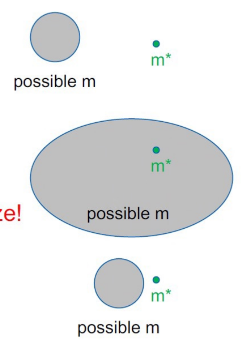
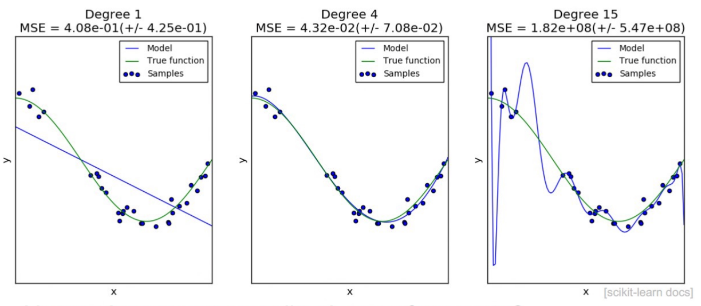
Bias-Variance Trade-Off

- high bias, low variance:
- low bias, high variance:
- good trade-off:



Under- vs. Overfitting



How to improve generalization performance?
- underfitting: more training or increase model *capacity*
- overfitting: less training or decrease model *capacity*

2) Generalization Error

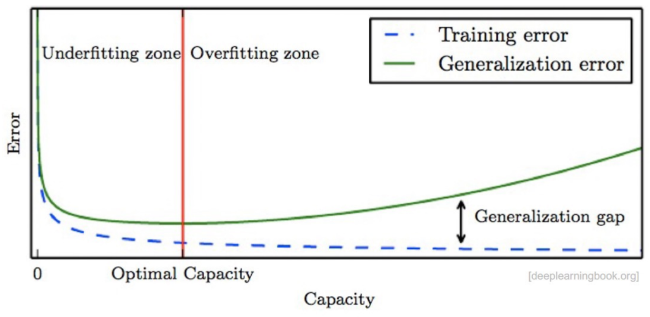
when traing with SGD
Stochastic Gradient descent

Can you think of a situation where no extra validation data is required to measure the generalization error?

Exception: 1st Epoch

- minibatch SGD follows the gradient of the **true generalization error** as long as no examples are repeated
- i.e. if each training sample is used only once, there is no need for a validation set

Model Capacity



evaluation and selection only based on performance on unseen data!

Regularization Techniques

- early stopping (7.8)
- weight regularization (7.1)
  - L1 = penalize non-zero values
  - L2 (“weight decay”) = penalize extreme weights
- adding [input / label] noise / denoising (7.5)
- sparse representations (WTA) (7.10)
- (layer-wise) unsupervised pretraining
- neighborhood-based methods
  - cross-trial / similarity constraint encoding
  - Siamese / triplet nets, magnet loss
- more data

Regularization is applied to reduce the train Error. where the model generalization should work for unseen data.

Regularization Techniques

- dataset augmentation (.4)
- semi-supervised learning (.6)
- multi-task learning (.7)
- parameter tying & sharing (.9)
- bagging / ensemble (.11)
- dropout (.12) = randomly set activations to zero
- DropConnect (.12) = randomly set weights to zero
- adversarial training (.13)
- tangent methods (.14) – rather outdated

The main intuitive difference between the L1 and L2 regularization is that L1 regularization tries to estimate the median of the data while the L2 regularization tries to estimate the mean of the data to avoid overfitting.

$x = [23, 29, 20, 32, 23, 21, 33, 25]$

$$\text{mean (Avg)} = \frac{\text{Sum}(x)}{\text{len}(x)}$$

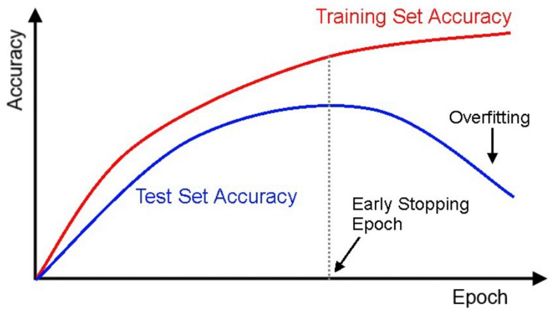
Median = 20, 21, 23, $\underbrace{23, 25}_{\text{middle.}}$, 29, 32, 33

$$\frac{23 + 25}{2} = 24$$

Another difference between them is that L1 regularization helps in feature selection by eliminating the features that are not important. This is helpful when the number of feature points are large in number.

Early Stopping

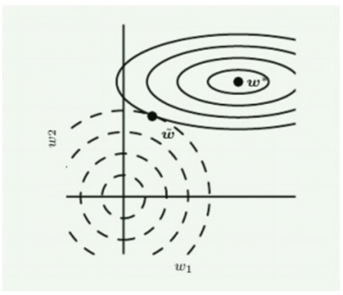
idea: monitor performance on validation set and stop when peak is reached



L2 regularization: L2 regularization is also called weight decay. This is because addition of the penalization term has modified the learning rule to multiplicatively shrink the weight vector by a constant factor on each step, just before performing the usual gradient update. The larger the weights, the bigger is the penalization and weights are gradually driven towards origin. Uses Euclidian distance.

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

$$\Omega(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$$



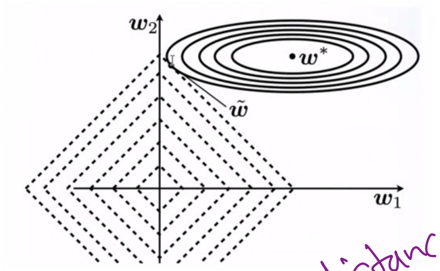
W* is the unregularized objective, i.e the left part of the objective function and dotted lines represent the error surface of the penalization objective. W~ is the equilibrium where both the objectives meet at equilibrium.

L1 regularization:

L1 regularization involves sum of the absolute values of individual parameters.

$$\Omega(\boldsymbol{\theta}) = \|\boldsymbol{w}\|_1 = \sum_i |w_i|,$$

L1 regularization results in a solution that is more sparse. This sparsity property of the L1 regularizer is used as a feature selection mechanism. Features with sparse weights can be discarded. Uses Manhattan distance.



mahattan distance

L1, L2 regularization used to Penalyze the bigger params.

eg: Loss = $\frac{1}{n}\sum_{i=1}^{n}(y_i - \underbrace{h_\theta(x_i)}_{\hat{y}})^2 + \lambda\sum_{i=1}^{n}\theta_i^2$

if θ is (L2) - Penalyze more more.

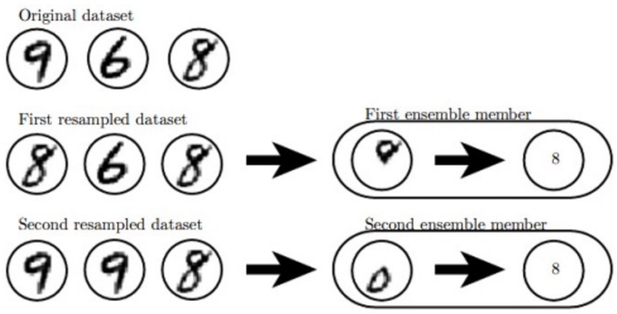
in (L1) we use $\lambda\sum_{i=1}^{n}|\theta_i|$

Explain data augmentation?

One way to make sure model generalizes better is to train on more data. It may happen that the data that we have is not enough to train the model, so we generate fake data out of the real data and add it to the training set. These techniques are mainly used for classification problems such as object detection. Since images are high dimensional, they can be mirrored, rotated, translated etc. Injecting noise to the input can also be thought of as data augmentation. Noise can also be added to the hidden units, providing multiple layers of abstraction.

Explain bagging?

Bagging, also called bootstrapping and aggregating is a regularization technique for reducing the generalization error by combining different models. The idea is to train several models separately, and make all the models vote on the output of the test example. The reason this works is that different models will usually not make all the same errors on the test set.



Bagging is a method that allows the same kind of model, training algorithm and objective function to be reused several times. If there are k models, then k datasets are created by rapidly sampling from the original dataset with replacements. The size of these k datasets are same as that of the original set. Each of these k models are then trained and predict for a test example.

Explain dropout? How does it relate to ensemble training?

Dropout like the name tells, is the process of dropping a node from the network based on a probability. It is a regularization technique since it limits the models capacity. Dropout trains the ensemble consisting of all subnetworks that can be formed by removing non output units from an underlying base network. A node can be removed from the network by multiplying its output value with 0. It can be thought of as averaging many models with shared weights.

To train with dropout, minibatch based learning is used. Each time an example is loaded into minibatch, a different set of binary masks are used to apply to all input and hidden units in the network.

Differences between dropout and bagging training:

Dropout	Bagging
Models share parameter, with each model inheriting different set of parameter from the original network.	Models are all independent.
Most models are not explicitly trained. The models are large enough that it would be infeasible to sample all possible subnetworks. Instead just a tiny fraction of the possible subnetworks are trained for each step.	Each model is trained to converge on its respective training set.

Similarity: dropout follows the bagging algorithm. For example, the training set encountered by each subnetwork is indeed a subset of the original training set sampled with replacement

https://medium.com/analytics-vidhya/l1-vs-l2-regularization-which-is-better-d01068e6658c