COMP551 Regularization Tutorial

Lulan Shen

McGill University

lulan.shen@mail.mcgill.ca

October 7, 2022

Bias and Variance

There are two major sources of error in machine learning:

- **Bias** of trained model: is the average model close to the true solution.
- **Variance** of trained model: does it vary a lot if the training set changes?

In machine learning, we informally think of

- the difference between the Bayes error and the train error as algorithm's **bias**.
- the difference between training and validation error as algorithm's **variance**.

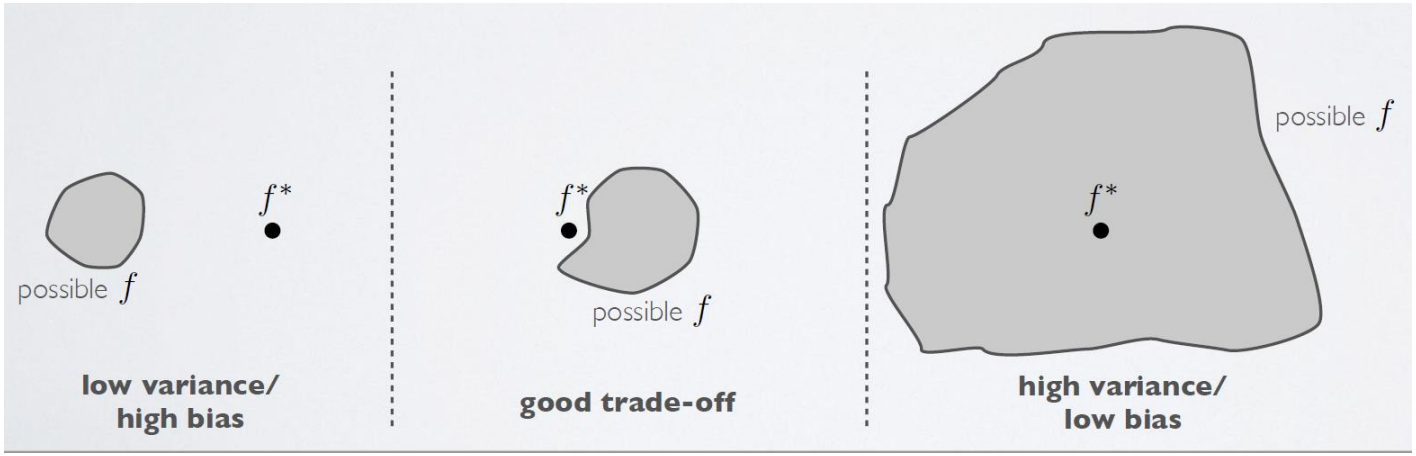
Note: Bayes error is the lowest possible prediction error that can be achieved.

Bias and Variance

Consider the object recognition problem, e.g. cats vs dogs.

- Assume human error is close to 0.

Train set error:	1%	15%	15%	1%
Validation set error:	10%	16%	25%	2%
	Low bias	High bias	High bias	Low bias
	High Variance	Low variance	High variance	Low variance



Bias and Variance in Machine Learning

Bias-variance trade-off:

- Generalization is the behaviour of the model on **unseen examples**.
- **Generalization error** can be seen as the sum of the (squared) **bias** and the **variance** if the error metric is mean square error (MSE).
- Test set serves to estimate the generalization performance (error).

Bias and Variance in Machine Learning

- High bias (training data performance) - Underfitting
  - Try a more complex model.
- High variance (validation set performance) - Overfitting
  - Try a simpler model.
  - Get more data.
  - Regularization.

Regularization (definition): Putting extra constraints on a machine learning model, to improve performance on the **test set** (not training set), either by encoding prior knowledge into the model or by forcing the model to consider alternative hypotheses that explain the training data.

Regularization

We denote the regularized objective function by J:

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \lambda \Omega(\boldsymbol{\theta}), \tag{1}$$

where $\lambda \in [0, \infty)$ is a hyperparameter that balances the relative effect of the regularization term, and $\Omega(\boldsymbol{\theta})$ is the parameter norm penalty term.

- $\lambda = 0$: no regularization
- Larger λ means more regularization.
- λ can be selected manually, or by cross-validation.

An effective regularizer is one that makes a profitable trade, that is, it reduces variance significantly while not overly increasing the bias.

Regularization

For neural network models, we typically have

$$\boldsymbol{\theta} = [\boldsymbol{w}^T b]^T \tag{2}$$

We typically choose to use a parameter norm penalty Ω that penalizes only the weights of the affine transformation at each layer and leaves the biases unregularized.

We therefore use the vector $\boldsymbol{w}$ to indicate all the weights that should be affected by a norm penalty, while the vector $\boldsymbol{\theta}$ denotes all the parameters, including both $\boldsymbol{w}$ and the unregularized parameters.

Ridge Regression (aka L2-Regularization)

- Only applied on weights, not on biases.
- The $L2$ parameter norm penalty is commonly known as **weight decay**.

$$\Omega(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{w}||_2^2 = \frac{1}{2}\sum_i \boldsymbol{w}_i^2 = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} \tag{3}$$

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w} \tag{4}$$

$$\nabla_{\boldsymbol{w}}\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = \nabla_{\boldsymbol{w}}J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \lambda\boldsymbol{w} \tag{5}$$

Ridge Regression (aka L2-Regularization)

Update the weights:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha(\nabla_{\boldsymbol{w}} J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \lambda \boldsymbol{w}), \tag{6}$$

After rearrangement,

$$\boldsymbol{w} \leftarrow (1 - \alpha\lambda)\boldsymbol{w} - \alpha \nabla_{\boldsymbol{w}} J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}). \tag{7}$$

The addition of the weight decay term multiplicatively **shrinks the weight vector** by a constant factor on each step, just before performing the usual gradient update.

Ridge Regression (aka L2-Regularization)

What happens over the entire course of training? Take linear regression as an example.

The closed-form solution:

$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}, \tag{8}$$

is changed to (details see Lecture Regularization pp15)

$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{9}$$

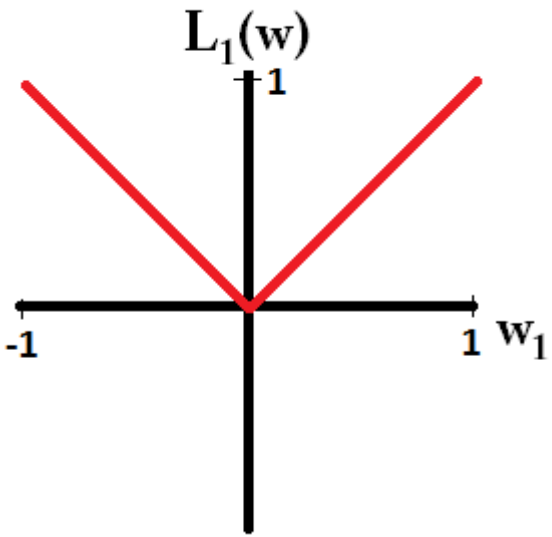
Lasso Regression (aka L1-Regularization)

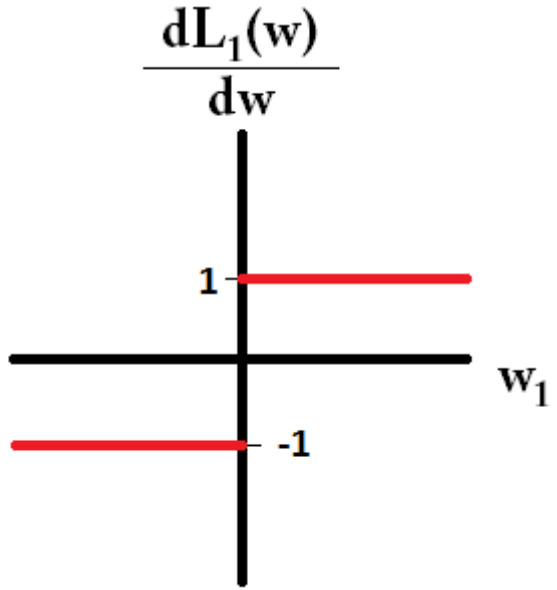
- Also only applied on weights, not on biases.
- Unlike L2, L1 will push certain weights to be exactly 0.
- No closed-form solution.

$$\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w}||_1 = \sum_i |\boldsymbol{w}_i| \tag{10}$$

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \lambda ||\boldsymbol{w}||_1 \tag{11}$$

$$\nabla_{\boldsymbol{w}} \tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = \nabla_{\boldsymbol{w}} J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \lambda \text{sign}(\boldsymbol{w}) \tag{12}$$





Practice

You want to train a linear classifier on a dataset for which you know that many features are irrelevant, i.e. not useful for predicting the output. Which type of regularization should you use?

- L2 regularization
- L1 regularization

Practice

You want to train a linear classifier on a dataset for which you know that many features are irrelevant, i.e. not useful for predicting the output. Which type of regularization should you use?

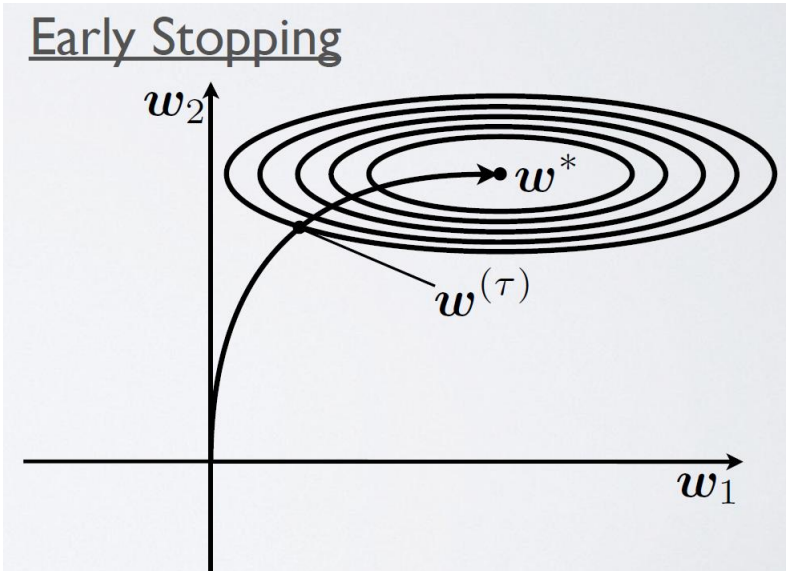
- L2 regularization
- L1 regularization

Solution: L1 regularization tends to set some weights to 0. Features that don't help in predicting the output are more likely to have a 0 weight, so for this problem L1 regularization would be appropriate to use. In contrast, with L2 regularization all weights will tend to be to be small, but non-zero, so there is a greater chance that the model will fit to the irrelevant features.

Early Stopping

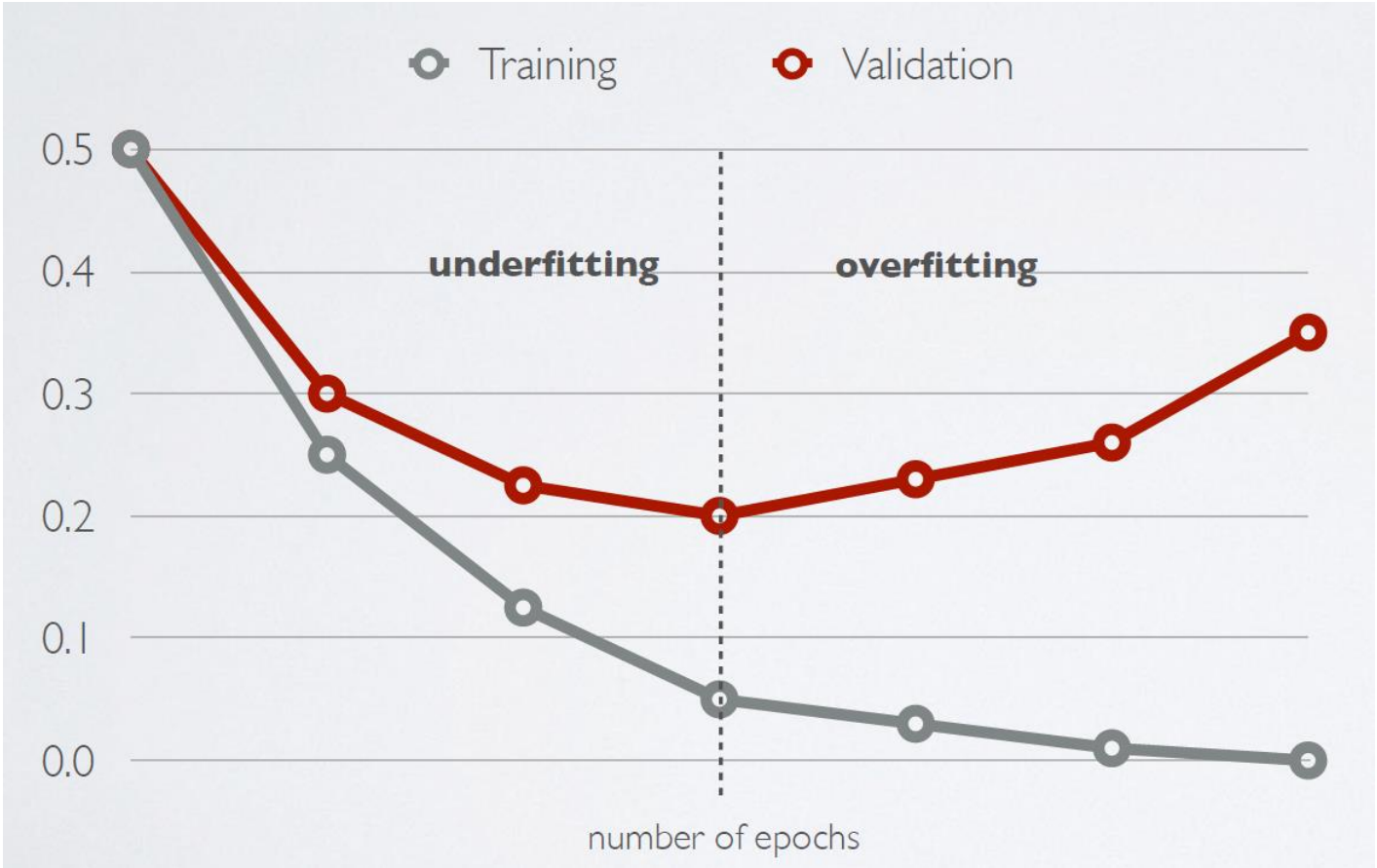
Early stopping acts as a regularizer.

- Early stopping has the effect of restricting the optimization procedure to a relatively small volume of parameter space in the neighbourhood of the initial parameter value.
- Assuming a simple linear model with a quadratic error function and simple gradient descent, early stopping is equivalent to L2 regularization.



Early Stopping

To select the number of epochs, stop training when validation set error increases (with some look ahead).



Early Stopping with Retraining

- Sometimes you really don't want to "waste" the validation set by not training on it.
- There are two basic strategies for retraining with the validation data.
  - Retrain with train+valid for the same number of epochs as determined by initial early stopping.
  - Continue training w/ train+valid until the loss on valid = early-stopped loss on the train data. Not guaranteed to stop.

Algorithm 2 A meta-algorithm for using early stopping to determining at what objective value we start to overfit, then continuing training.

Let $\boldsymbol{X}^{(\text{train})}$ and $\boldsymbol{y}^{(\text{train})}$ be the training set
Split $\boldsymbol{X}^{(\text{train})}$ and $\boldsymbol{y}^{(\text{train})}$ into $\boldsymbol{X}^{(\text{subtrain})}$, $\boldsymbol{y}^{(\text{subtrain})}$, $\boldsymbol{X}^{(\text{valid})}$, $\boldsymbol{y}^{(\text{valid})}$
Run early stopping (Alg. **??**) starting from random $\boldsymbol{\theta}$ using $\boldsymbol{X}^{(\text{subtrain})}$ and $\boldsymbol{y}^{(\text{subtrain})}$ for training data and $\boldsymbol{X}^{(\text{valid})}$ and $\boldsymbol{y}^{(\text{valid})}$ for validation data. This updates $\boldsymbol{\theta}$
$\epsilon \leftarrow J(\boldsymbol{\theta}, \boldsymbol{X}^{(\text{subtrain})}, \boldsymbol{y}^{(\text{subtrain})})$
while $J(\boldsymbol{\theta}, \boldsymbol{X}^{(\text{valid})}, \boldsymbol{y}^{(\text{valid})}) > \epsilon$ **do**
  Train on $\boldsymbol{X}^{(\text{train})}$ and $\boldsymbol{y}^{(\text{train})}$ for n steps.
end while

References

- Armanfard N., ECSE551 Lecture Notes, McGill University, Montreal, Canada, 2020
- Courville A., IFT6135 Lecture notes, University of Montreal, Montreal, Canada, 2021
- Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. Deep Learning. MIT Press.

The End