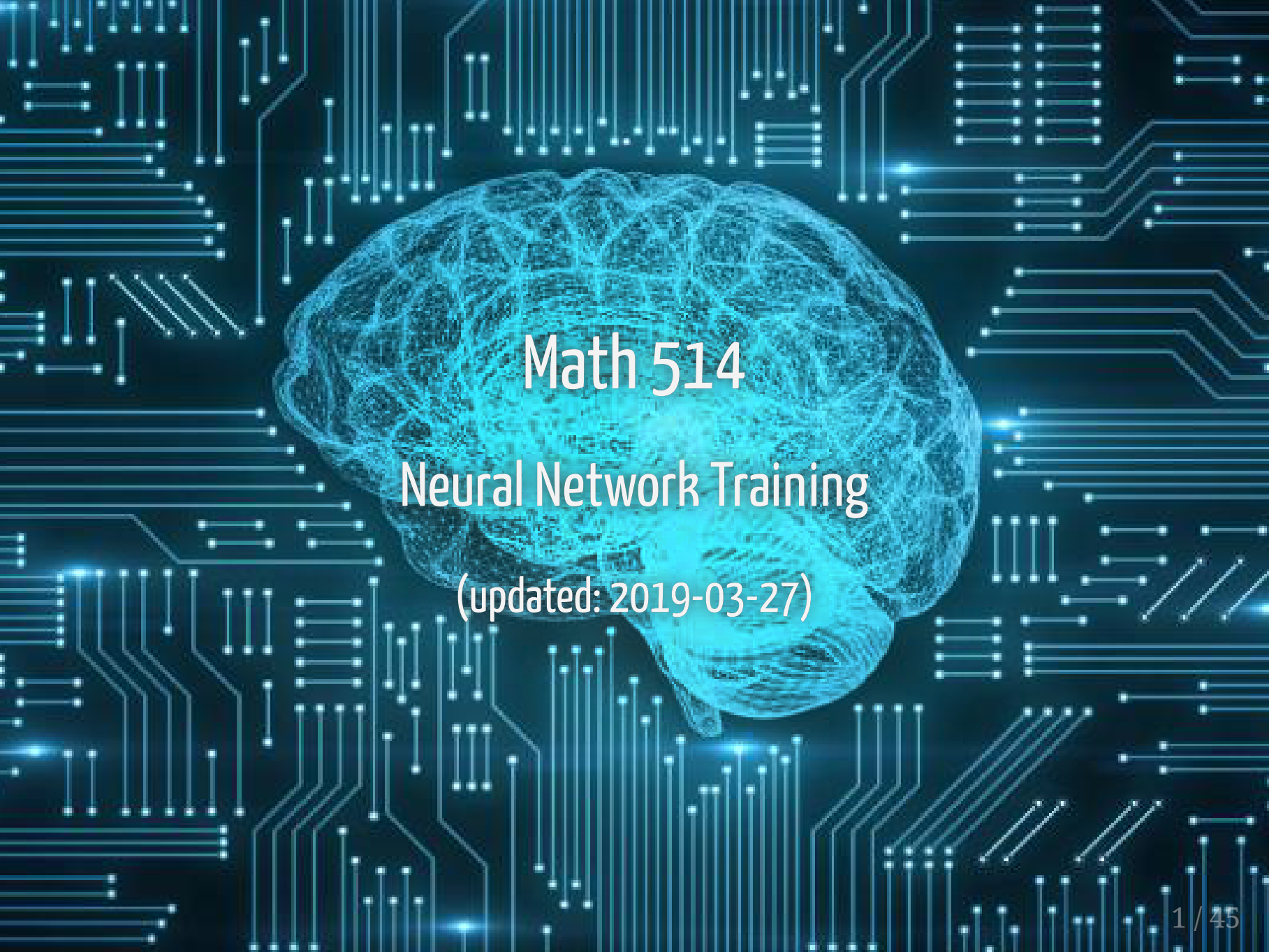


Math 514

Neural Network Training

(updated: 2019-03-27)

Class 8 - Training and Tuning

We now have many of the pieces needed to build a neural network:

- **Data set**
- **Task description** (classification)
- **Cost function/output activation** for task
- **Network architecture**
  - Number of layers
  - Number of nodes/layer
  - Hidden node activation
- **Algorithm Basics**
  - Forward Propagation
  - Backward Propagation
  - Optimizer
  - Rate decay

Today:

- Training/generalization
- Debugging
- Tuning

Training

The goal of training a neural network is to minimize a cost function while maximizing the ability to *generalize*.

"Learning with deep neural networks displays good *generalization* behavior in practice, a phenomenon which remains largely unexplained. A major issue still left unresolved ... is the precise relationship between optimization and *implicit generalization*."

Exploring Generalization in Deep Learning
Neyshabur, Bhojanapalli, McAllester and Srebro - 31st NIPS Conference, (2017)

Generalization

Increasing model complexity (more layers/nodes) means more parameters and gradient descent can drive cost function to essentially zero.

Consider credit card fraud model:

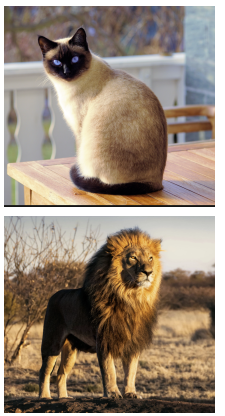
- Fraud is very rare.
- Imagine that the date/time of each fraud is part of the data
- A complex model could 'memorize' the dates and times of every fraud
- How will this model work on unseen fraud data?
  - If training set performance is based on memorizing dates/times then performance will be poor on a new data set

Generalization

Useful models must perform well on data not used during model development.

- The key assumption is that the "unseen" data is drawn from the same distribution as the training data.

Model Development



Model Deployment

Need to understand what "perform well" means.

Generalization

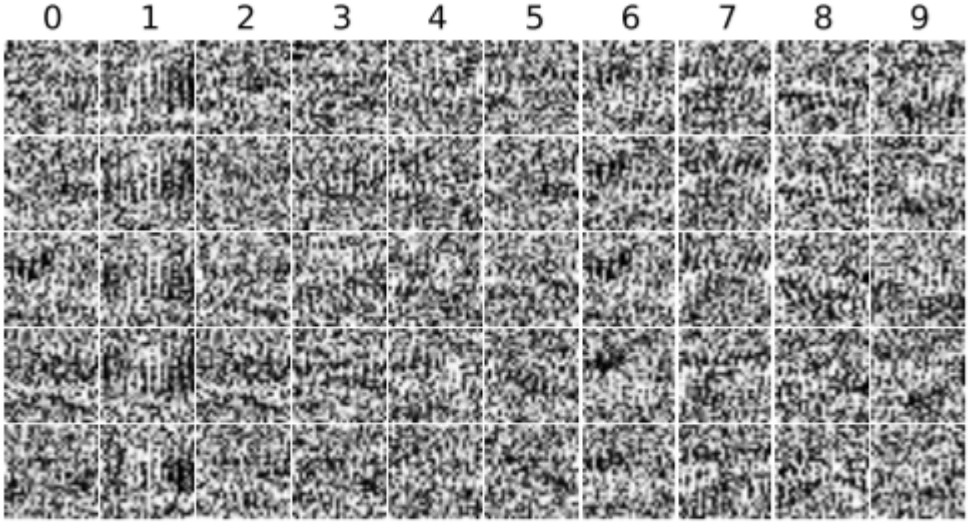
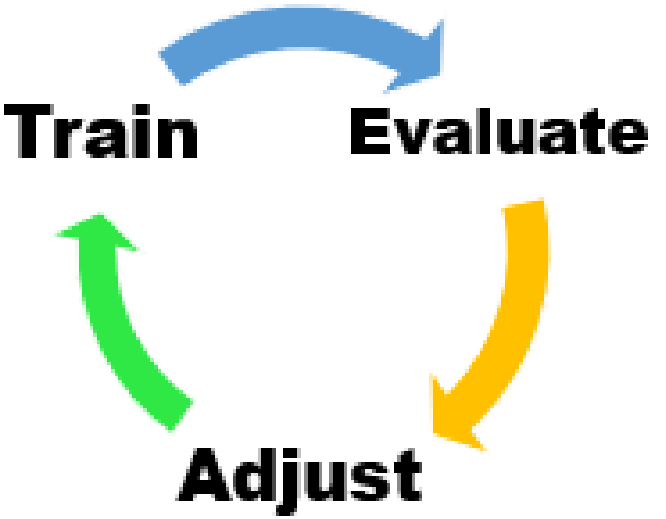


Figure 4. Directly encoded, thus irregular, images that MNIST DNNs believe with 99.99% confidence are digits 0-9. Each column is a digit class, and each row is the result after 200 generations of a randomly selected, independent run of evolution.

Training/Tuning

Training is iterative:

- Will use gradient descent to incrementally improve the model's adjustable parameters.
- The process is complex because of the numerous 'hyper-parameters.'
  - Optimization algorithm: α, β_1, β_2
  - Number of layers
  - Number of nodes/layer
  - Learning rate, rate schedule
  - Activation function
  - Batch size
  - Number of epochs

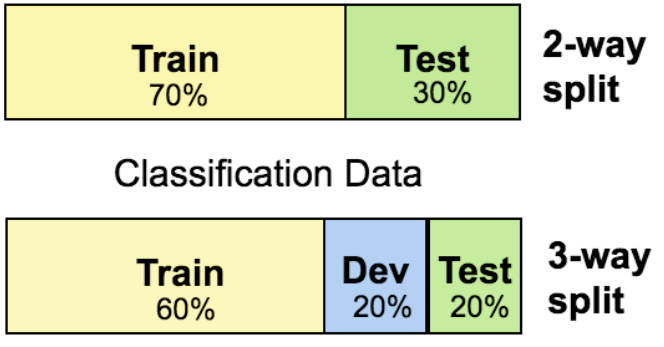


"One never finishes optimizing, one only abandons...." Neilsen

Performance

Evaluating model performance

- How small is the cost function?
- What is the prediction accuracy?
  - Classification models are trained using proxy cost functions. Cost and accuracy metrics can move in opposite directions.
- Accuracy is measured on data "held-out" from training.
  - Increasing model complexity can usually push training data accuracy to near 100%.

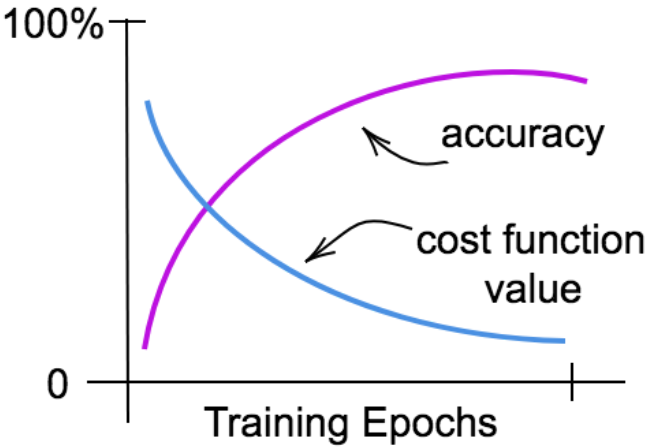


- Train data: Learn parameters.
- Dev/Validation: Tune hyper-parameters.
- Test: Score model/evaluate performance.
- All data randomly assigned.

Performance

As data set grows, percentages allocated to dev/test shrink.

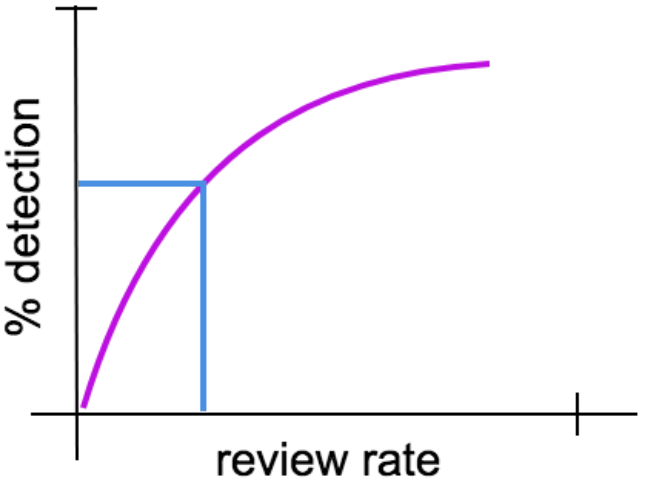
- 10k is probably enough for test. That's just 1% if 1,000,000 samples are available for training.
- If many (automated) models are evaluated, then test data in 2-way split loses value as measure of performance.
- Complex models can memorize training data responses. Performance on 'test' is what matters.



Rare Events

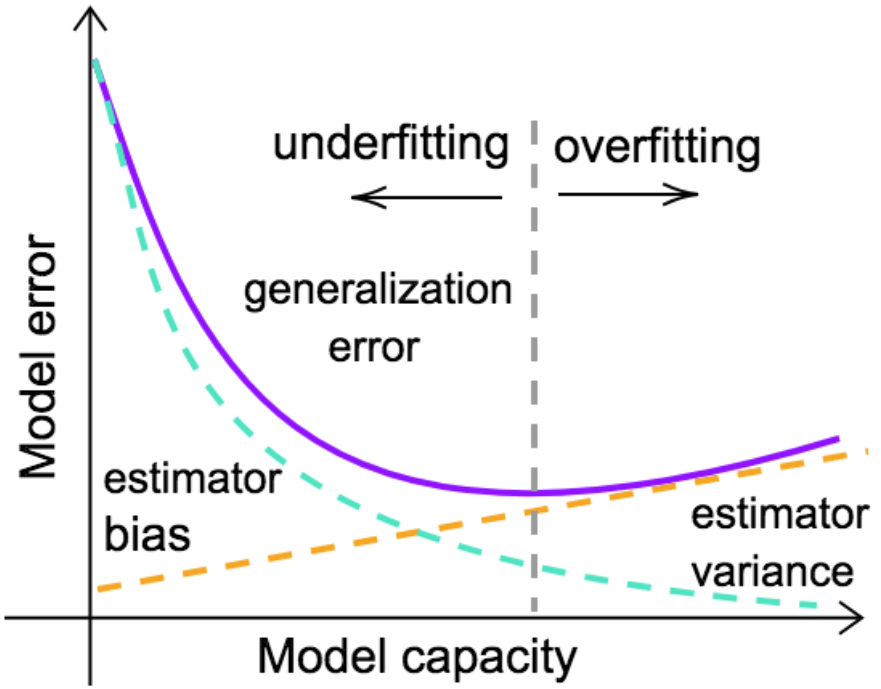
Credit card fraud is a rare event - maybe 1 in 1,000 or 10,000 transactions is fraudulent.

- A model that predicted 'not fraud' on all transactions would be right over 99% of the time.
- Stratified sampling is used to amplify the fraud signal before making the train-test split.
- A different accuracy measure is used.
- Scored transactions are sorted by score (highest first).



Model Capacity, Bias and Variance

- A model that underfits the data has **high bias.**
- A model that overfits the data has **high variance.**



Underfitting/Overfitting

- If there is bias, the model class does not contain the data function f and it **"underfits**."
- If there is variance, the model is overly general and it learns the noise. This is **"overfit**."
- If the model class contains the data function f then have unbiased estimates and the bias decreases when averaged over more models.
- Averaged high variance models have low bias.

To do:
Learn techniques to tune model complexity and optimize the bias-variance trade-off.

Decomposing Model Errors

Given a random variable X, what is the best single number to describe X

- Depends on the meaning of 'best'

Want to find d that minimizes $E\left[(X-d)^2\right]$

Let $\mu = E[X]$

$$\begin{aligned} E\left[(X-d)^2)\right] &= E\left[(X-\mu+\mu-d)^2)\right] \\ &= E\left[(X-\mu)^2\right] + E\left[(\mu-d)^2\right] + 2E\left[(X-\mu)(\mu-d)\right] \\ &= E\left[(X-\mu)^2)\right] + (\mu-d)^2 + 2(\mu-d)E\left[(X-\mu)\right] \\ &= \mathrm{Var}[X] + (\mu-d)^2 \end{aligned}$$

This shows 3 things:

- The expected squared error can be decomposed into variance +bias
- The error is minimized by setting bias to zero ($d = \mu = E[X]$)
- **Variance is irreducible error - property of the data and not dependent on estimate of d**

Decomposing Model Errors

Given data set $\{x^{(i)}, i = 1, \cdots, m\}$, find best estimate using squared error

Cost function

$$C = \frac{1}{2} \sum_{i=1}^{m} (x^{(i)} - d)^2$$

$$\frac{\partial C}{\partial d} = -\sum_{i=1}^{m} (x^{(i)} - d) = 0$$

$$d = \frac{1}{m} \sum x^i$$

This is the empirical result equivalent to last slide

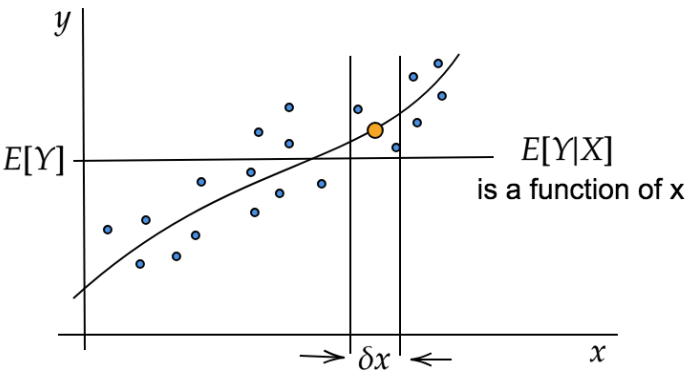
Decomposing Model Errors

Know $E[Y]$ is best constant estimate of Y when minimizing squared error

- Will show that $E[Y|X]$ is the best estimate of the Y among all functions of X

Will need the following results:

- $E[(U+V)|W] = E[U|W] + E[V|W]$
- $E[g(U)|U] = g(U)$
- $E[g(U)V|U] = g(U)E[V|U]$



Note:
Conditional expectation is the average w.r.t the conditional probability distribution
$p(y|x) = p(x,y)/p(x)$

Decomposing Model Errors

Want to show that among all functions of X, that $f(x) = E[Y|X]$ has the smallest squared error

$$
\begin{aligned}
E\left[(Y - g(X))^2|X\right] = & \\
& E\left[\Big(Y - E[Y|X] + E[Y|X] - g(x)\Big)^2|X\right] \\
& = E\left[(Y - E[Y|X])^2|X\right] + E\left[([Y|X] - g(X))^2|X\right] + \\
& 2E\left[(Y - E[Y|X])(E[Y|X] - g(X))|X\right]
\end{aligned}
$$

Last term is zero because

1. $E[Y|X]$ is a function of X so $E[\ E[Y|X]\ |X] = E[Y|X]$
2. $E\left[(Y - E[Y|X])|\ X\right] = E[Y|X] - E[Y|X] = 0$

Using (1) again gives:

$$
E\left[(Y - g(X))^2|X\right] = \boxed{E\left[(Y - E[Y|X])^2|X\right] + (E[Y|X]) - g(X))^2}
$$

This is minimized when $g(X) = E[Y|X]$

Bias-Variance

- Best squared error estimate of X is $E[X]$
- Best squared error regression of Y on X is $E[Y|X]$

Want to use these results to analyze models

Assume there is a collection of data sets $\left\{\mathbf{x}^{(i)}, t^{(i)}\right\}, \quad i = 1, \ldots, n$ drawn from the same joint distribution

Just showed $E\left[(t - g(X))^2|X\right] = E\left[(t - E[t|X])^2|X\right] + (E[t|X] - g(X))^2$

$E\left[(t - E[t|X])^2|X\right]$ does not depend on $g(x)$. It is irreducible error. Will decompose the second term.

Bias-Variance

$(E[t|X] - g(X))^2$ is a function of X and gives the squared differences between the ideal regression $E[t|X]$ and the model $g(x)$

This depends on the choice of training data, so want to average over data sets. Recall that

$$E\left[(X-d)^2)\right] = \mathrm{Var}[X] + (E[X] - d)^2$$

Let $f(X) = E[t|X]$ be the optimal regression using earlier result on decomposition of squared error

$$E_{\mathcal{D}}\left[\Big(g(X) - f(X)\Big)^2\right] = \underbrace{E_{\mathcal{D}}\left[(g(x) - E_{\mathcal{D}}\left[g(x)\right])^2\right]}_{variance(g)} + \underbrace{\Big(E_{\mathcal{D}}\left[g(X)\right] - f(X)\Big)^2}_{bias^2(g)}$$

Bias-Variance

Given a collection of data sets $\left\{x_j^{(i)}, t_j^{(i)}\right\}$ $i = 1, \cdots, m$ and $j = 1, \cdots, n$. The mean model, averaged over the n data sets is the following function if x

$$E_{\mathcal{D}}\left[g(x)\right] = \bar{g}(x) = \frac{1}{n}\sum_{j=1}^{n} g_j(x)$$

The mean squared bias computed over a sample of x-values $x^{(k)}$ is

$$\frac{1}{K}\sum_{k=1}^{K}(\bar{g}(x^{(k)}) - f(x^{(k)}))^2$$

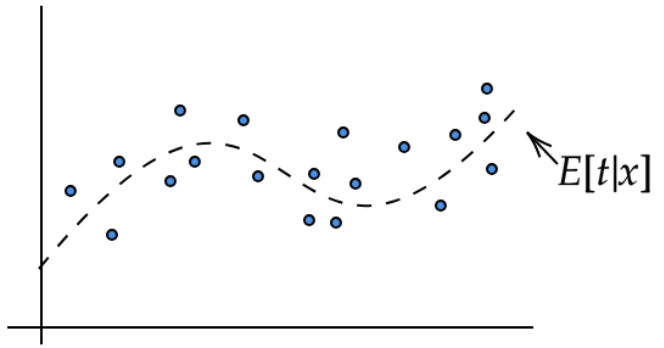
The variance at a point x is

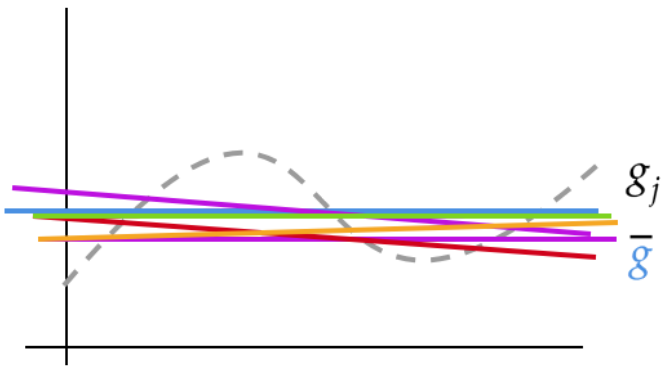
$$\frac{1}{n}\sum_{j=1}^{n}\Big(g_j(x) - \bar{g}(x)\Big)^2$$

The mean variance term computed over a sample of x-values $x^{(k)}$ is

$$E_{\mathcal{D}}\left[(g_j(X) - \bar{g}(X))^2\right] = \frac{1}{K}\cdot\frac{1}{n}\sum_{j=1}^{M}(g_j(x_j^{(k)}) - \bar{g}(x^{(k)}))^2$$

Bias-Variance





- Bias term measures averaged square distance between $\bar{g}$ and the dashed line $f(x) = E[t|X = x]$
- Variance term measures variability between g_j and $\bar{g}$.

Low order fit

- High bias - does not follow data trend
- Low variance - each fit is similar

High order fit

- Low bias - follows noise in the data
- High Variance - fits vary with each sample

Bias-Variance

Bias- Variance dilemma:

"Neural Networks and the Bias/Variance Dilemma" German, Bienenstock and Doursat - Neural Computation 4, 1 - 58 (1992)

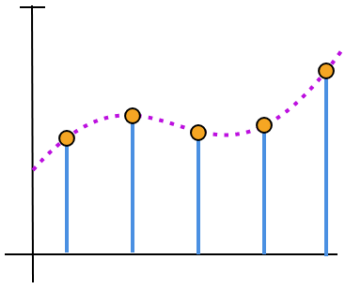
- Bias indicates model is not general (complex) enough. Called *underfitting*.
- High Variance indicates model is too general and is learning irrelevant details (noise). Called *overfitting*.
- The optimal model makes the best trade-off between bias and variance.

Additional lessons:

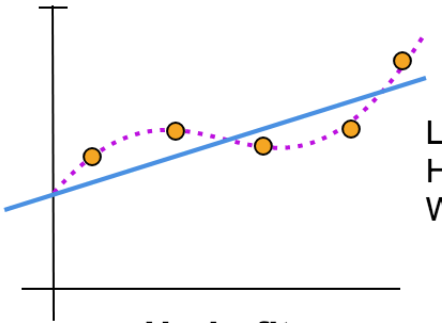
- Variance is reduced by more data, so model with low bias and lots of data ideal.
- High variance models have low bias, so averaging high variance models is effective.

Underfitting/Overfitting

Fit which follows dashed curves will match any data set sample.

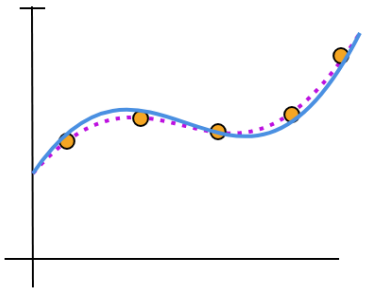


Dashed curve represents mean data distribution $E[Y|X]$. Training data samples include noise.

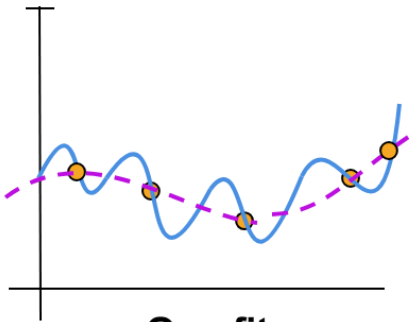


Low capacity.
High bias.
Won't generalize.

Underfit



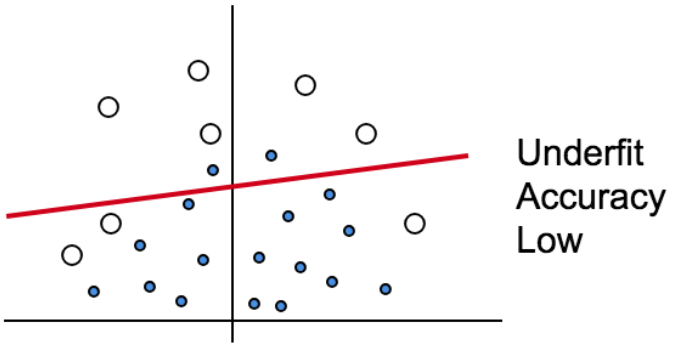
Generalizes well to any sample.

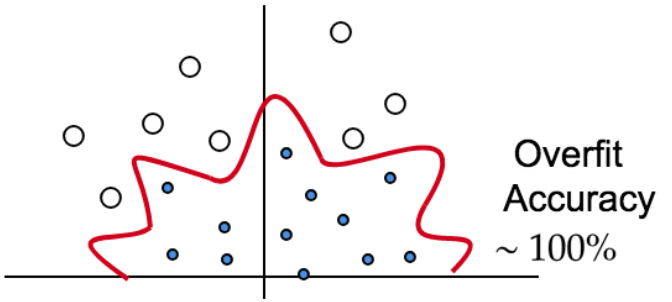


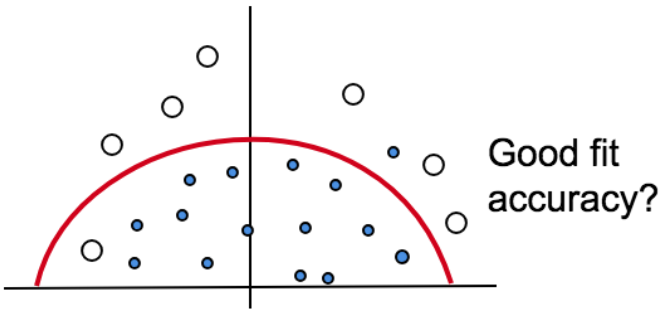
High capacity.
Low bias.
Won't generalize

Overfit

Bias-Variance







Note: If data is separable then training accuracy of 100% is good.

Bias-Variance

In general don't have lots of data sets.

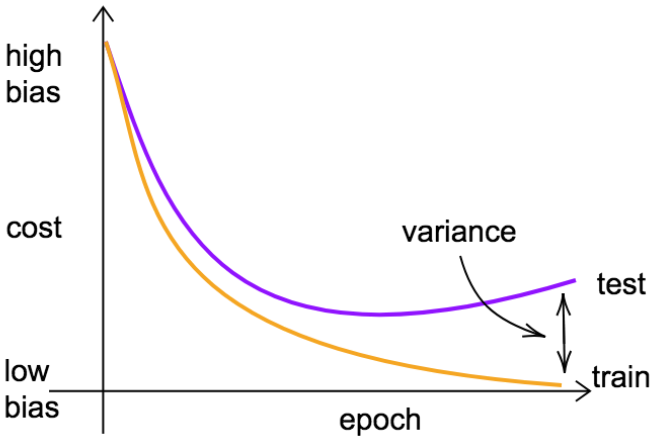
- Train - Test
- Train - Develop - Test

As training progresses, have a sequence of models

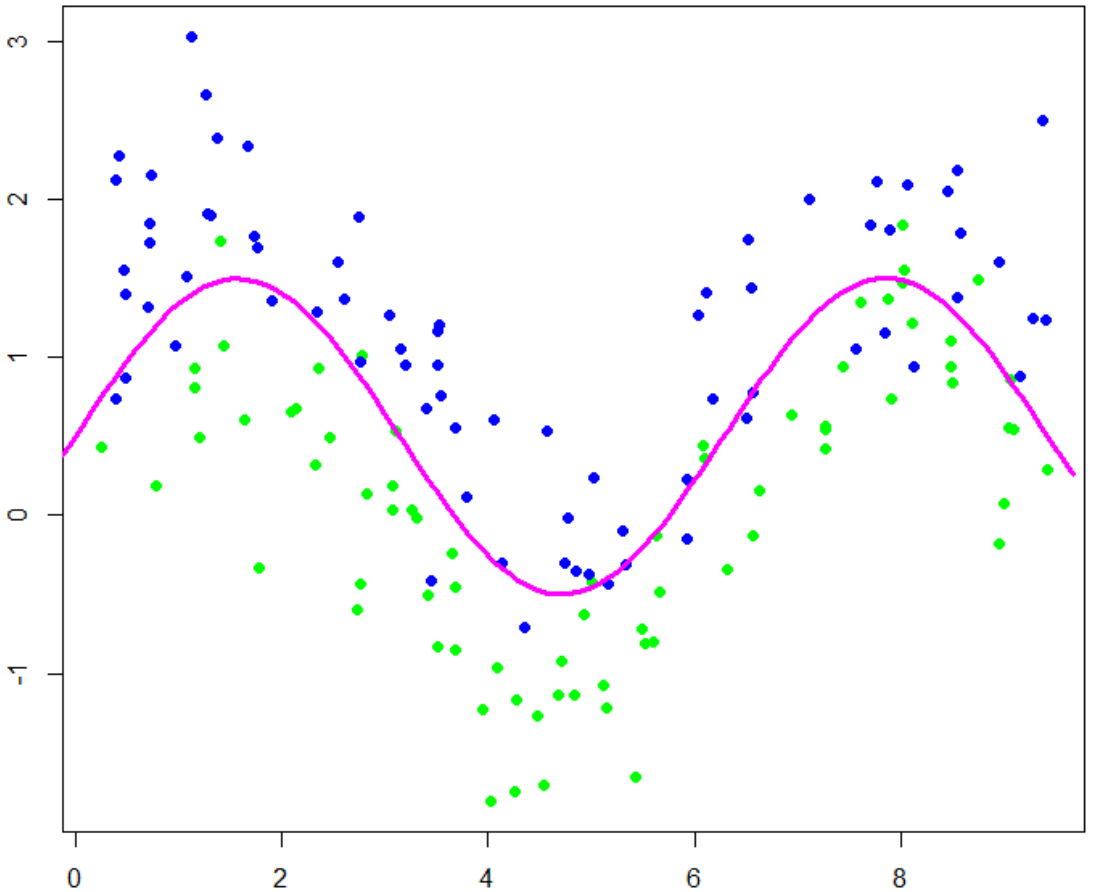
- Every weight update is a new model.

Want to monitor models to optimize the bias-variance trade-off.

- Will compare metrics on the train and test data.

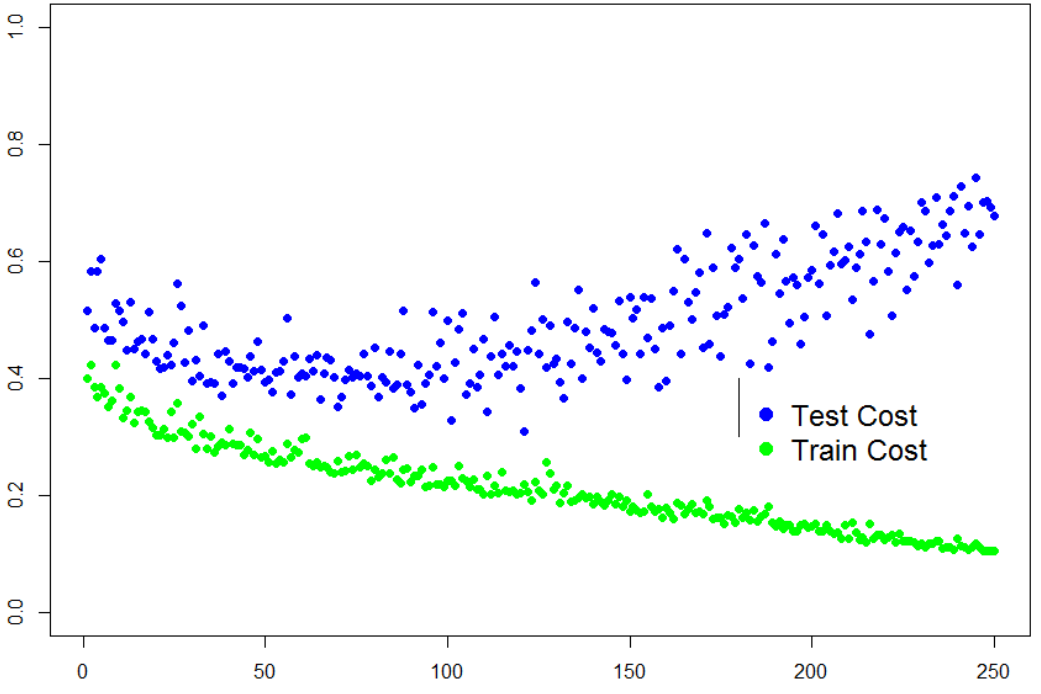


Synthetic Data Set

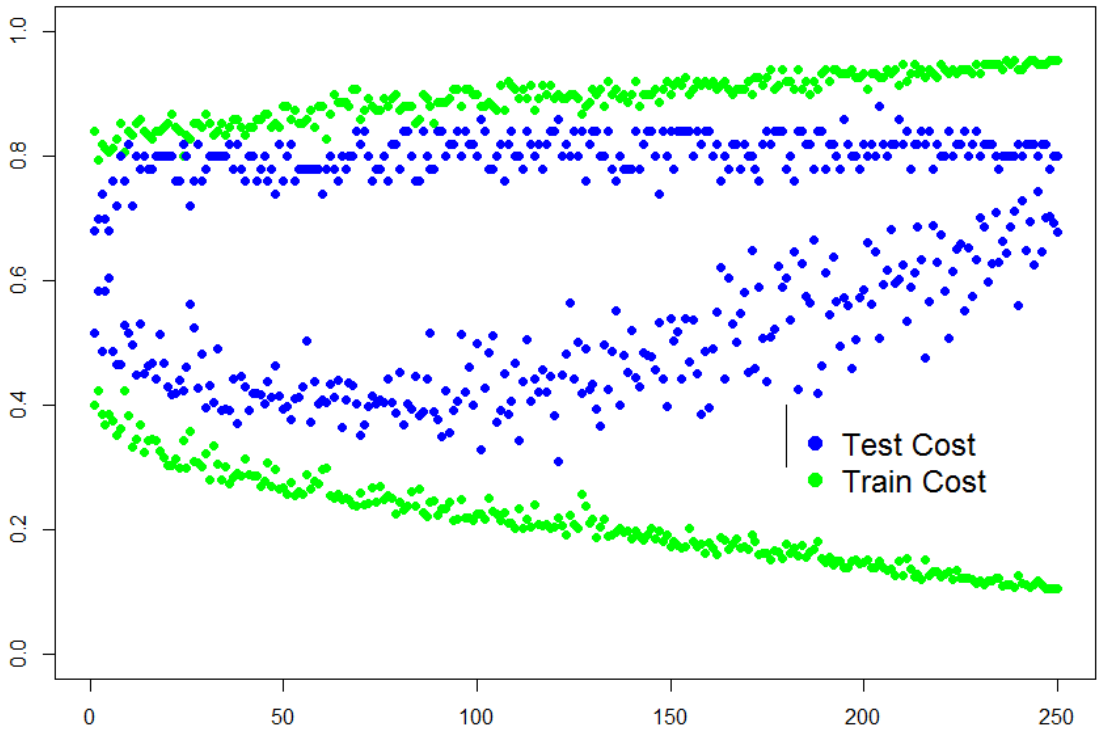


Overfitting - Cost

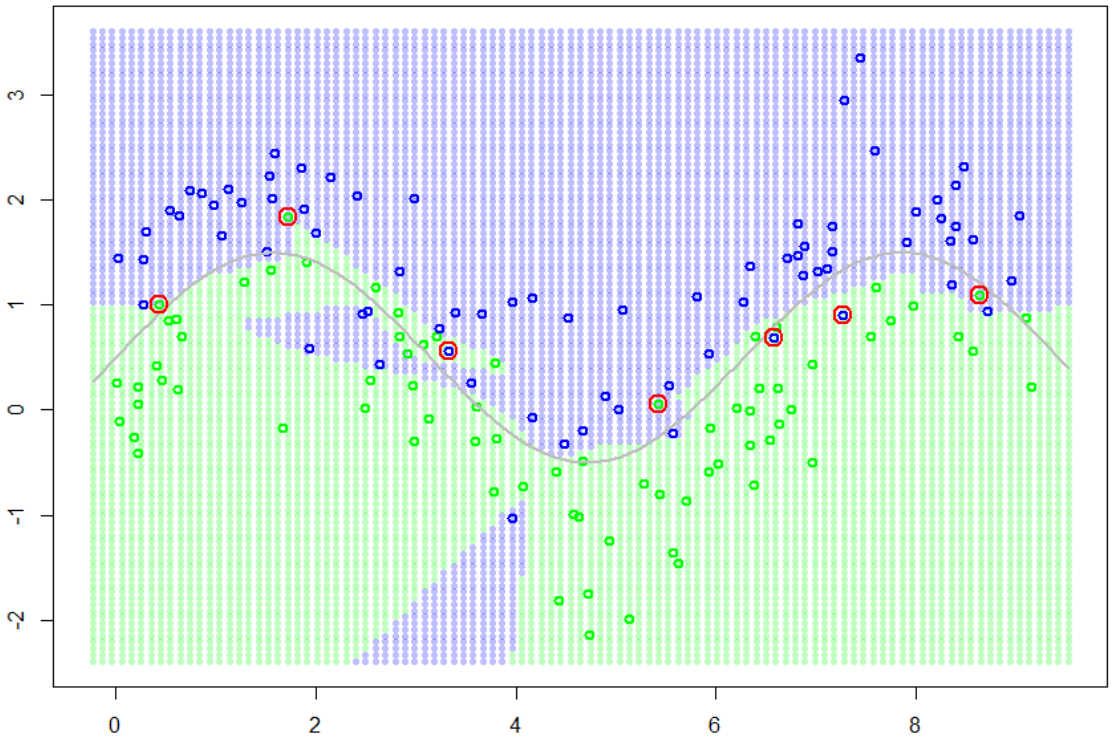
- 3 Hidden layers, 100/100/50
  - Over 15,000 weights
- 150 Epochs
- Nesterov, $\beta = .9$
- Rate decay $.5 - .05$, linear
- Sigmoid output, NegLL cost
- Initial model: no regularization



Overfitting - Cost/Accuracy



Overfitting - Decision Boundary



Practical Considerations

Neural networks can be difficult to train. One key to increasing predictability is to normalize the input data and carefully initialize model weights and biases.

- Modest sized inputs and weights help avoid tails of activation functions.

Controlling model complexity (overfitting) is called regularization. There are many techniques. One of the simplest is weight decay.

1. Dataset normalization
2. Weight initialization
3. Weight decay regularization.

Dataset Normalization

Normalization parameters are determined from the training dataset and then re-used for the dev/test data.

For data $\left\{\mathbf{x}^{(i)}\right\}$ $, i = 1, \cdots, m$

$$\hat{\boldsymbol{\mu}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}^{(i)}$$

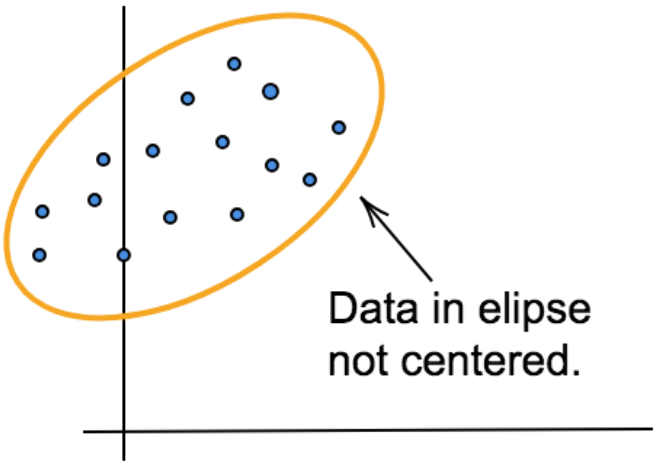
$$\hat{\boldsymbol{\sigma}}^2 = \frac{1}{m-1}\sum_{i=1}^{m}(\mathbf{x}^{(i)} - \hat{\boldsymbol{\mu}})^2$$

$$\tilde{\mathbf{x}} = \frac{\mathbf{x} - \hat{\boldsymbol{\mu}}}{\hat{\boldsymbol{\sigma}}}$$

- Note that the equations are vector equations. Each data component is centered and scaled by $\hat{\boldsymbol{\sigma}}$

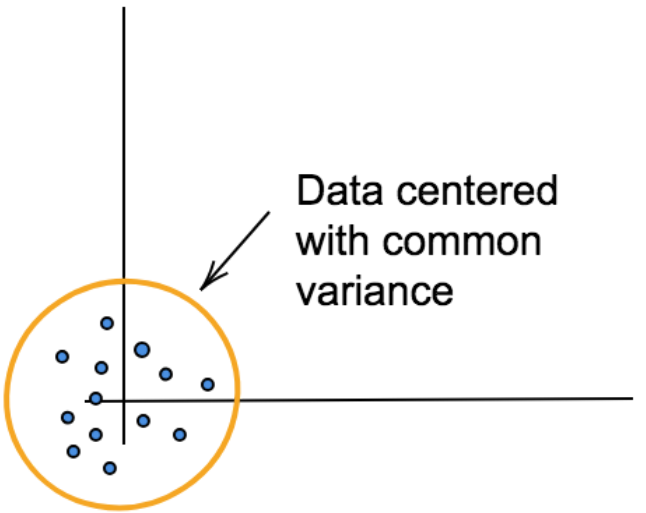
Dataset Normalization

Input Data



Transformed Input Data

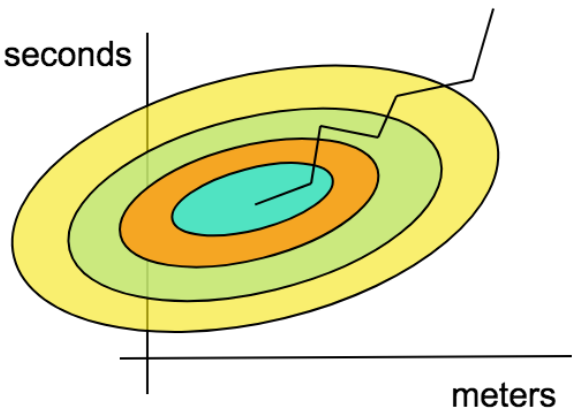
All components of the transformed data will have zero mean and variance 1

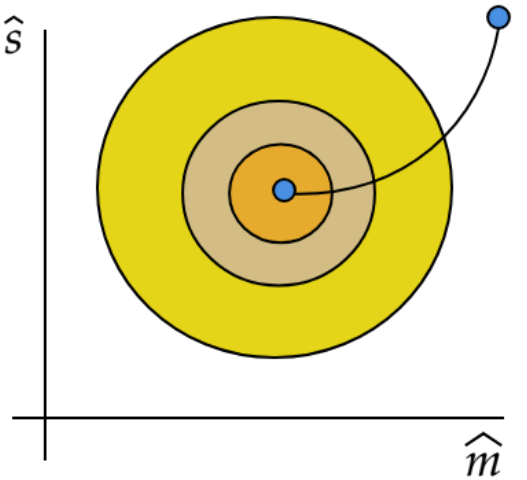


Dataset Normalization

Benefit of dataset normalization:

- All components of the transformed data $\underline{\mathbf{x}}$ have a common scale.
- This should reduce scale effects on the shape of local minima.





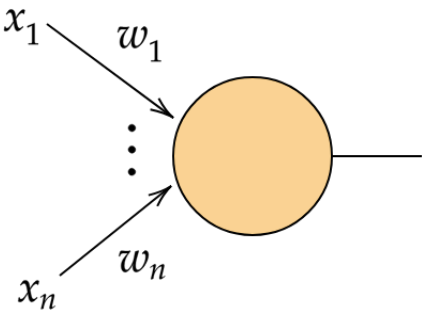
Note:
The components of $\underline{\mathbf{x}}$ can also be decorrelated. This will be covered later.

Weight Initialization

Weight initialization plays an important role in network dynamics. The idea is to start the network randomly in a way that is invariant as the network size changes.

First:
Weights should not be initialized to 0. Initializing with 0's introduces symmetries which persist.

The input dataset was normalized, so each component has common variance '1'. Now want to carry this idea over to the weight space.



Bias terms can be initialized to zero. The net-value when $\mathbf{b} = 0$ is:

$$\mathbf{z} = \sum_{i=1}^{n} \mathbf{w}_i \mathbf{x}_i$$

Weight Initialization

Will show if $E[X] = E[Y] = 0$ then

$$\mathrm{Var}[XY] = \mathrm{Var}[X]\,\mathrm{Var}[Y]$$

$$\mathrm{Var}[Z] = \mathrm{Var}\left[\sum W_i X_i\right]$$

Assuming independence, zero means and common variances

$$\begin{aligned} &= \sum_i \mathrm{Var}[W_i X_i] \\ &= \sum_i \mathrm{Var}[W_i]\,\mathrm{Var}[X_i] \\ &= n\,\mathrm{Var}[W]\,\mathrm{Var}[X] \end{aligned}$$

If the activation function acts like an identity function for small z, then the node output variance will be like $\mathrm{Var}[X]$ if:

$$n\,\mathrm{Var}[W]\,\mathrm{Var}[X] = \mathrm{Var}[X]$$

or

$$\mathrm{Var}[W] = \frac{1}{n}$$

Weight Initialization

$$\text{Var}[W^{[\ell]}] = \frac{1}{n_\ell}$$

$$\text{Var}[W^{[\ell]}] = \frac{2}{n_\ell + n_{\ell+1}}$$

ReLU activation is only non-zero when $z > 0$, so weight variance is doubled

$$Var[W^{[\ell]}] = \frac{2}{n_\ell}$$

$$Var[W^{[\ell]}] = \frac{4}{n_\ell + n_{\ell+1}}$$

There are a number of other similar initialization expressions.

Weight Initialization

Assume X, Y are independent. It follows that

$$E[XY] = E[X] \cdot E[Y]$$

$$E[f(X) \cdot g(Y)] = E[f(X)] \cdot E[g(Y)]$$

This implies

$$E[(XY)^2] = E[X^2 Y^2] = EX^2 \cdot EY^2$$

Result follows from

$$\begin{aligned}
\operatorname{Var}[XY] &= E[(XY)^2] - (E[XY])^2 \\
&= EX^2 EY^2 - (EX)^2 (EY)^2 \\
&= (\operatorname{Var} X + (EX)^2)(\operatorname{Var} Y + (EY)^2) - (EX)^2 (EY)^2 \\
&= \operatorname{Var} X \operatorname{Var} Y + \operatorname{Var} X (EY)^2 + \operatorname{Var} Y (EX)^2
\end{aligned}$$

So if $EX = EY = 0$, $\operatorname{Var}[XY] = \operatorname{Var} X \cdot \operatorname{Var} Y$

Regularization

Weight Decay

Regularization is used when a model has a high variance

- Large gap between training and test cost function

With a σ activation function, showed that large weight values are associated with very steep decision boundaries. It follows that adding a penalty for large weight values may smooth irregular decision boundaries.

L_2 regularization

$$\tilde{C} = C(\mathbf{x}, t; W) + \frac{\lambda}{2m}\|W\|^2$$

$$\|W\|^2 = \|W\|_F^2 = \sum_l \sum_{i,j} (W_{ij}^{[l]})^2$$

L_1 regularization

$$\tilde{C} = C(\mathbf{x}, t; W) + \frac{\lambda}{m}\|W\|_1$$

$$\|W\|_1 = \sum_l \sum_{i,j} |W_{ij}^{[l]}|$$

Regularization

Gradient descent with L_2 regularization

$$\frac{\partial}{\partial W_{ij}^{[l]}}\Big(\frac{\lambda}{2m}\sum_{l}\sum_{i,j}(W_{ij}^{[l]})^2\Big) = \frac{\lambda}{m}W_{i,j}^{[l]}$$

Update rule:

$$\begin{aligned} W^{[l]} &= W^{[l]} - \alpha(\nabla_W \mathcal{C} + \frac{\lambda}{m}W^{[l]}) \\ &= \underbrace{(1-\alpha\frac{\lambda}{m})}_{\text{decay coefficient}} W^{[l]} - \alpha\nabla_W\mathcal{C} \end{aligned}$$

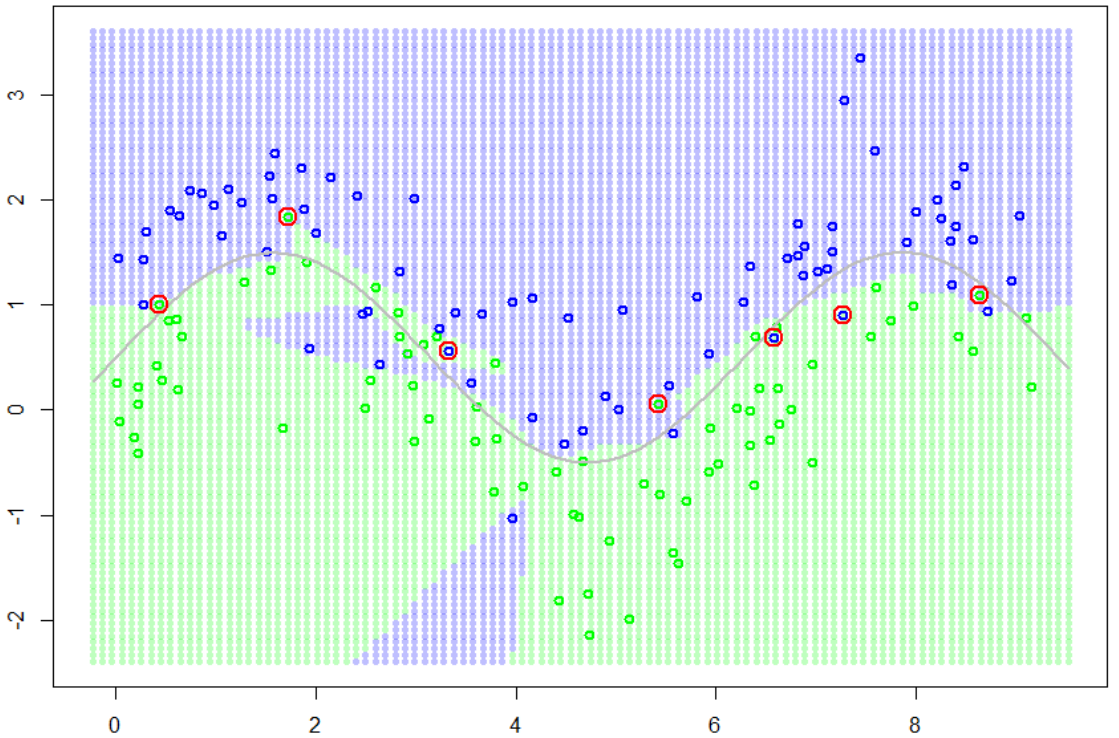
Note: The $1/m$ requires that λ be expressed in terms of batch size.

Regularization

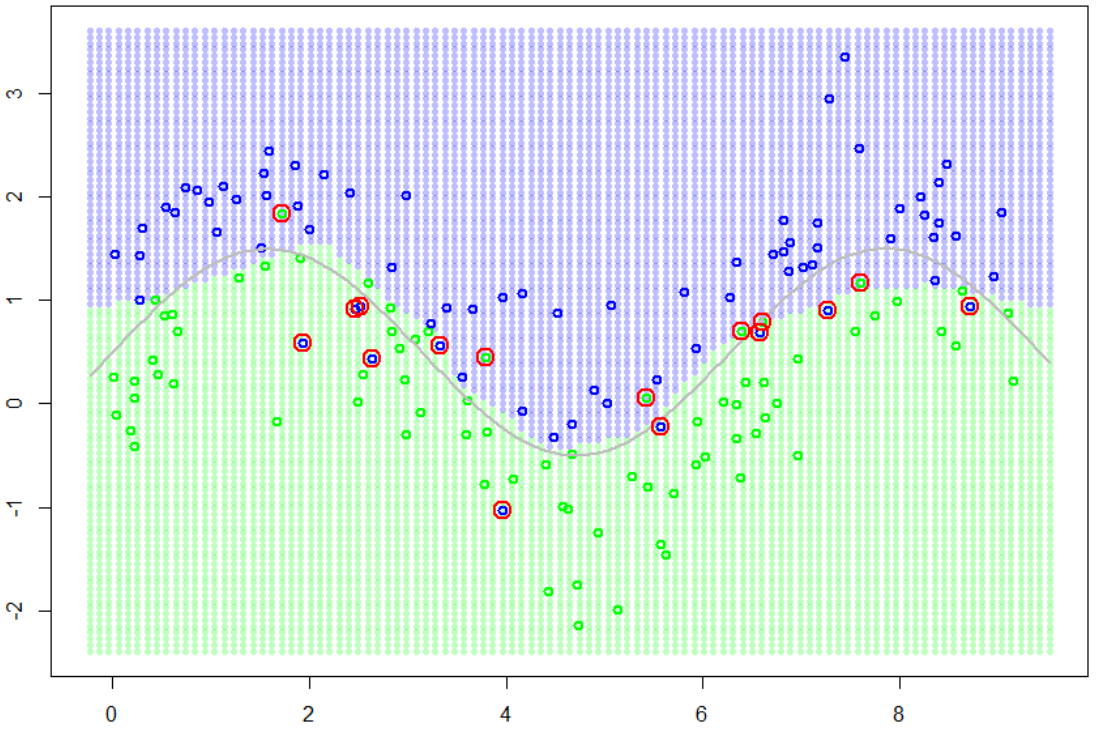
Heuristic argument why L_2 regularization helps:

- Reducing the size of the weights effectively removes some nodes from the network and implicitly reduces model complexity.
- Smaller weights avoid the tails of squashing activation functions and result in linear-like dynamics.
- Rule of thumb is to use complex network and regularize using weight decay or other methods (later lectures).

Regularization

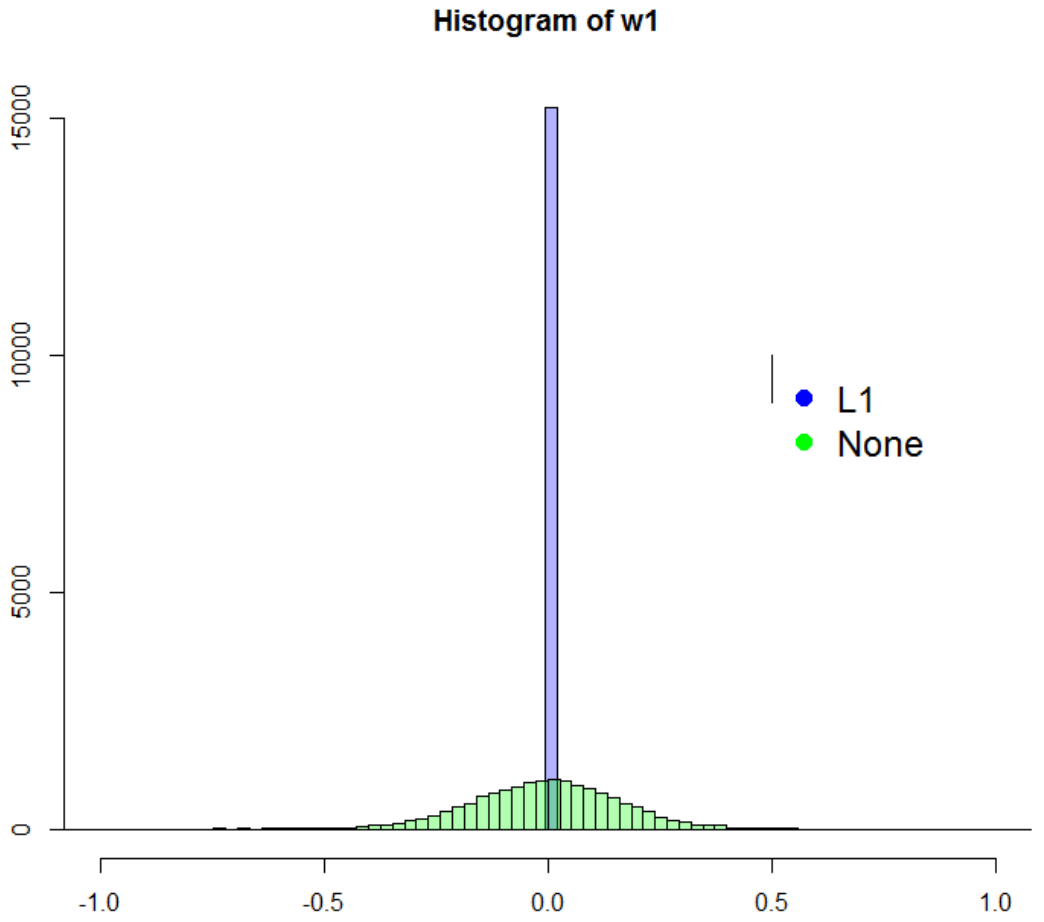


Regularization

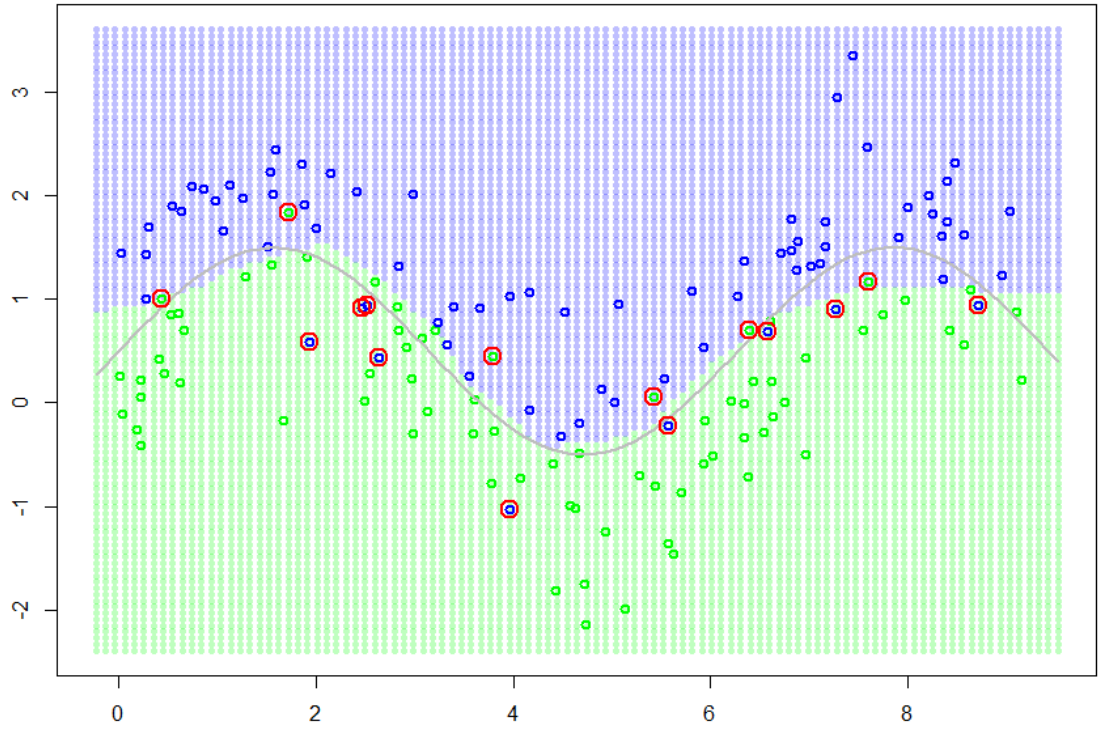


L1 .025

Regularization

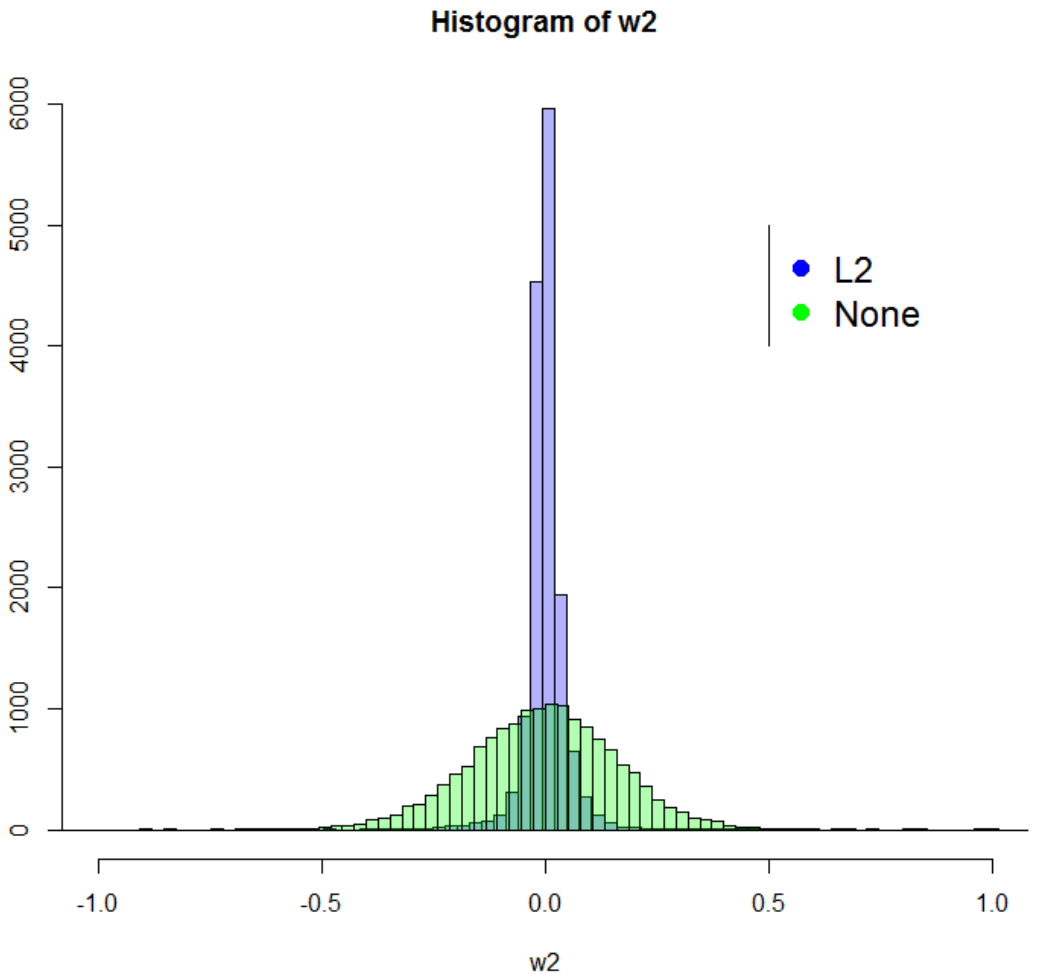


Regularization



L2 .05

Regularization



Bias/Variance

25 high-variance models averaged