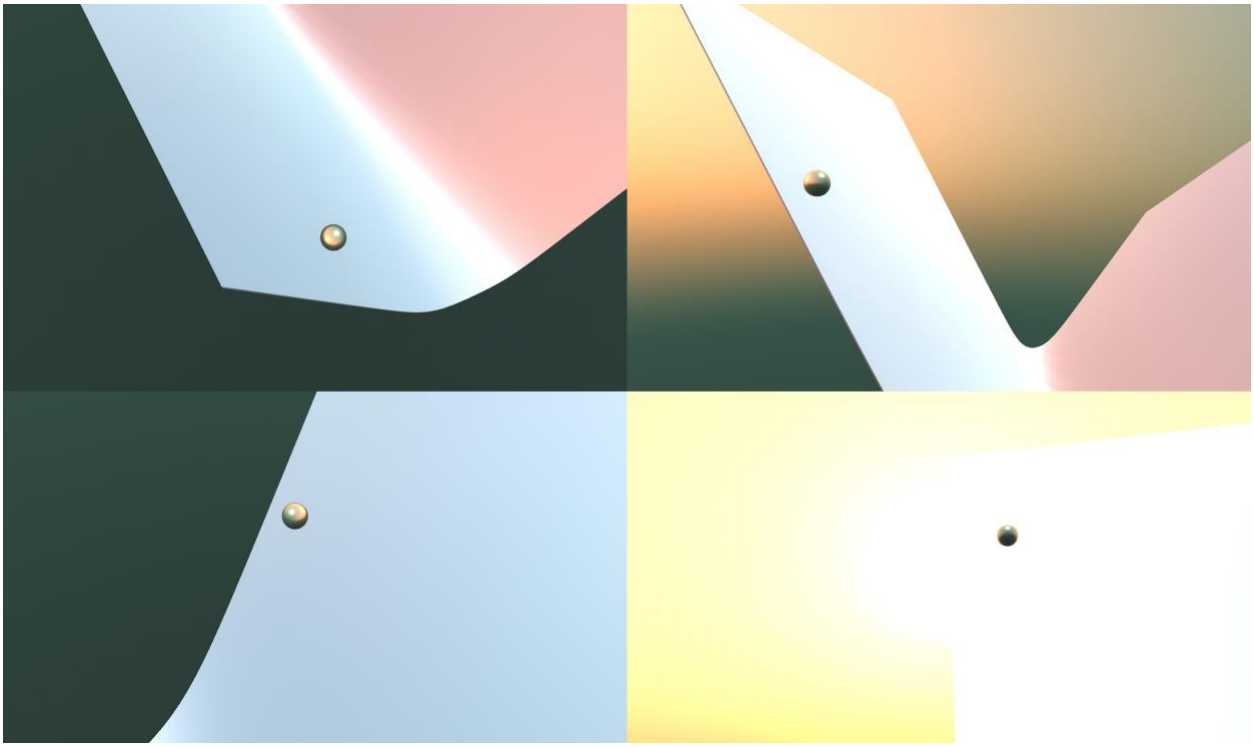
Shouvik Mani, Yang Yang, Hai Pham
10-615: Art and Machine Learning
May 7, 2018

Visualizing 3D Loss Landscapes for High-Dimensional Machine Learning Models



Sample 3D loss landscapes for various models generated using a PCA approach and a ball rolling down the path of gradient descent on each landscape

Concept

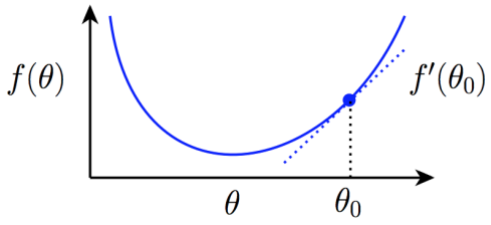
A machine learning problem can be characterized as an optimization problem where the goal is to minimize a loss function. A loss function is defined in terms of the data and the parameters of a model. For example, this is the least squares loss function.

$$\ell(\hat{y}, y) = (\hat{y} \ - y)^2$$

And the optimization problem is to find the parameters to minimize the loss (where θ are the parameters of the model):

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{m} \ell(h_\theta(x^{(i)}), y^{(i)})$$

For simple models that have few parameters, the loss function is easy to visualize. Further, when the loss function is a convex function, it is easy to optimize and reach the global optima in the loss landscape.



However, in many cases, the loss function is non-convex and multi-dimensional in terms of model parameters. These loss functions are not only computationally expensive to evaluate due to the large feature space, but also difficult to visualize given their large parameter space. Our project aims at visualizing the loss landscapes of high-dimensional models and illustrate how different optimizers traverse and look for an optimal point.

Current progress include [1] and [2] try to interpolate/reduce a multi-dimensional space into a 3D or 2D one where people can perceive easily. Those approaches, although theoretically sound, do not provide an appealing, multiview, animated and artistic landscapes. Furthermore, none of these works have released their code for public use. Our team, on one hand, wants to provide a new open source tool which can visualize such learning landscape with optimizers (such as SGD, RMSProp, Adagrad or Adam). This can be useful, for instance, in a case where we want to compare two solutions and reason why one solution is better than another one. For example, in deep learning,

where most of the models have millions of parameters, even one set of training hyperparameters will yield very different models at different points in time or on different environments. This hurdle of understanding how a deep learning model learns is due to our limited knowledge about how an optimizer learns to find the global minimal area in a non-convex settings, which are the cases of deep learning loss functions to optimize.

Despite the booming trend of high-dimensional models like deep learning, how these models learn is still a myth, partly due to the lack of intuitive tools which can visualize how a model is trained given a loss function. We hope to make the intuitive and artistic loss visualizations to stimulate more interests of the research community to work more on fundamental theoretical problems in machine learning.

Literature Review

Although there have been a myriad of research focusing on improving techniques of deep learning on improving the models, training techniques, and visualization of the parameters, there have been very limited works on understanding the optimization process in terms of visual aspect.

Ian Goodfellow *el al* [2] proposes a simple technique to interpolate the loss landscape (which is multi-dimensional) to a two-dimensional (2D) curve. They independently ran two rounds of training to find a two set of optimal solutions. Finally they fit the two solution with a 2D curve, where the vertical axis is the loss value, and the data axis is the fitting variable. Likewise, they introduce only one free variable to interpolate the loss surface. Nonetheless, this 2D plot is not intuitive enough to understand the intrinsic characteristics of an optimization surface.

An extension from that work is the recent result from Hao Li *et al* [1] in which they introduce another free variable, and thus extend the 2D plot into an much more intuitive 3D surface. By this technique, they are also able to investigate the effect of such training techniques as batch normalization on how they can help facilitate the training by transforming the optimization surface into a less non-convex one and make the optimizer easier and faster to find the optimal solution.

Nonetheless, the aforementioned works only focus on how to help human perceive the loss surface, yet have not been able to study how an optimizer learns to discover the optimal solution given that surface. Aiming to address this drawback, our work focuses on two targets: provide an intuitive visualization of the loss surface, and at the same

time visualize how the chosen optimizer traverses and finds the solution on a loss surface.

Another line of work which is close to ours is from Lorch *et al* [4]. They uses Principal Component Analysis (PCA) to reduce the dimensionality of neural network parameters to help reduce the complexity of the problem. They do not not, however, investigate visualizing loss landscapes. Our work will reemploy PCA to reduce the dimensionality of a problem, but also in parallel focuses on visualization perspective.

Methodology

We employ three main techniques to visualize the loss landscape, each of which address the complexity of different components: the features in the data and the parameters of the model. Finally, we use the popular tool Unity to visualize our landscapes in an artistic way.

1. Reduce the dimension of features and visualize

PCA is known as a powerful linear technique to find the most prominent features from a multidimensional dataset. Specifically, PCA finds the axes of highest variance in a dataset, enabling us to reduce the dimensionality of the data. The features of the dataset is naturally the first point that we want to address to reduce the complexity of the problem. Theoretically, in a linear model, the dimensionality of the features is the same as the dimensionality of the parameters; and so by reducing the dimensionality of the features, we also achieve the reduction of the parameters space accordingly.

We pick the simple IRIS dataset which has four features and use PCA to reduce it to two features. In this way, we can visualize the results, since the third dimension is loss values for various combinations of the two features. The result of this PCA step is shown in Figure 1.

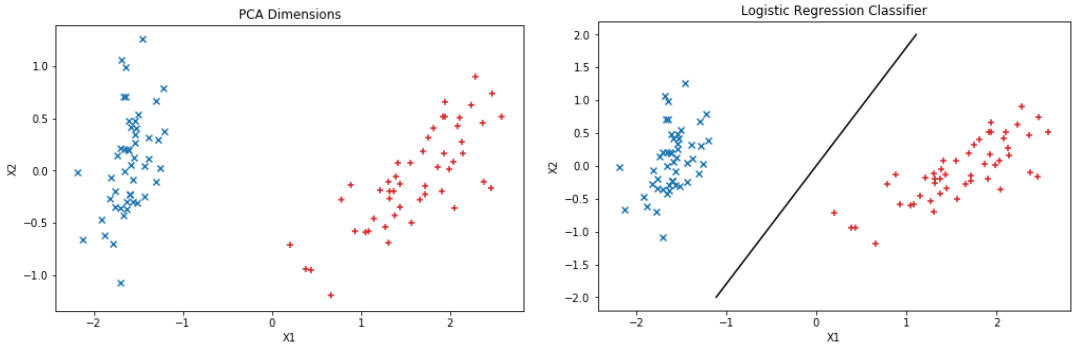


Figure 1. IRIS dataset in 2D after PCA pre-processing step (left) and the classification solution (right) on this dataset.

Since this problem boils down to a simple one, we run the logistic regression easily and are able to find it quickly as shown in Figure 1. Next, to draw the loss surface, we densely sample the points around the optimal solution θ* and calculate the loss one more time for all those new points. With this technique, we come up with a contour plot and the loss surface as shown in Figure 2.

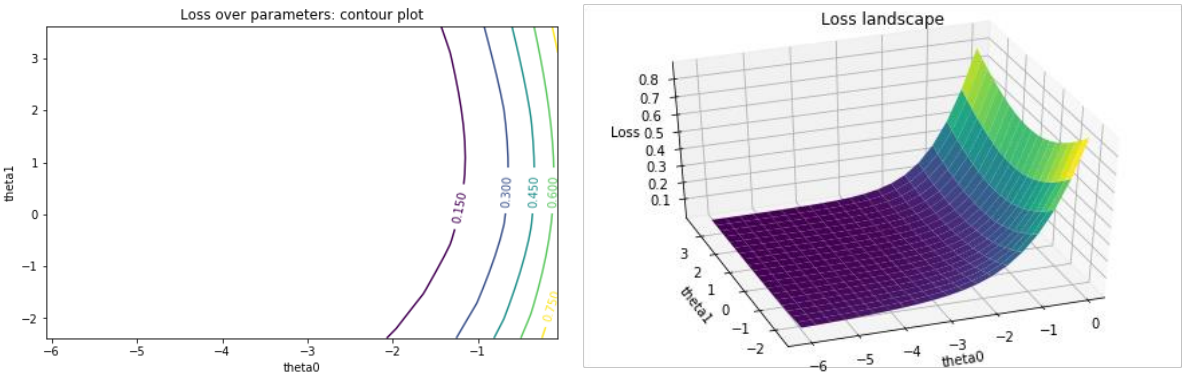


Figure 2. Contour plot (left) and the loss surface (right) on the IRIS dataset

To visualize the learning progress, we randomly initialize the start point on this surface, and calculate the loss value of a limited number of steps (which we set to 500) and visualize this whole progress in a single plot, as shown in Figure 3.

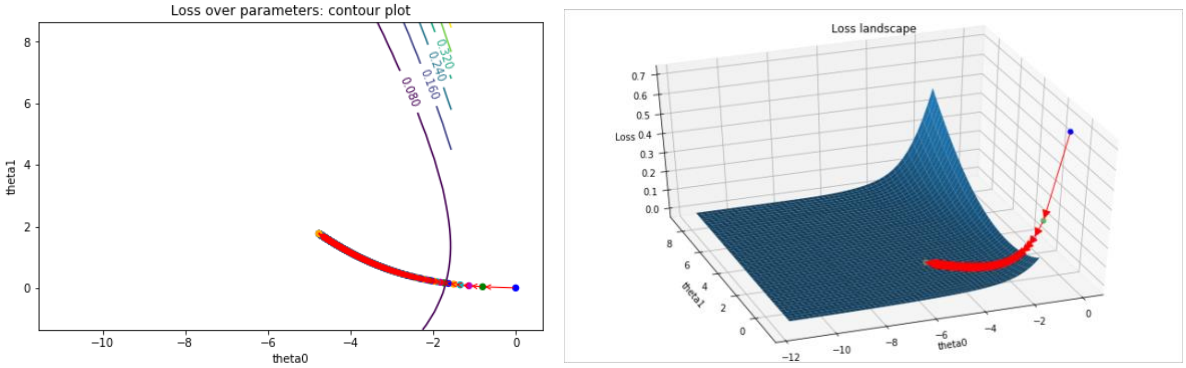


Figure 3. Stochastic Gradient Descent (SGD) training process for IRIS dataset

However, since there is a time dimension involved in the training progress, and given the limitation of a static image for visualization, we resort to Unity to visualize the learning as a set of videos, which will be illustrated in the **Result** section below.

2. Visualize loss surface if the parameter space is still high

If PCA cannot reduce much of the dimensionality, we only select two dimensions out of the parameter space to visualize. To make sure we choose the best features (we consider each parameter represent each feature), we record the learning of all features and select the most prominent ones that change the most during the training.

Similar to what has been done before, we still densely sample the points around those chosen features while keeping the others constant, calculate the loss to visualize the surface, and draw the gradient moves over this surface. The results for IRIS (4 dimensions) is shown in Figure 4 and MNIST (784 dimensions) in Figure 5.

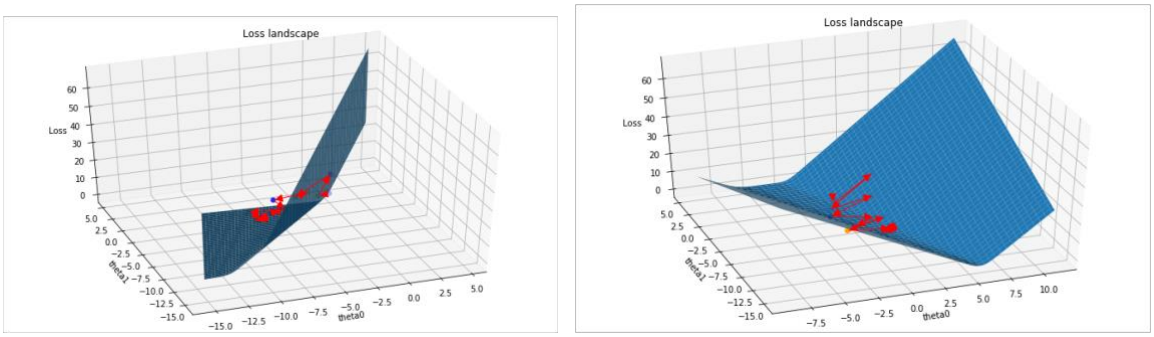


Figure 4: Loss surface and learning progress for IRIS of 2 pairs of parameters

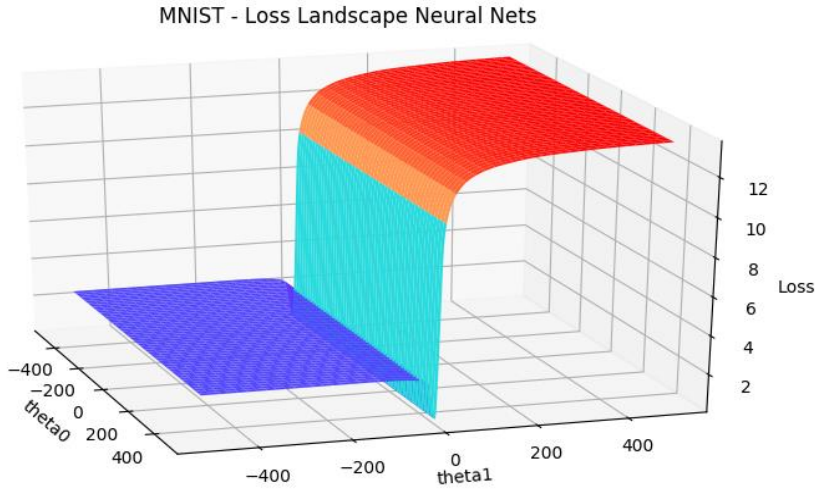


Figure 5: Loss surface of MNIST classification loss surface with 2 most prominent features

3. Reduce the dimensionality of parameters and visualize

In addition to using PCA to reduce the dimensionality of the features, we also experimented with PCA to reduce the dimensionality of the parameters of the model. We hypothesized that this would be a more direct way to reduce the size of the model and give us more intuitive 3D loss landscapes.

The main challenge for this approach was that the loss functions were defined in a higher dimension (the same number of dimensions as the model's original parameter space), so if we reduced the parameter dimensions, we could not evaluate the loss for those reduced parameters any more. To circumvent this issue, we devised an algorithm called PCA Loss Landscape, which does the following:

- Compute parameters $\theta^{(1)}...\theta^{(t)}$ for t epochs of gradient descent
- Use PCA to reduce $\theta^{(1)}...\theta^{(t)}$ parameters to two dimensions of highest variance during gradient descent
- Sample a grid of points around each parameter $\theta^{(1)}...\theta^{(t)}$. Project sampled points back to the original parameter space through an inverse PCA.
- Evaluate loss for (projected) sampled points. Plot loss for sampled points against the reduced 2D parameters.

The key idea behind this algorithm is to treat the model parameters themselves as data. We can obtain more data points of the parameters as we optimize them over iterations of gradient descent. We can then apply PCA to reduce this parameter dataset to two dimensions. By densely sampling a grid of points around the reduced two-dimensional parameters, and then projecting those sampled points back to the original parameter space through an inverse PCA, we can get the reduced parameters to a higher dimension where we can evaluate the loss. Finally, we can plot the loss for the (projected) 2D points against the two reduced parameters to form a 3D loss landscape.

We used this approach on two models, logistic regression and SVM, and two datasets, IRIS and the Wisconsin breast cancer dataset. Figures 6 and 7 show the grid of sampled points around the reduced 2D parameters and the resulting 3D loss landscape by projecting those points and evaluating their loss.

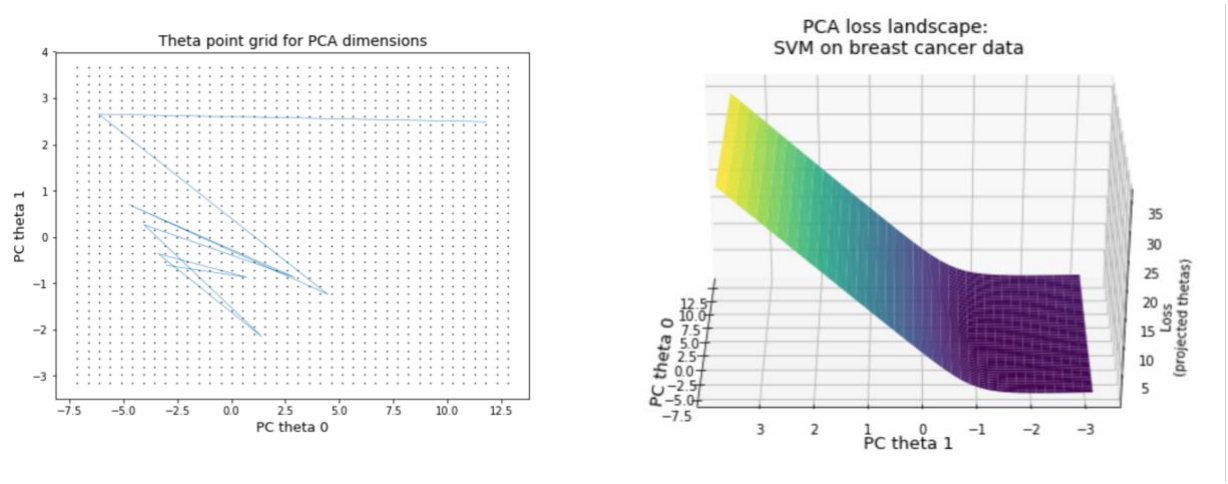


Figure 6: For SVM trained on the Wisconsin breast cancer dataset, we reduce 30 parameters to 2 parameters. The grid of sampled points in black and iterations of gradient descent in blue (left). The resulting 3D loss landscape (right).

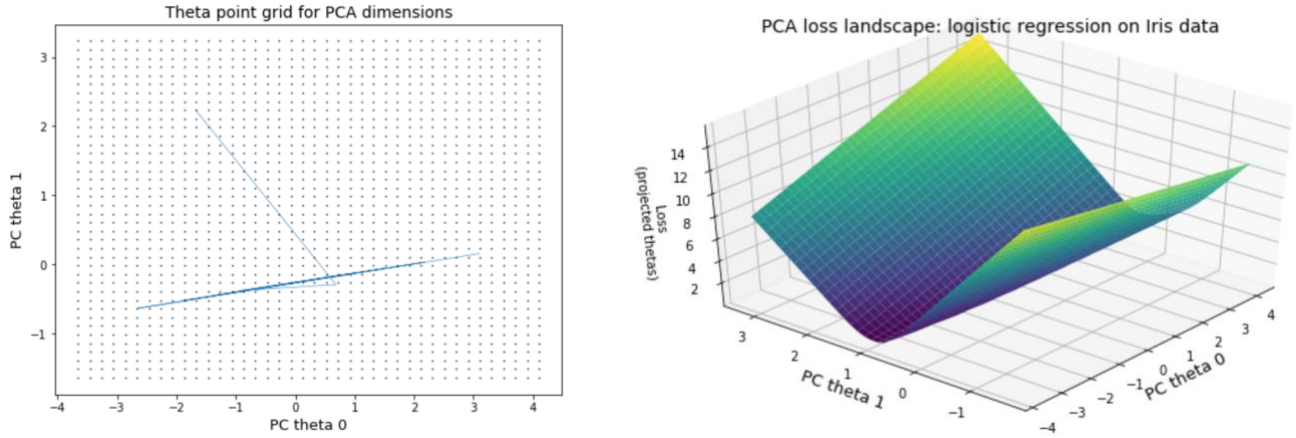


Figure 7: For logistic regression trained on IRIS, we reduce 4 parameters to 2 parameters. The grid of sampled points in black and iterations of gradient descent in blue (left). The resulting 3D loss landscape (right).

4. Visualization with Unity

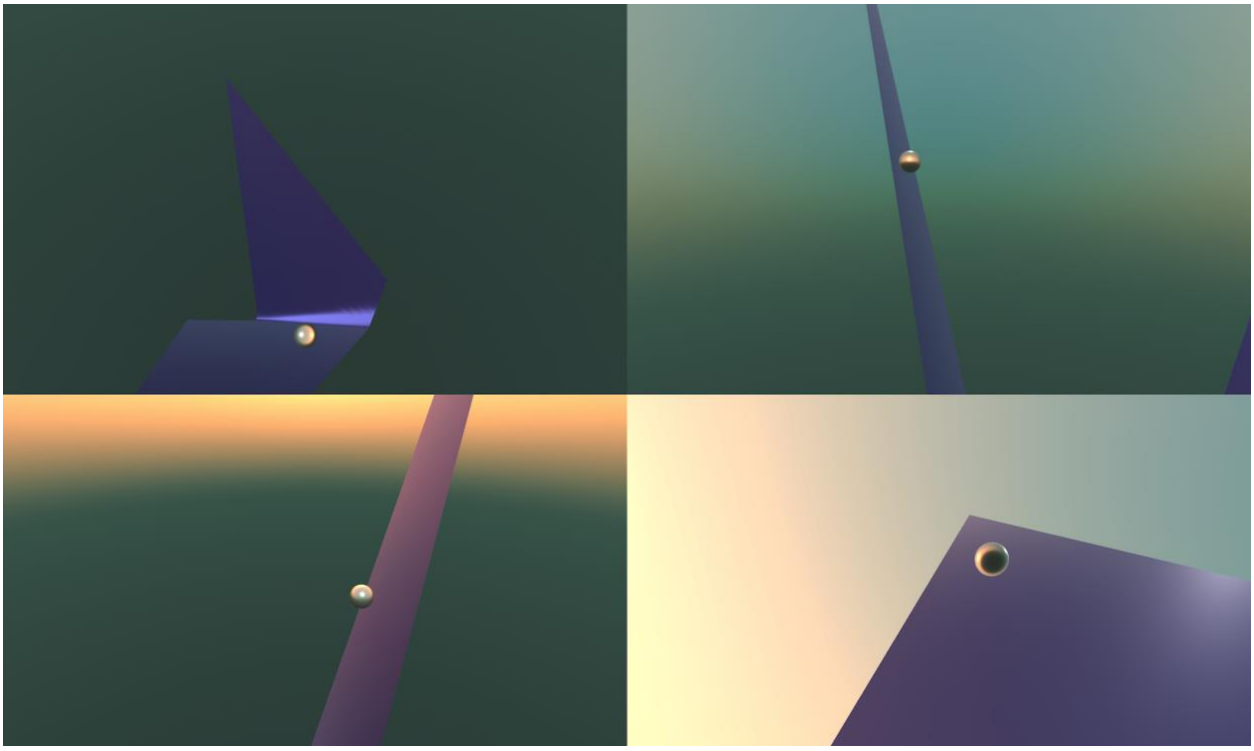
We exported all the loss landscape points and gradient descent points to point cloud files and used Unity to draw the loss landscapes.

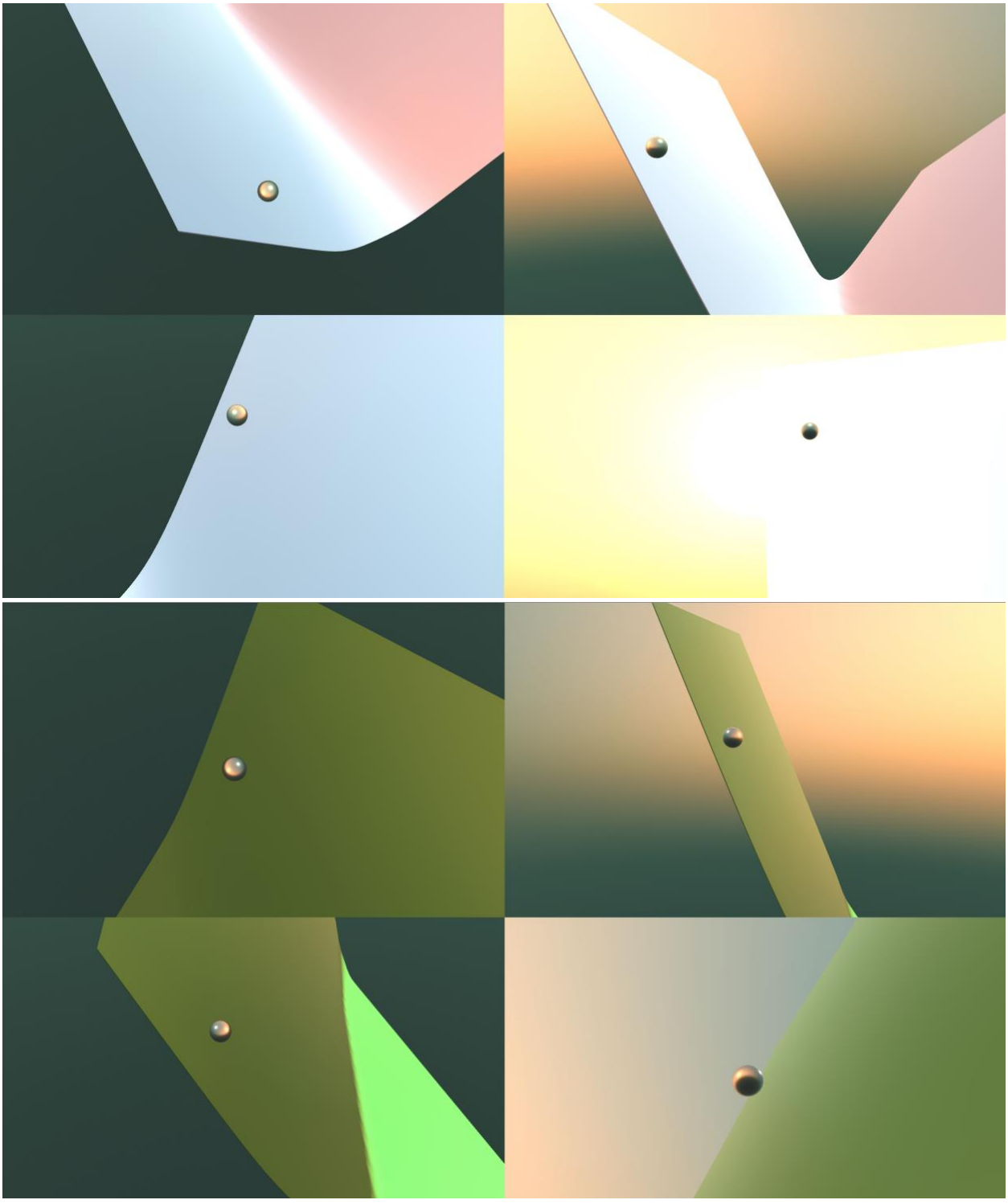
We also made the gradient descent visualization as episodes in which each one captures the motion of the optimizer in two particular dimensions. We demonstrate this

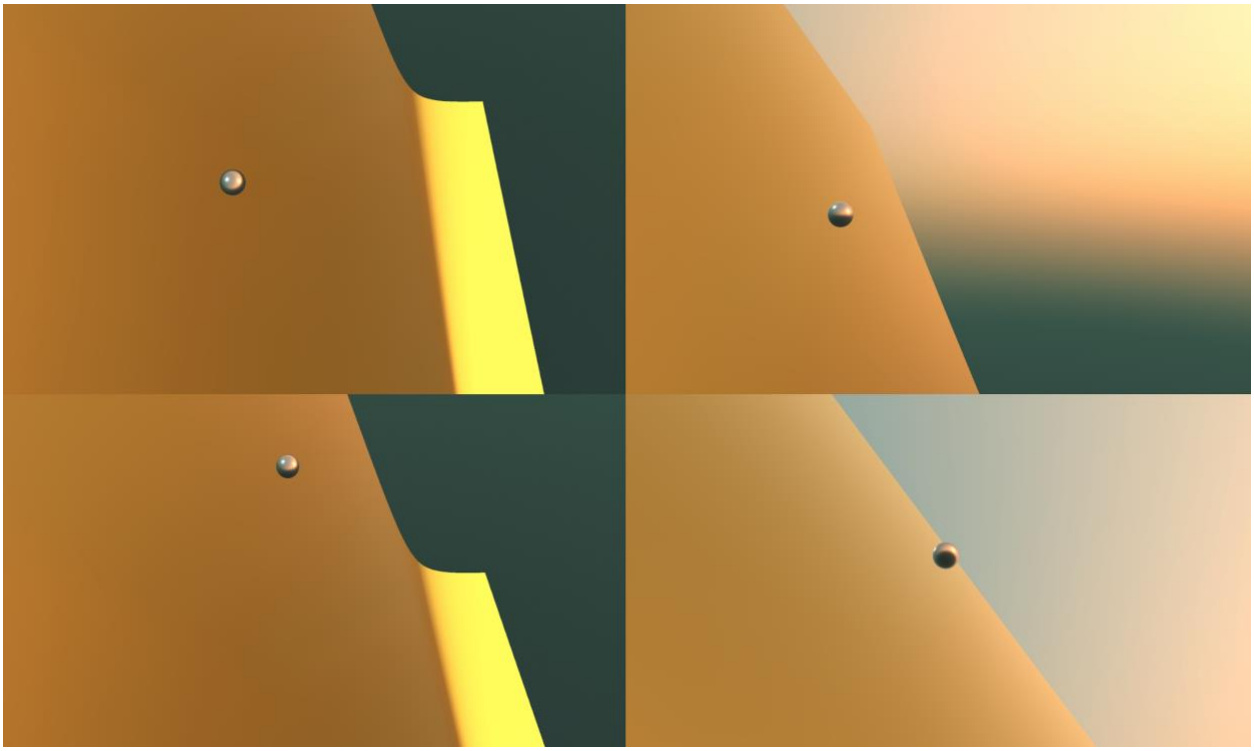
by adding a ball that rolls down the loss landscape following the path of gradient descent down the landscape. Those results are presented in the **Result** section.

Results

We deconstructed the high dimensional spaces to a series of metaphorical three-dimensional spaces where a little ball that keeps looking for the lowest point in the loss landscapes with gravity. When the little ball finds the lowest point in one loss landscape, it switches to another loss landscape which constructed by another three dimensions in the high dimensional landscape and keeps dropping to the lowest point. It gives us, human beings, an intuitive idea of how loss functions works in the high dimensional spaces of machine learning algorithms.







Reflection

From a technical standpoint, we were quite surprised by the results of our experiments. We expected that our first approach of reducing the dimensionality of the features would create intuitive, convex loss landscapes (since the loss functions themselves were convex), and our results confirmed our hypothesis. But we had no reason to assume that our third approach of reducing the dimensionality of the model parameters would also create intuitive and convex loss landscapes. Our somewhat novel approach of applying PCA on the parameter space let us visualize the loss landscapes of complex models like the 30-parameter SVM trained on the Wisconsin breast cancer dataset and the 4-parameter logistic regression trained on IRIS. This is an accomplishment that may have major implications for understanding how these complex models learn from data.

We also had some accomplishments from the artistic point of view. Our 3D visualizations of the loss landscapes helped simplify and explain a complicated topic in optimization. The ball rolling down the loss landscape helped demonstrate the concept of gradient descent. These visualizations serve as an artistic representation of machine learning models and show, over the passage of time (or epochs), how these models learn by finding local optima.

There is a plenty of room for future work on this project. First, we hope to make more realistic visualizations of the ball rolling down the loss landscape by adding shadows

and physics-based effects. Also, we would like to have better transitions from one landscape to another. Further, we hope to extend our PCA loss landscape algorithm to more complex models like neural networks and non-linear models. A good place to start would be by using non-linear kernels like RBF for the logistic regression and SVM models and trying to create 3D loss landscapes for these models. Finally, it would be beneficial to try non-linear dimensionality reduction methods like local-linear embedding and autoencoders, which would be useful if the model parameters are not linear.

We found working on this project to be a rewarding and challenging experience. We learned a lot about optimization and dimensionality reduction and applied those concepts by creating a meaningful work of art. Our work can be used as an educational tool to explain how machine learning models learn or simply as an artistic representation of these high-dimensional models. It was a great learning experience to work on a team with various skill sets (machine learning, math, statistics, art/design), and our unique strengths are reflected in our final work.

Reference

[1] Li, H., Xu, Z., Taylor, G. and Goldstein, T., 2017. Visualizing the Loss Landscape of Neural Nets. arXiv preprint arXiv:1712.09913.

[2] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. ICLR, 2015.

[3] Olafur Eliasson, and Paul Hester. "Photographs - Glasstire." Olafur Eliasson: Photographs - Glasstire. 2004. Accessed 1997.

[4] Lorch, E. Visualizing Deep Network Training Trajectories with PCA. IMCL 2016

Appendix

Code

The code for all of our experiments is available at github.com/shouvikmani/art-ml-final. There are a few key files:

- experiments_w_gradient: Reducing the dimensions of the features and visualizing the loss landscape (and gradient) for logistic regression on IRIS data
- sandbox/mnist.py: Visualizing the loss landscape using the two most highly-changing parameters for a neural network trained on MNIST

- pca_loss_landscape.ipynb: Reducing the dimensions of the parameters of a model and visualizing the loss landscape for logistic regression and SVM on IRIS data and Wisconsin breast cancer dataset

Video
We will send the link to the final video via email.