Regularization

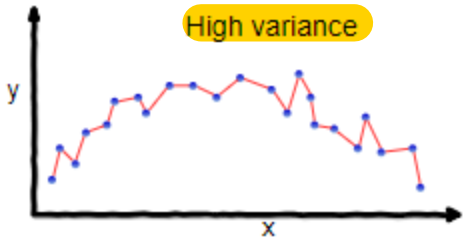
The problem of overfitting

What is a bias

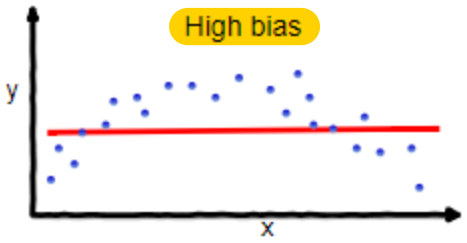
- Bias are the **simplifying assumptions** made by a model to make the target function easier to learn
- Bias is the difference between the average prediction of our model and the correct value which we are trying to predict.
- Model with **high bias** pays very little attention to the training data and **oversimplifies the model**. It always leads to high error on training and test data

What is Variance

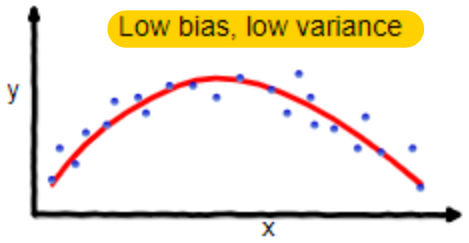
- **Variance is the variability of model prediction** for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.



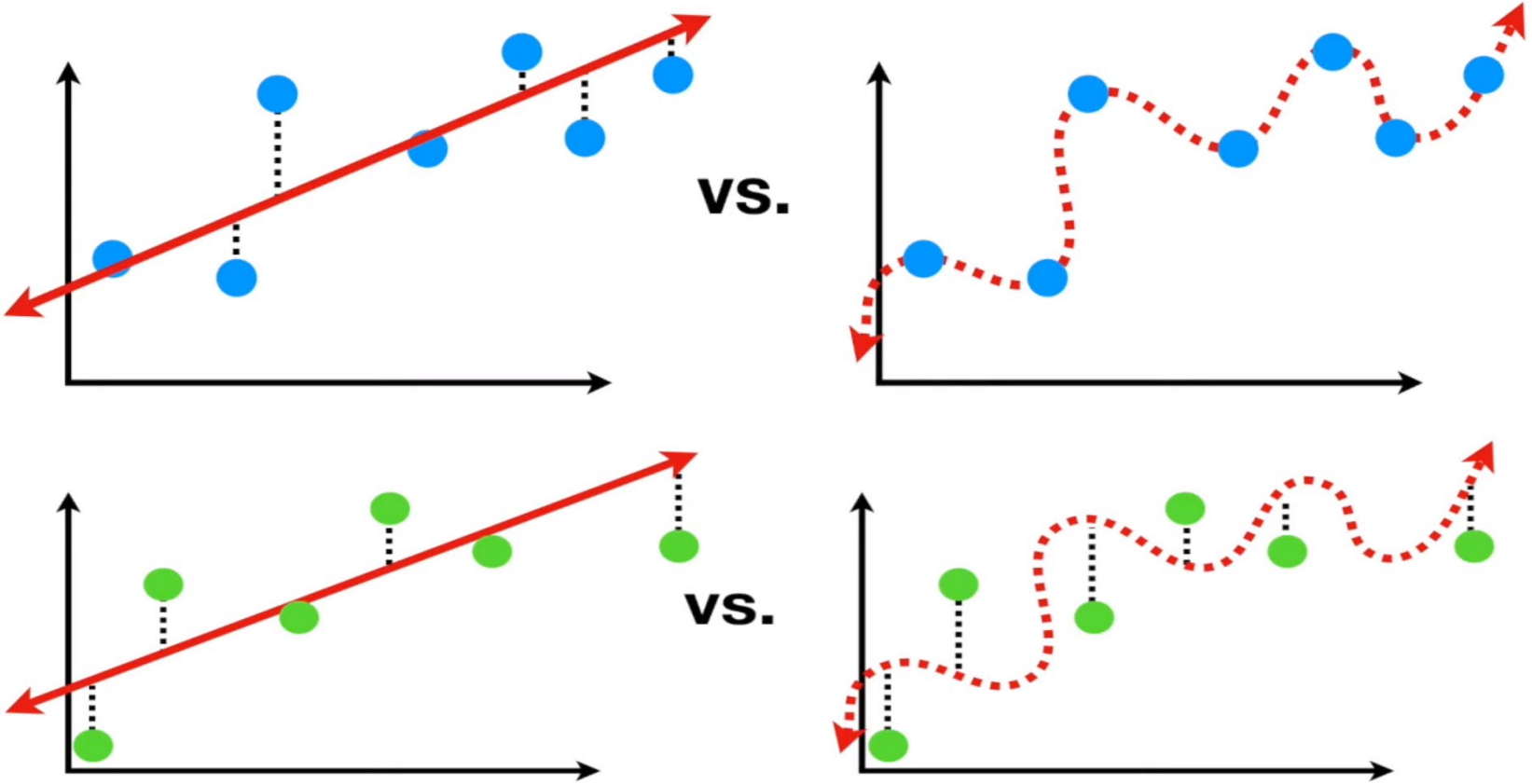
overfitting



underfitting



Good balance


vs.
vs.

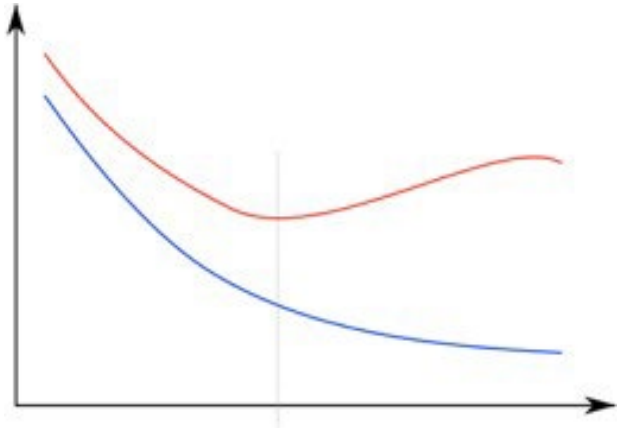
Overfitting

- Our model is very good with training data set (with memorization)
- Not good at test dataset or in real use

Overfitting/underfitting

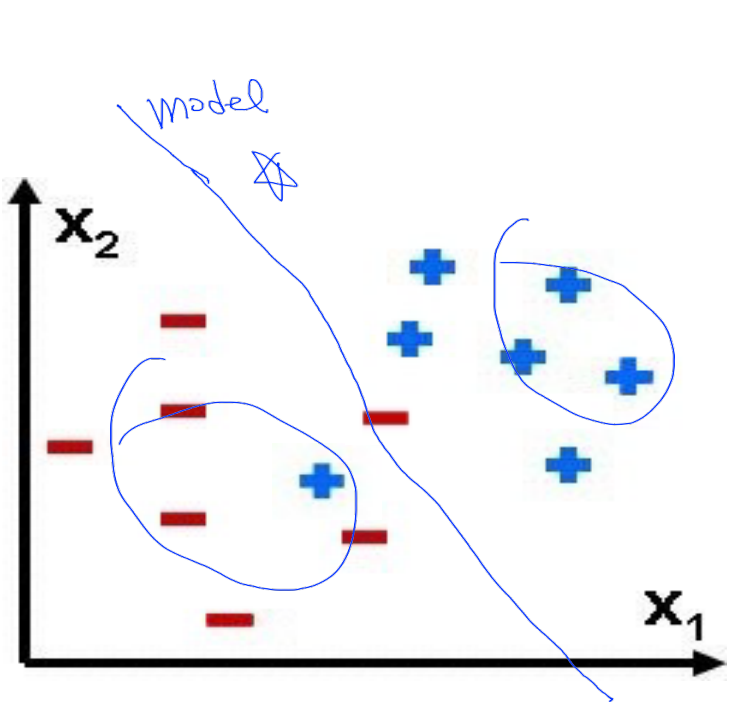
- Why is it not good?

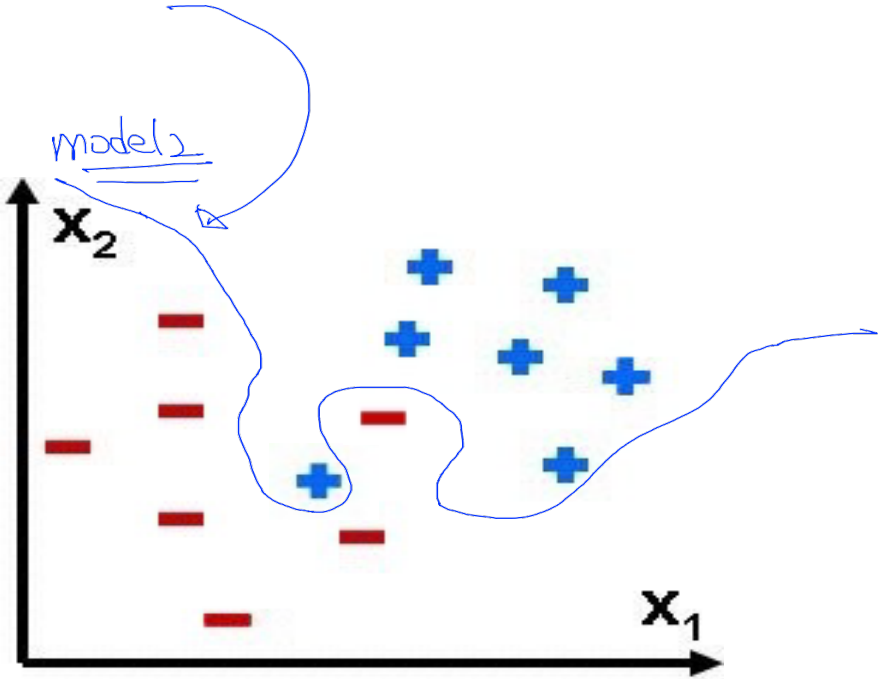
Am I overfitting?



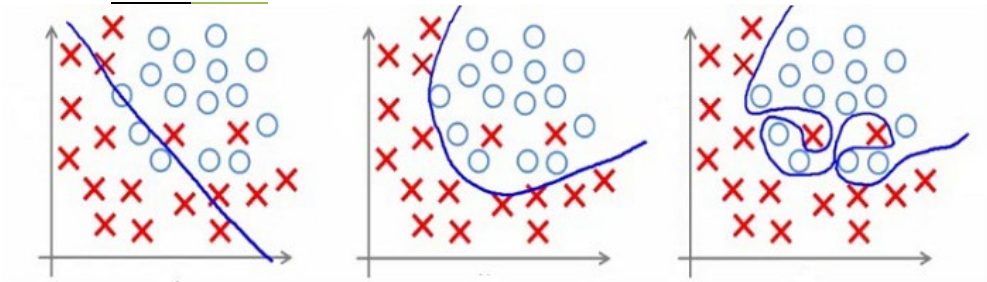
- Very high accuracy on the training dataset (eg: 0.99)
- Poor accuracy on the test data set (0.85)

Overfitting





Overfitting vs Underfitting



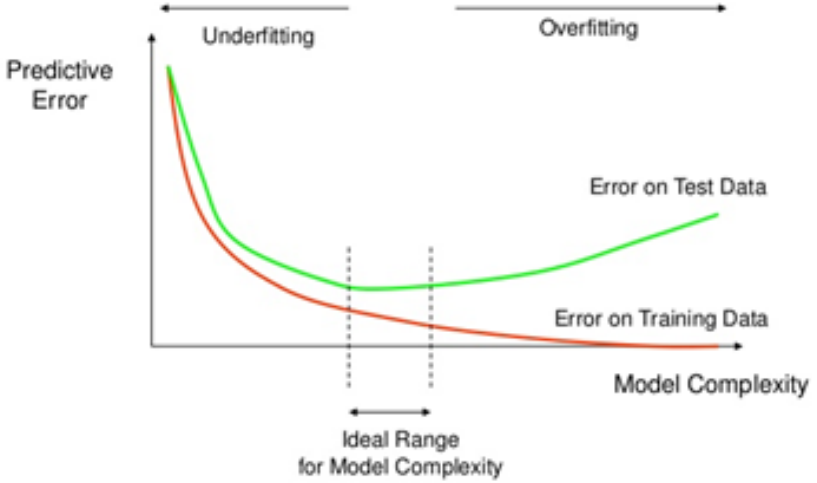
Under-fitting

(too simple to explain the variance)

Appropriate-fitting

Over-fitting

(forcefitting -- too good to be true)

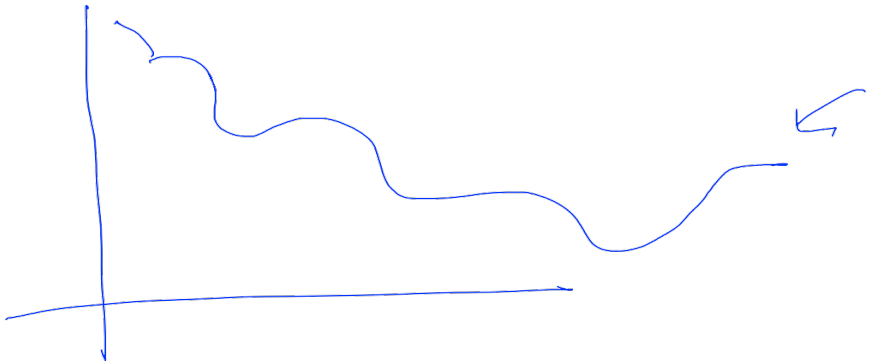


Solutions for overfitting

- More training data!
- Reduce the number of features
- **Regularization**

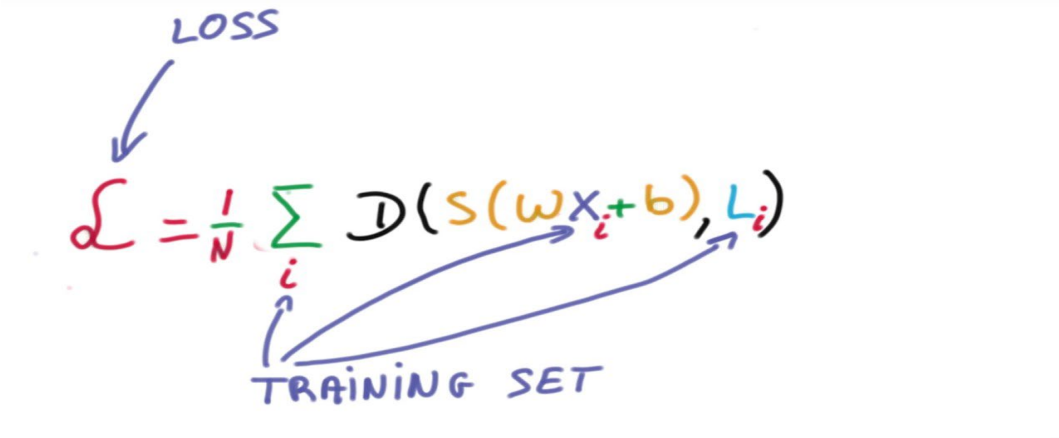
Regularization

- Let's not have too big numbers in the weight



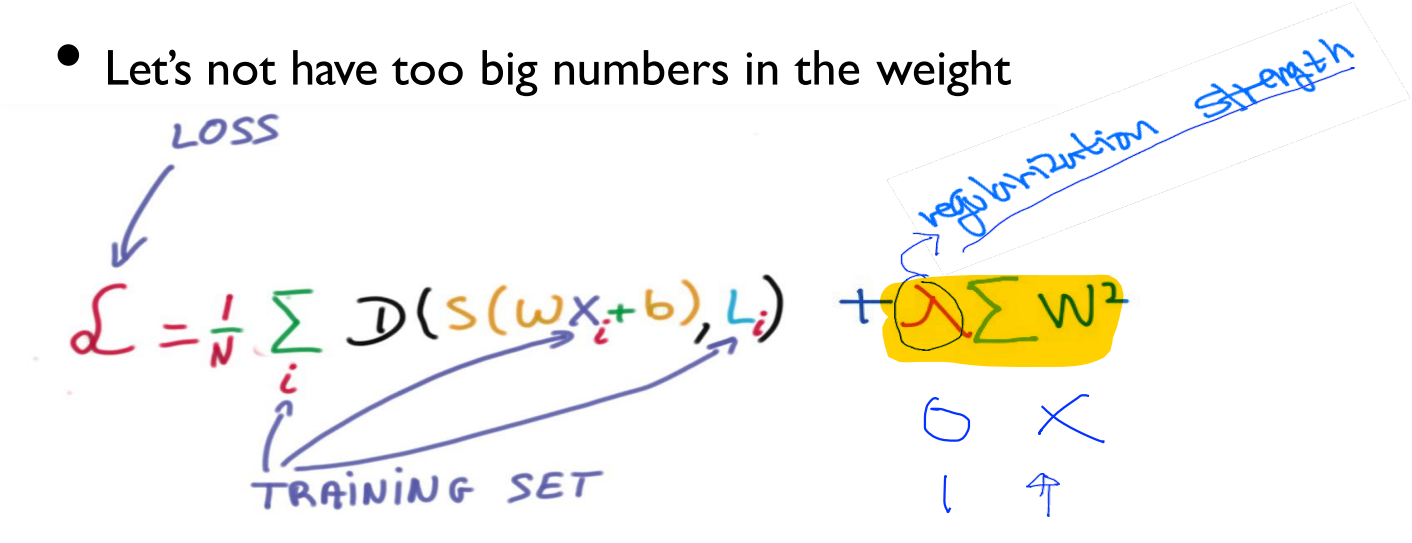
Regularization

- Let's not have too big numbers in the weight



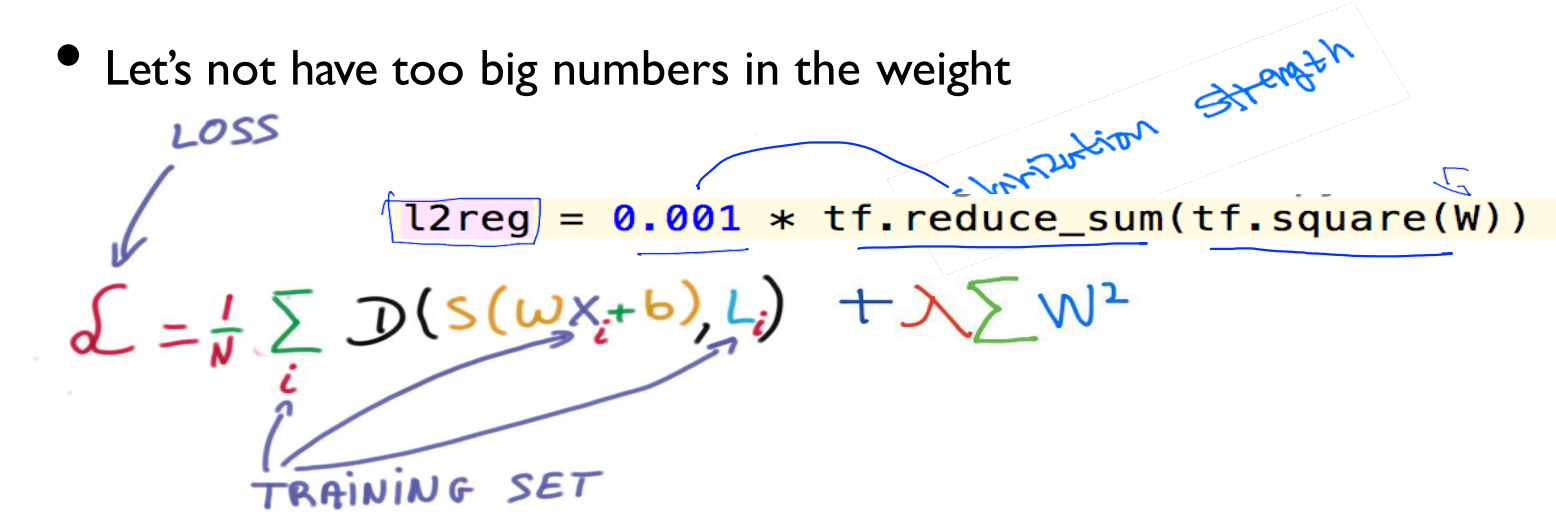
Regularization

- Let's not have too big numbers in the weight



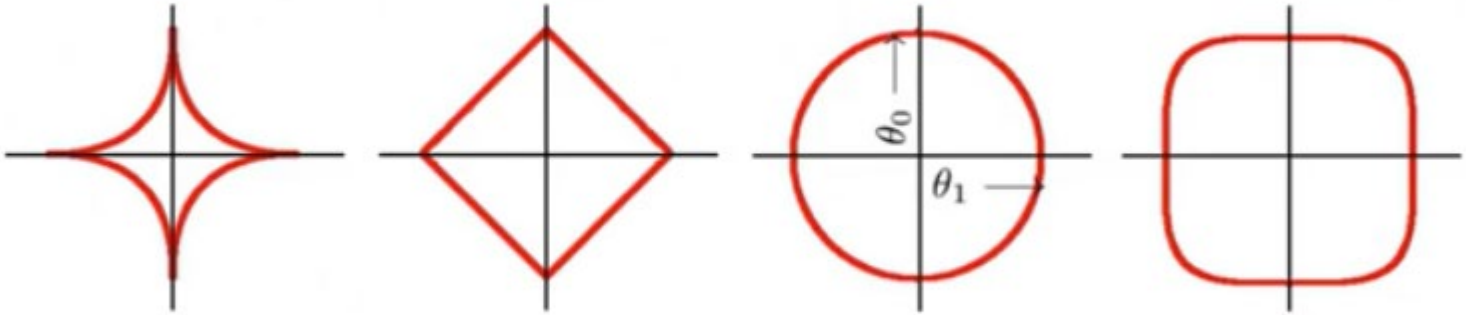
Regularization

- Let's not have too big numbers in the weight



- More generally, for the L_p regularizer: $\left(\sum_i |\theta_i|^p \right)^{\frac{1}{p}}$

Isosurfaces: $\|\theta\|_p = \text{constant}$



p=0.5 | p=1 | p=2 | p=4

Lasso | Quadratic

Lasso Regression | ***Ridge Regression***

L_0 = limit as $p \to 0$: "number of nonzero weights", a natural notion of complexity

Intuition

$$w_{\text{new}} = w - \eta \frac{\partial L}{\partial w}$$

- Loss Function

$$L = (\hat{y} - y)^2$$
$$= (wx + b - y)^2$$

- $\eta = 1$,
- $H = 2x(wx+b-y)$

L

$$w_{\text{new}} = w - H \quad \text{——} \quad (0)$$

L1

$$w_{\text{new}} = \begin{cases} (w - H) - \lambda, & w > 0 \quad \text{——} \quad (1.1) \\ (w - H) + \lambda, & w < 0 \quad \text{——} \quad (1.2) \end{cases}$$

pushing *w* towards 0.

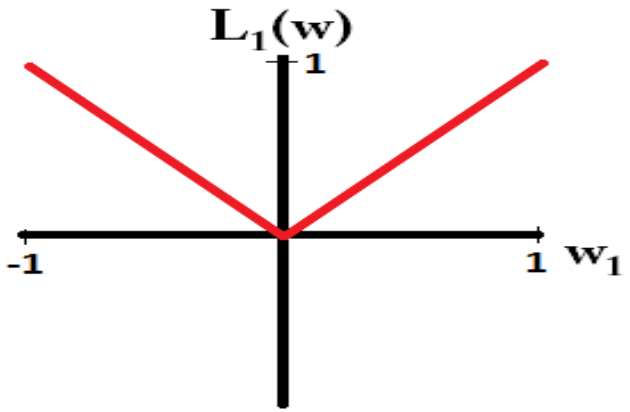
$$L_1 = (wx + b - y)^2 + \lambda |w|$$

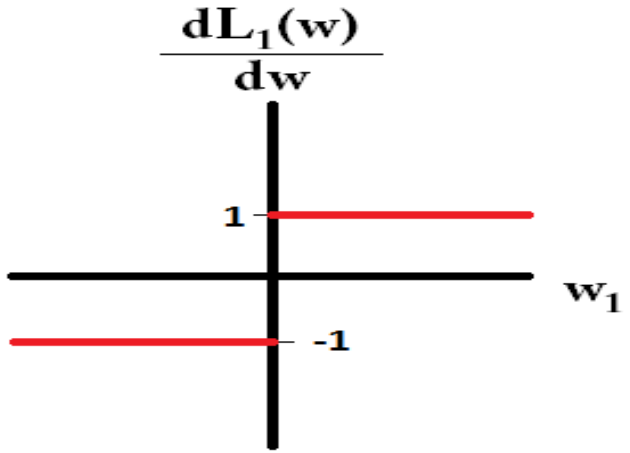
$$L_2 = (wx + b - y)^2 + \lambda w^2$$

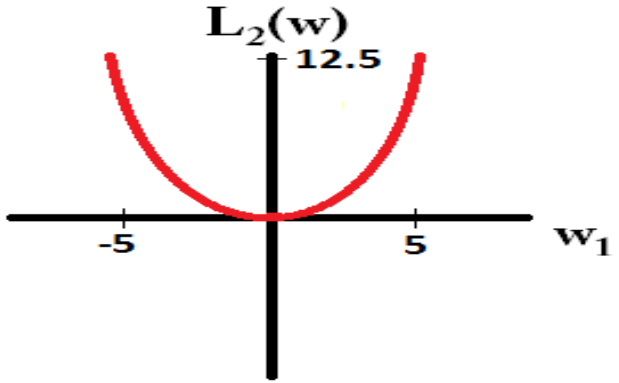
$$\hat{y} = 0.4561x_1 - 0.0007x_2 + 0.3251x_3 + 0.0009x_4 + 0.0001x_5 - 0.9142x_6 - 0.553$$

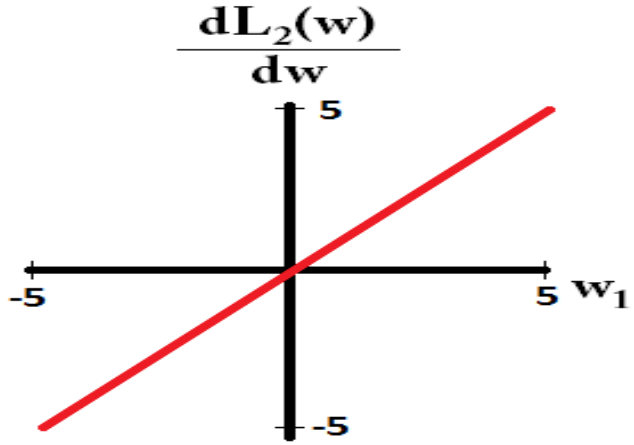
L2

$$w_{\text{new}} = (w - H) - 2\lambda w \quad \text{——} \quad (2)$$

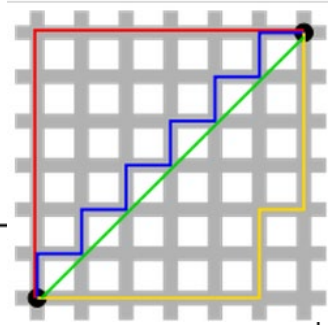

$L_1(w)$
1
-1
1
w_1


$\frac{dL_1(w)}{dw}$
1
-1
w_1


$L_2(w)$
12.5
-5
5
w_1


$\frac{dL_2(w)}{dw}$
5
-5
5
w_1
-5

L1 vs L2



Comparison of L1 and L2 regularization	
L1 regularization	**L2 regularization**
Sum of absolute value of weights	Sum of square of weights
Sparse solution	Non-sparse solution
Multiple solutions	One solution
Built-in feature selection	No feature selection
Robust to outliers	Not robust to outliers (due to the square term)

- Generally, the L2-norm is preferred in neural networks because it is differentiable for the backpropagation while the L1-norm is not.
- For most of general machine learning algorithms, the L2-norm doesn't perform well when there are outliers in the dataset. When outliers affect the cost significantly, the L1-norm would be preferred, or the L2-norm would be work after removing outliers.

Ex: The most common ways to prevent overfitting in **neural networks**:

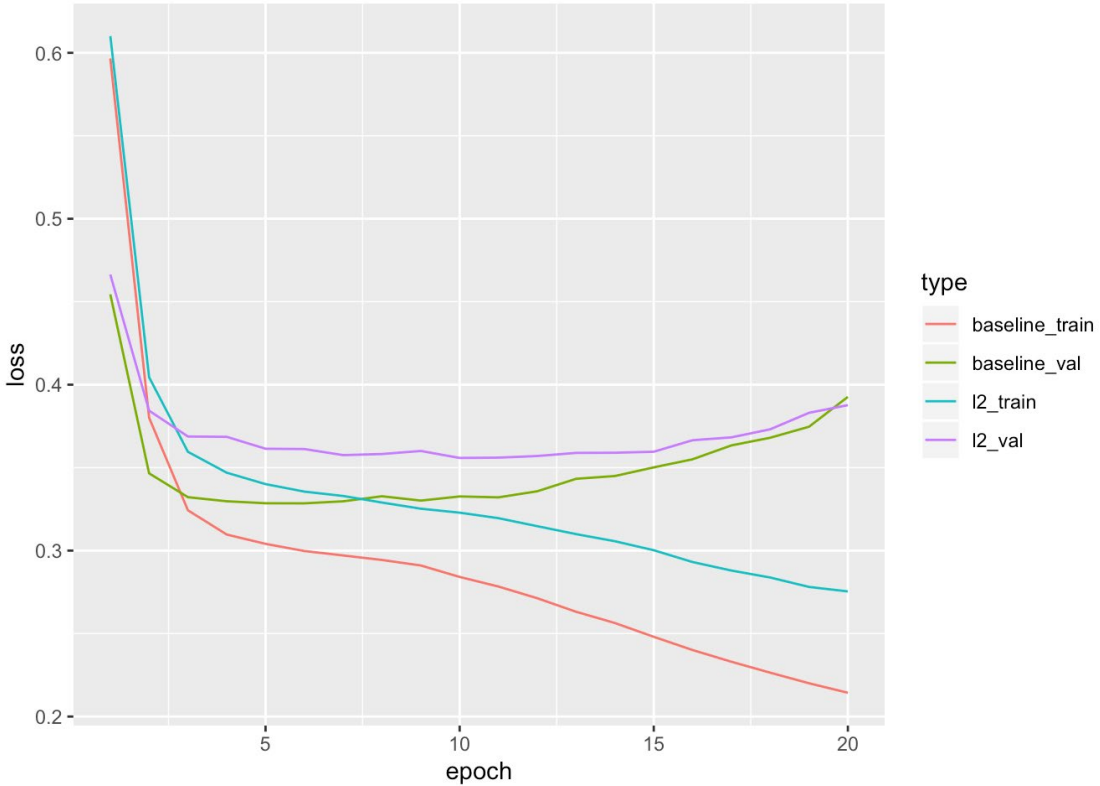
- Get more training data.
- Reduce the capacity of the network.
- Add weight regularization.

Configuration	Training Acc	Cross-val Acc	Test Acc
Base Model	99-100%	91-92%	90-91%
L2 Regularization	98-99%	92-93%	92-93%
Dropout Reg	99-100%	93-94%	93-94%

Ex: Add weight regularization in NN

- Thus a common way to mitigate overfitting is to put constraints on the complexity of a network by forcing its weights only to take small values

- This is called "weight regularization", and it is done by adding to the loss function of the network a cost associated with having large weights.



As you can see, the L2 regularized model has become much more resistant to overfitting than the baseline model, even though both models have the same number of parameters

What are Features?

- Features are attributes that their value make an instance.
- With features we can identify instances.
- Features are determinant values that determine instance belong to which class.

Why Feature Selection?

- Simple models are easier to interpret
- Shorter training time
- Enhanced generalization by reducing overfitting
- Variable redundancy

Feature Selection vs Feature Engineering

- Feature engineering allows us to create new features
- Feature selection, on the other hand, allows us to select features from the feature pool
- Dimensionality reduction modifies or transforms features into a lower dimension.

Feature Selection Methods

- Filter Methods
  - Apply different filters such as information gain, or relief algorithm [https://en.wikipedia.org/wiki/Relief_(feature_selection)]
- Wrapper Methods: uses a predictive machine learning algorithm to select the best feature subset
  - Forward/backward search

Wrapper Methods: Process

- **Search for a subset of features:** Using a search method (described next), we select a subset of features from the available ones.
- **Build a machine learning model:** In this step, a chosen ML algorithm is trained on the previously-selected subset of features.
- **Evaluate model performance:** And finally, we evaluate the newly-trained ML model with a chosen metric.
- **Repeat:** The whole process starts again with a new subset of features, a new ML model trained, and so on.

- **Forward Feature Selection:** This method starts with no feature and adds one at a time.
- **Backward Feature Elimination:** This method starts with all features present and removes one feature at the time.
- **Exhaustive Feature Selection:** This method tries all possible feature combinations.
- **Bidirectional Search:** And this last one does both forward and backward feature selection simultaneously in order to get one unique solution.

The Problem with Imbalanced Classes

- An imbalanced dataset means instances of one of the two classes is higher than the other, in another way, the number of observations is not the same for all the classes in a classification dataset

1. Oversampling: SMOTE (Synthetic Minority Over-sampling Technique): the minority class is over-sampled by producing synthetic examples. (https://arxiv.org/pdf/1106.1813.pdf)
2. Undersampling: this technique balances the imbalance dataset by reducing the size of the class which is in abundance (https://towardsdatascience.com/under-sampling-a-performance-booster-on-imbalanced-data-a79ff1559fab)
3. Cost-Sensitive Learning Technique: Use confusion matrix, F1, Recall etc. (https://www.csd.uwo.ca/~xling/papers/cost_sensitive.pdf)
4. Ensemble Learning Techniques: the ensemble technique is combined the result or performance of several classifiers to improve the performance of single classifier. (https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f)