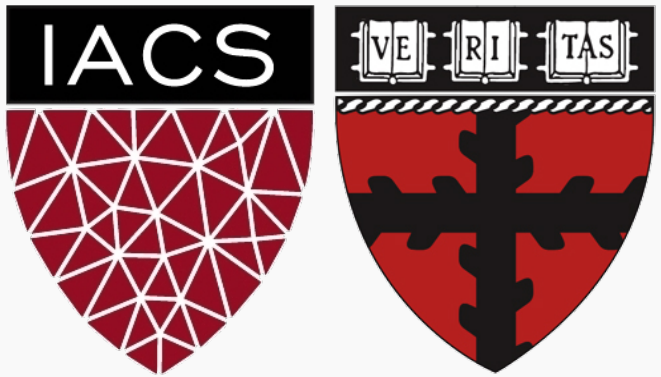
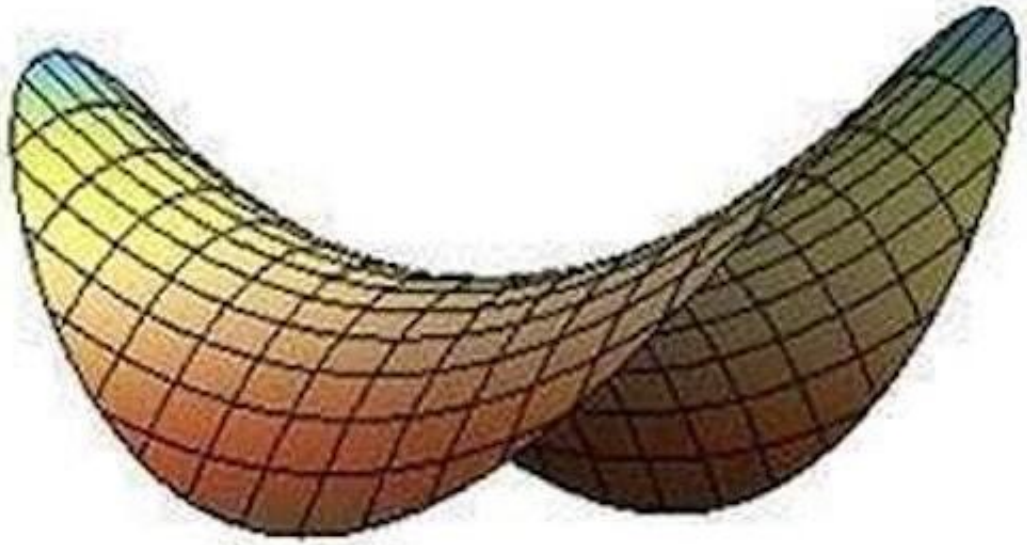
Neural Network Regularization
Norm Penalties and Early Stopping

CS109B Data Science 2

Pavlos Protopapas, Mark Glickman





$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = cz$$

Pringles are examples of hyperbolic paraboloids.

Outline

Regularization of NN

- Norm Penalties
- Early Stopping
- Data Augmentation
- Dropout

Regularization

Regularization is any modification we make to a learning algorithm that is intended to **reduce its generalization error** but not its training error.

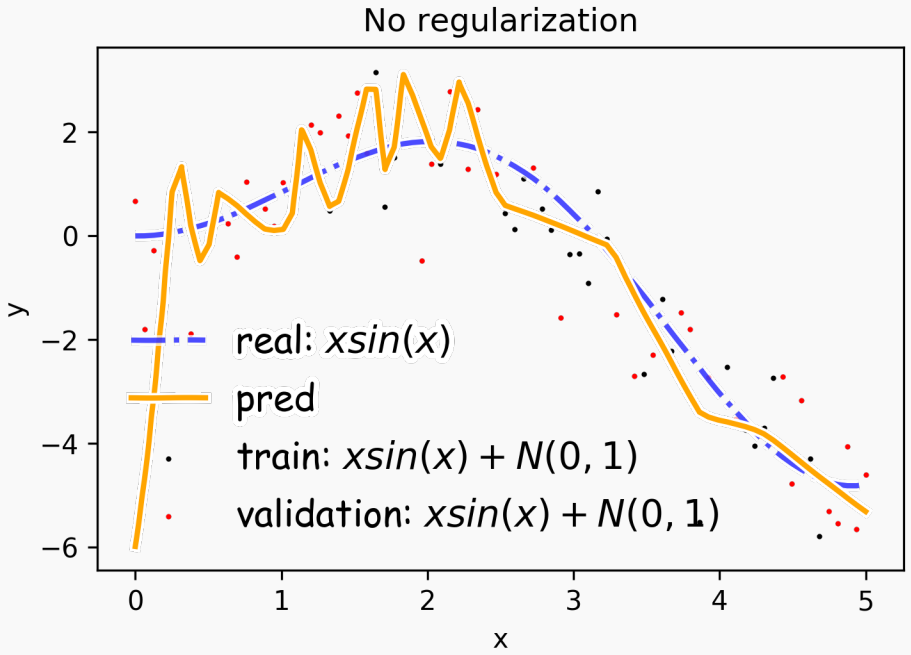
Outline

Regularization of NN

- **Norm Penalties**
- Early Stopping
- Data Augmentation
- Dropout

Overfitting

Fitting a deep neural network with 5 layers and 100 neurons per layer can lead to a very good prediction on the training set but poor prediction on validations set.



Norm Penalties

We used to optimize:

$$L(W; X, y)$$

Change to …

$$L_R(W; X, y) = L(W; X, y) + \alpha\Omega(W)$$

Biases not penalized

L_2 regularization:

- Weights decay
- MAP estimation with Gaussian prior

$$\Omega(W) = \frac{1}{2} \| W \|_2^2$$

L_1 regularization:

- encourages sparsity
- MAP estimation with Laplacian prior

$$\Omega(W) = \frac{1}{2} \| W \|_1$$

Norm Penalties

We used to optimize:

$$W^{(i+1)} = W^{(i)} - \lambda \frac{\partial L}{\partial W}$$

Change to:

$$L_R(W; X, y) = L(W; X, y) + \frac{1}{2}\alpha \parallel W \parallel_2^2$$

$$W^{(i+1)} = W^{(i)} - \lambda \frac{\partial L}{\partial W} - \lambda\alpha\, W^{(i)}$$

Weights decay in proportion to size

Biases not penalized

L_2 regularization:

- **Decay of weights**
- MAP estimation with Gaussian prior

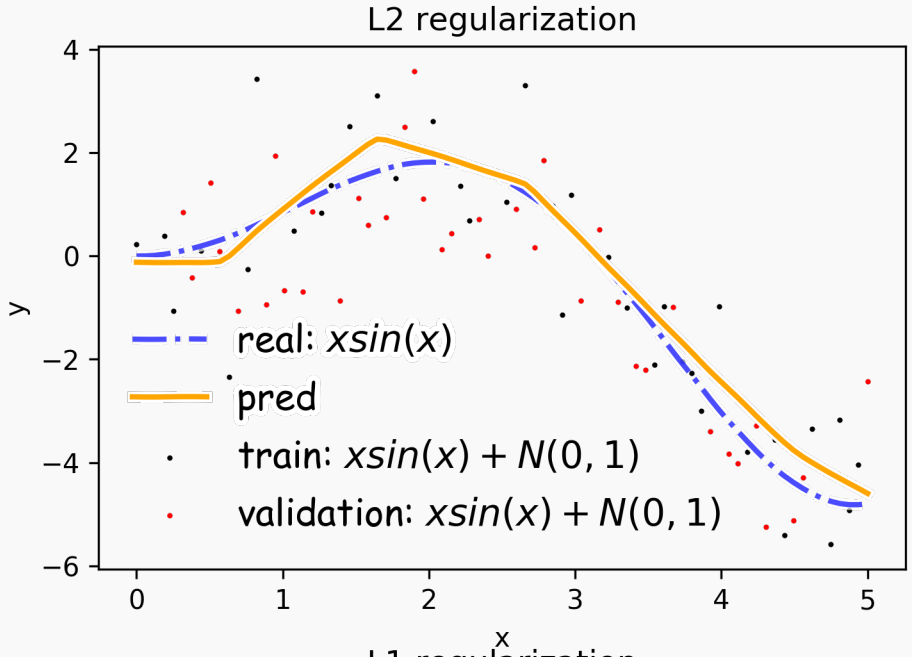
$$\Omega(W) = \frac{1}{2} \parallel W \parallel_2^2$$

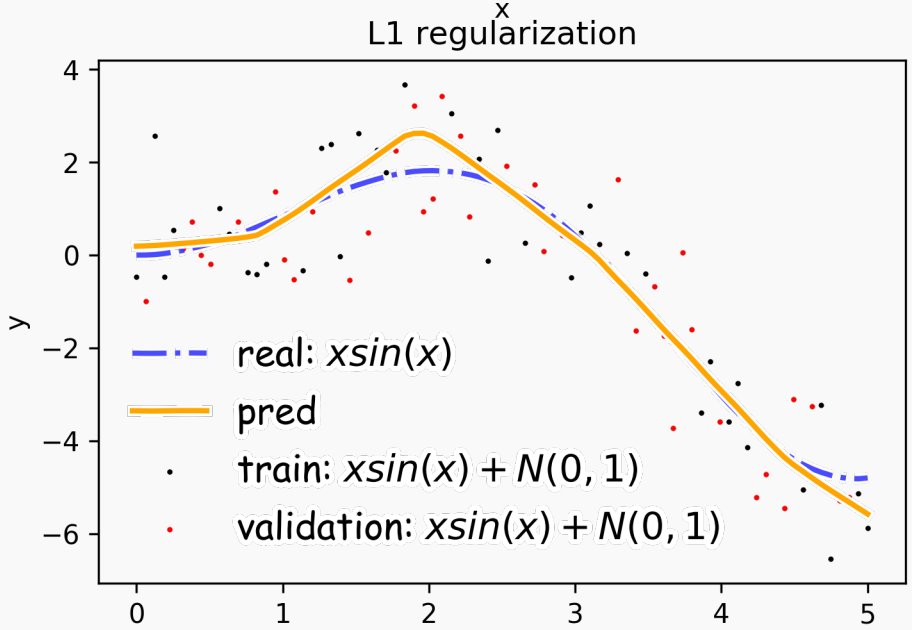
L_1 regularization:

- encourages sparsity
- MAP estimation with Laplacian prior

$$\Omega(W) = \frac{1}{2} \parallel W \parallel_1$$

Norm Penalties



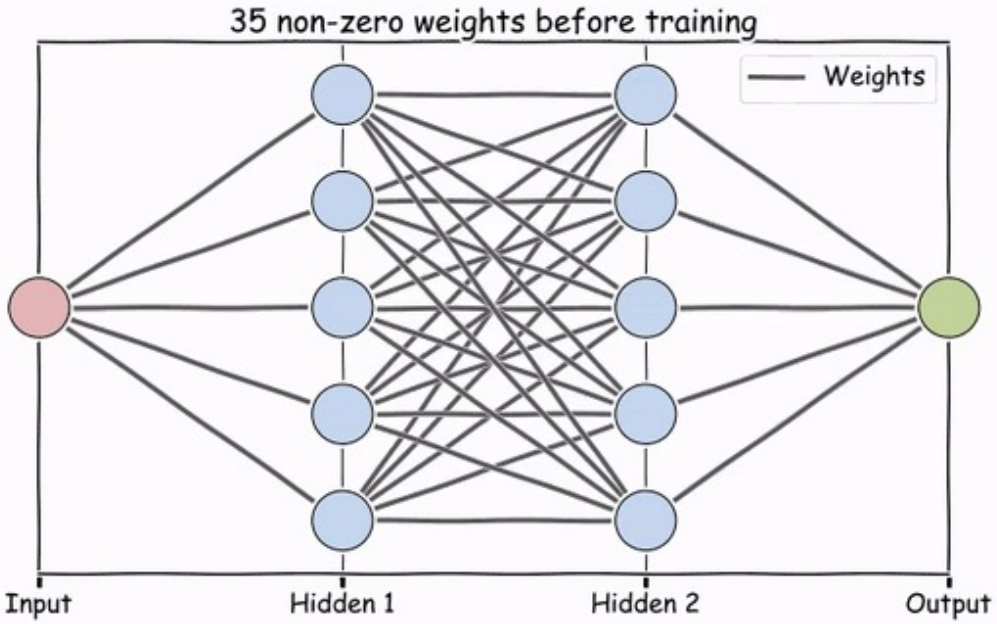


$$\Omega(W) = \frac{1}{2} \| W \|_2^2$$

$$\Omega(W) = \frac{1}{2} \| W \|_1$$

Exercise: Model pruning using Lasso regularization

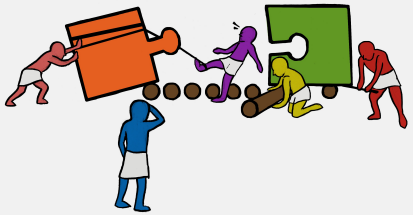
The goal of this exercise is to perform model pruning using L_1 regularization. Your final



NOTE: This graph is only a sample. The data will be the same, the model predictions will depend on your regularization parameters.

Instructions:

- Generate the predictor and response data using the helper code given.
- Split the data into train and test sets.
- Build a simple neural network with 2 hidden layers with 5 neurons each. Add an appropriate L_1 regularization to each layer.
- Compile the model with MSE as the loss.
- Fit the model on the training data and use the helper function `plot_weights()` in order to visualize the non-zero weights after a given set of epochs.
- Adjust the amount of L_1 regularization to see how quickly the network weights become zero.



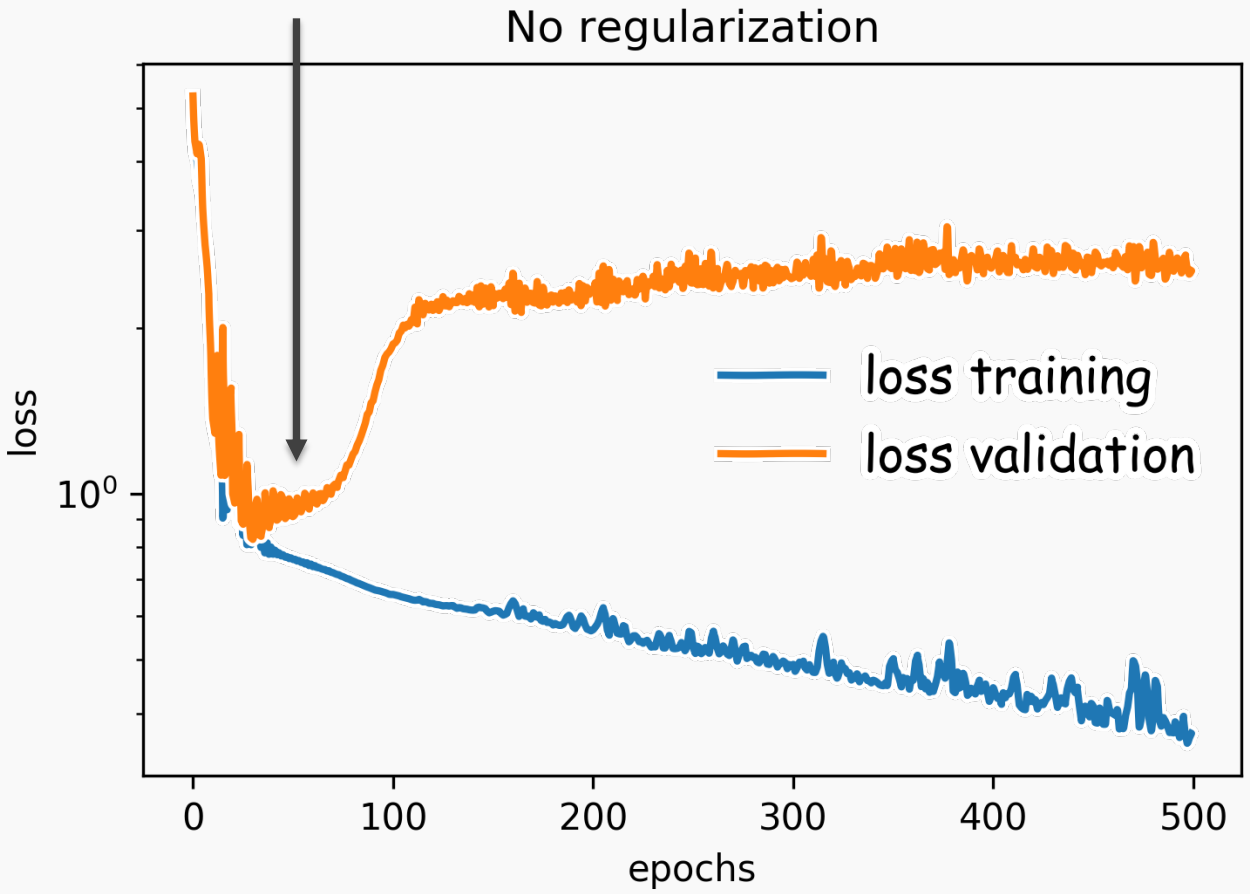
Outline

Regularization of NN

- Norm Penalties
- **Early Stopping**
- Data Augmentation
- Dropout

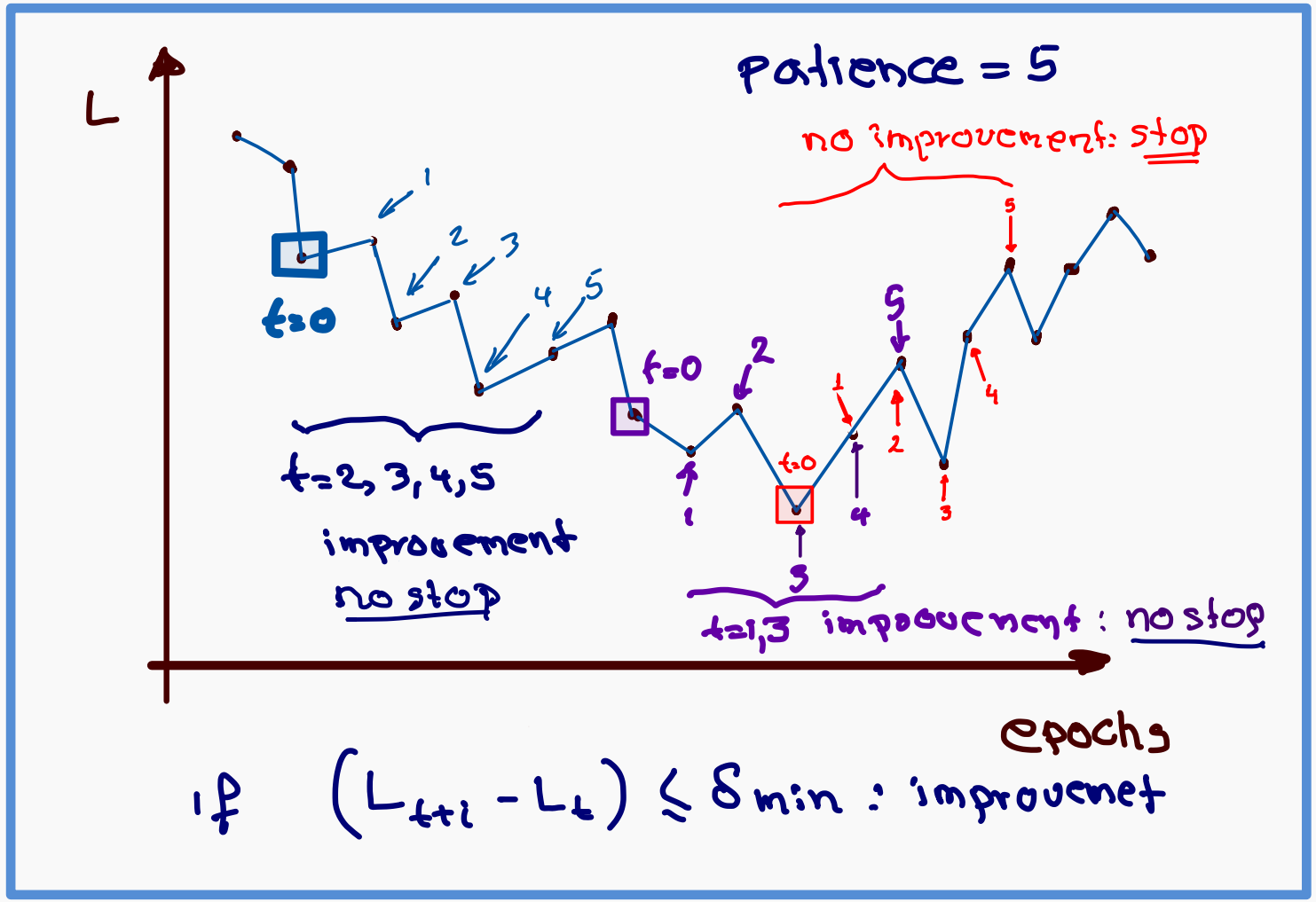
Early Stopping

Early stopping: terminate while validation set performance is better. Sometimes is worth waiting a little before stopping. This is called `patience`.



Patience is defined as the number of epochs to wait before early stop if no progress on the validation set.

The patience is often set somewhere between 10 and 100 (10 or 20 is more common), but it really depends on the dataset and network.


patience = 5
L
no improvement: stop
t=0
t=2, 3, 4, 5
improvement
no stop
t=0
t=0
t=1, 3 improvement : no stop
epochs
if $(L_{t+1} - L_t) \lesssim \delta_{min}$: improvenet

The threshold beyond which the monitored quantity is considered to have changed

Number of epochs with no improvement after which training will be stopped

The quantity monitored based on which early stopping takes place

```
tf.keras.callbacks.EarlyStopping(
    monitor='val_loss', min_delta=0, patience=0, verbose=0,
    mode='auto', baseline=None, restore_best_weights=False
)
```

Specifies whether the metric value after each epoch is displayed or not

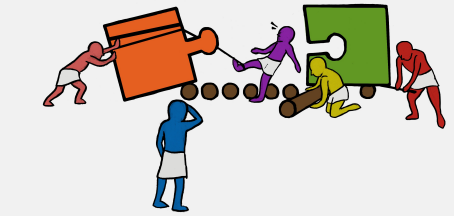
One of {"auto", "min", "max"}. In "min" mode, training will stop when the quantity monitored has stopped decreasing; in "max" mode it will stop when the quantity monitored has stopped increasing; in "auto" mode, the direction is automatically inferred from the name of the monitored quantity

The baseline value for the monitored quantity. Training will stop if the model doesn't show improvement over the baseline

Whether to restore model weights from the epoch with the best value of the monitored quantity. If False, the model weights obtained at the last step of training are used

Exercise: Early stopping

The goal of this exercise is to perform early stopping



- Build a neural network with 5 hidden layers and 100 neurons each
- Use the 'early stopping' callback in model training
- Compare the test MSE of the unregularized model with the model trained with early stopping