CS 329M: Topics in Artificial Intelligence **Spring 2017**

Lecture 8: May 22

Lecturer: James Zou *Scriber: Allen Nie, George Pakapol Supaniratisai, Lan Huong*

8.1 Recap + Onto Robust Machine Learning

For most settings in machine learning, we assume that our test data would come from the same distribution as the training data. It is possible, however, that our test data is drawn from a different distribution. For example, in the paper discussed last week, we discussed how to address the datasets with imbalanced labels.

In this lecture, we will talk about another kind of robustness: **adversarial robustness**. As you may have learned, adversarial attacks has gained attention through neural network settings for image classification, where we can add only some noise (visually undetectable) to trick the classifier into an incorrect label with high confidence. However, such idea is not specific to neural networks, but also a simple linear models. We will look at some theories on robustness of linear regression, then we move onto neural networks.

8.2 Robust Linear Regression

In a standard linear regression problems, we have an input matrix $X \in \mathbb{R}^{n \times m}$, with output $Y \in \mathbb{R}^n$. We seek to find the (unknown) parameter β that minimizes the objective:

$$\min_{\beta \in \mathbb{R}^m} ||Y - X\beta||_2$$

Sometimes, our data matrix X would naturally have some noises, and we are interested in how well our model tolerate these perturbations. Suppose that in a test setting, our data set has been perturbed by $X' = X + U$, where $U \in \mathcal{U}$, a set of possible perturbations we are interested in. (You may think of an adversary that adds perturbation to each row of X in the same way.)

The goal of robust least square (Robust LS) is to solve

$$\min_{\beta \in \mathbb{R}^m} \max_{U \in \mathcal{U}} ||Y - (X + U)\beta)||_2$$

That is, find the parameter β that **minimizes the worst possible loss** calculated on any perturbed X'.

This appears to be difficult to solve. However, it happens so that, for some interesting set of perturbation $\mathcal{U}$, this problem exactly corresponds to solving **linear least squares with L_1 or L_2 regularization**. We will prove this later in this note.

We will consider the interesting sets $\mathcal{U}$. It is important to note that $\mathcal{U}$ is not just any arbitrary set. We cannot bound the loss when the perturbation can be arbitrarily large. In most realistic data, there are also constraints on $\mathcal{U}$, which, to our advantage, limits the size of the possible perturbations. Below are some possible constraints we may use to limit the size of perturbation.

8.2.1 Some possible Constraints on the perturbation set $\mathcal{U}$

(1) **Constraint on Frobenius norm** $\mathcal{U}_F = \{U = (u_1, ..., u_m) : ||U||_F \leq \lambda\}$, where each u_i is a column vector in $\mathbb{R}^n$. Note that $||U||_F = \sqrt{\sum_{i,j} U_{ij}^2} = \sqrt{\sum_j \sigma_j^2(U)}$.

Theorem 1 *Solving the following problem*

$$\min_{\beta} \max_{U \in \mathcal{U}_F} ||Y - (X+U)\beta||_2$$

is equivalent to the solution of the linear least square problem with L_2 norm penalty.

$$\min_{\beta} ||Y - X\beta||_2 + \lambda||\beta||_2$$

Note that this is L_2 regularization, a standard technique to prevent overfitting.

Proof: Let $T_1(\beta) = \max_{U \in \mathcal{U}_F} ||Y - (X+U)\beta||_2$, $T_2(\beta) = ||Y - X\beta||_2 + \lambda||\beta||_2$. By triangle inequality,

$$||Y - (X+U)\beta||_2 \leq ||Y - X\beta||_2 + ||U\beta||_2$$

Furthurmore,

$$||U||_F \geq ||U||_2 = \sup_{\beta} \frac{||U\beta||_2}{||\beta||_2}$$

As such, for any β, $||U\beta||_2 \leq ||U||_F||\beta||_2$. Substitute in the first inequality to get

$$||Y - (X+U)\beta||_2 \leq ||Y - X\beta||_2 + ||U||_F||\beta||_2 \leq ||Y - X\beta||_2 + \lambda||\beta||_2$$

$\Rightarrow T_1(\beta) \leq T_2(\beta)$.

To prove another direction, for a given β, it is enough to choose just one U s.t. $T_2(\beta) \leq ||Y - (X+U)\beta||_2$. To do so, we need to maximize the right side, which we expand as

$$||Y - X\beta - U\beta||_2$$

The idea is, we pick‡ U so that $-U\beta$ and $Y - X\beta$ points in the same direction (which means that perturbing X in a very specific direction). Since they are parallel, we rewrite the right hand side as $||Y - X\beta - U\beta||_2 = ||Y - X\beta||_2 + ||U\beta||_2 = ||Y - X\beta||_2 + \lambda||\beta||_2 = T_1(\beta)$

This concludes the proof that $T_1(\beta) = T_2(\beta)$. ■

‡ More Specifically, we can pick

$$U = -\lambda \cdot \frac{(Y - X\beta)\beta^\top}{||Y - X\beta||_2||\beta||_2}$$

so that $||U||_F = \lambda$, we substitute U to get

$$Y - X\beta - U\beta = \left(1 + \lambda \cdot \frac{||\beta||_2}{||Y - X\beta||_2}\right)(Y - X\beta)$$

$$\Rightarrow ||Y - X\beta - U\beta||_2 = ||Y - X\beta||_2 + \lambda||\beta||_2$$

(2) **Constraint on spectral norm** $\mathcal{U}_2 = \{U = (u_1, ..., u_m) : ||u_i||_2 \leq C_i, \forall i = 1, ..., m\}$

Theorem 2 *The solution of*

$$\min_{\beta} \max_{u \in U_2} ||Y - (X + U)\beta||_2$$

is equivalent to solving linear least square problem with weighted L_1 *norm penalty*

$$\min_{\beta} ||Y - X\beta||_2 + \sum_{i=1}^{m} C_i|\beta_i|$$

Remember that each weight C_i determines how much you are allowed to perturb in the dimension (column) i in the dataset. The corresponding result is a generalized LASSO, where we have a different cost weights C_i for each of the parameters. (LASSO is the special case where C_i are all the same.)

(3) **Arbitrary norm:** $\mathcal{U} = \{U = (u_1, ..., u_m) : \left|\left| \; ||u_1||_a, \; ||u_2||_a, \; ..., \; ||u_m||_a \; \right|\right|_s \leq l\}$

We can define any norm a on each column of matrix U, and then use s to indicate the norm type of the entire result.

Theorem 3 *Solving the robust least square problem for this perturbation set is equivalent to solving*

$$\min_{\beta} ||Y - X\beta||_a + l||\beta||_s^*$$

where $||\cdot||_s^*$ *is the dual norm of* $||\cdot||_s$.

Note that $||\cdot||_p$ and $||\cdot||_q$ are dual norms if $\frac{1}{p} + \frac{1}{q} = 1$. For example, $||\cdot||_2 \leftrightarrow ||\cdot||_2$ is a dual norm of itself, and $||\cdot||_\infty \leftrightarrow ||\cdot||_1$ are dual norms or each other.

8.3 The case for Neural network

8.3.1 Generation of adversarial examples

When our systems are more complex, as in neural networks, our theorems mentioned earlier will not apply anymore. However, a canonical way to generate bad examples (that lead to very high losses) involves calculating the direction in the input space we need to move towards with the smallest step size, in order to change the final output layer as fast as possible. As such, our modification can take the following form

$$X^{\text{adv}} = X + \epsilon \cdot \text{sign}(\nabla_X \mathcal{L}(X, Y_{\text{true}}))$$

where $\mathcal{L}$ is the loss function. Likewise, to confuse the classifier to classify any input $\vec{x}$ into any class c, we can modify $\vec{x}$ as

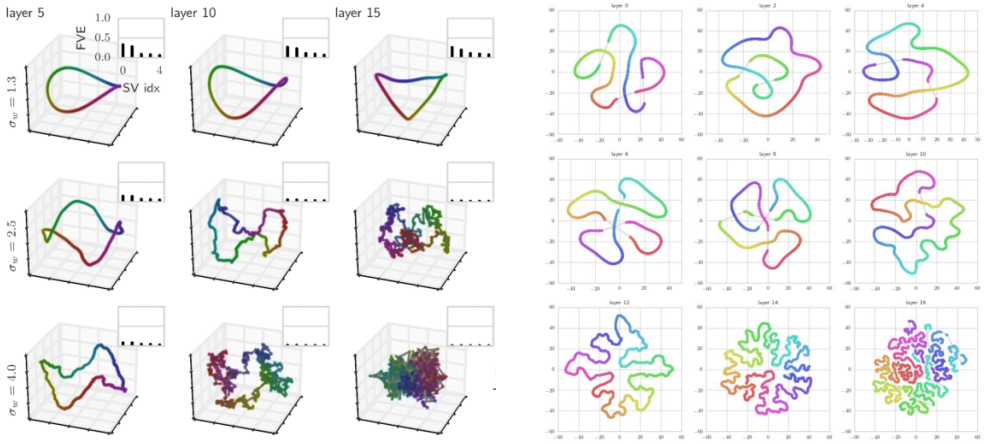
$$\vec{x}^{\text{adv}} = \vec{x} + \epsilon \cdot \text{sign}(\nabla_{\vec{x}} \text{score}(\vec{x})_c)$$

where $\text{score}(\vec{x})_c$ is the score that the classifier would assigned to class c for input $\vec{x}$.

8.3.2 Geometric transformation through neural network layers

One way to understand why it is easy to generate adversarial examples is to look at the geometric transformation of continuous curves in the input space, as input is propagated through the layers.

In Ganguli's work[1], we have a random neural network that takes inputs $\vec{x} \in \mathbb{R}^d$. (Weights are randomly initialized, and not trained on any data). We take a large sample from a set of points that form a continuous circular ring in $\mathbb{R}^d$. As each sampled points are forward-passed through the network, we can inspect the activations at each layers corresponding to these points and plot them (using t-SNE) to see how the geometry changes in each layer.



In the plot, adjacent points in the input layer have similar colors. We can see that, in deeper layers, the circular shape has been twisted into a fractal shape, similar to a space-filling curve. Such behavior indicates that the geometric transformation by neural networks is highly chaotic.

8.3.3 Distribution of distance in deeper representations for adjacent data points

Similar to the idea of geometric analysis, we can also sample from local data points and find the distribution of the distance of the activation in deeper layers. In general, we will find that the inputs that are originally close to one another can be sent far apart in the representation layers.

For example, we begin with an image input $\vec{x}$, and generate adjacent images $\vec{x}' = \vec{x} + \vec{\epsilon}$ around the input $\vec{x}$ by sampling from a spherical space, simply by adding small random gaussian noises $\vec{\epsilon}$. After that, we forward-pass the sampled data points through the network, and collect the activations in the final layer for each data points, and calculate the distance from the final-layer activation of the original data point. From the distribution of the distances, we will be able to see that, even though the data points have a small distance of $|\epsilon|$ in the first layer, they move apart in the final layer representation, with some data points being separated very far apart from the others.

Reference

[1] Ben Poole, Subhaneil Lahiri, Maithreyi Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in Neural Information Processing Systems NIPS, Deep Learning Workshop*, 2016