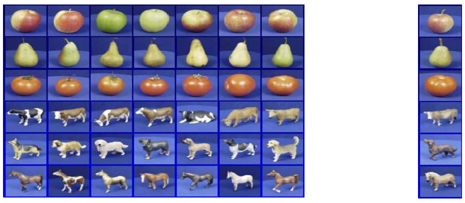
Overfitting and Underfitting

Generalization



Training set (labels known)

Test set (labels unknown)

- How well does a learned model generalize from the data it was trained on to a new test set?

Slide credit: L. Lazebnik

No Free Lunch Theorem

Slide credit: D. Hoiem

"No Free Lunch" Theorems

$Acc_G(L)$ = Generalization accuracy of learner L
= Accuracy of L on non-training examples
$\mathcal{F}$ = Set of all possible concepts, $y = f(\mathbf{x})$

Theorem: For any learner L, $\frac{1}{|\mathcal{F}|}\sum_{\mathcal{F}} Acc_G(L) = \frac{1}{2}$
(given any distribution $\mathcal{D}$ over $\mathbf{x}$ and training set size n)

Proof sketch: Given any training set S:
For every concept f where $Acc_G(L) = \frac{1}{2} + \delta$,
there is a concept f' where $Acc_G(L) = \frac{1}{2} - \delta$.
$\forall \mathbf{x} \in S, f'(\mathbf{x}) = f(\mathbf{x}) = y$. $\quad \forall \mathbf{x} \notin S, f'(\mathbf{x}) = \neg f(\mathbf{x})$.

Corollary: For any two learners L_1, L_2:
If $\exists$ learning problem s.t. $Acc_G(L_1) > Acc_G(L_2)$
Then $\exists$ learning problem s.t. $Acc_G(L_2) > Acc_G(L_1)$

Don't expect your favorite learner to always be best

Try different approaches and compare

Bias and Variance

- Bias – error caused because the model can not represent the concept
- Variance – error caused because the learning algorithm overreacts to small changes (noise) in the training data

TotalLoss = Bias + Variance (+ noise)

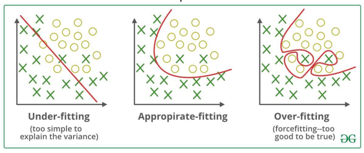
Bias-Variance Trade-off

E(MSE) = noise² + bias² + variance

Unavoidable error

Error due to incorrect assumptions

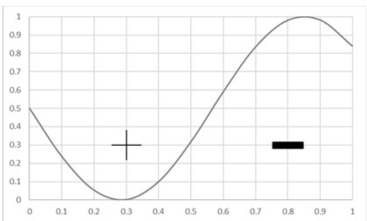
Error due to variance of training samples



See the following for explanations of bias-variance (also Bishop's "Neural Networks" book):
- http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf

Visualizing Bias

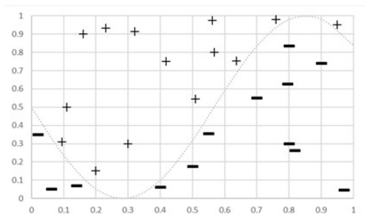
- Goal: produce a model that matches this concept



True Concept

Visualizing Bias

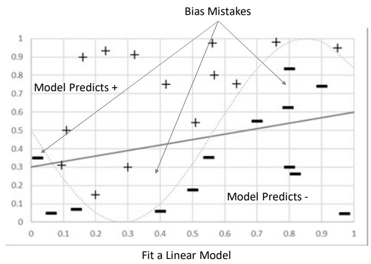
- Goal: produce a model that matches this concept
- Training Data for the concept



Training Data

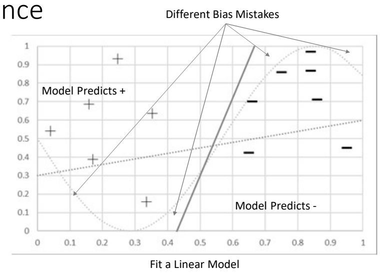
Visualizing Bias

- Goal: produce a model that matches this concept
- Training Data for concept
- Bias: Can't represent it…



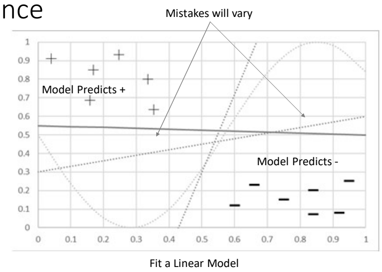
Visualizing Variance

- Goal: produce a model that matches this concept
- New data, new model

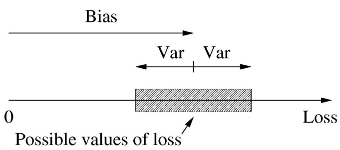


Visualizing Variance

- Goal: produce a model that matches this concept
- New data, new model
- New data, new model…
- Variance: Sensitivity to changes & noise

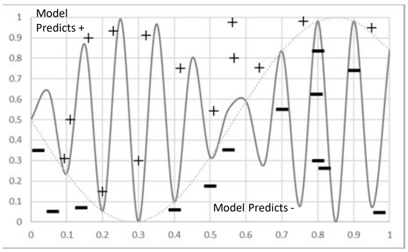


Another way to think about Bias & Variance



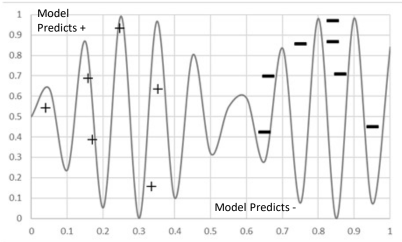
Bias and Variance: More Powerful Model

- Powerful Models can represent complex concepts
- No Mistakes!



Bias and Variance: More Powerful Model

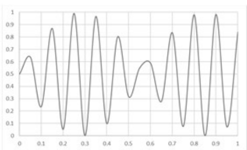
- But get more data…
- Not good!



Overfitting vs Underfitting

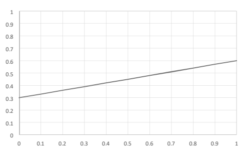
Overfitting

- Fitting the data too well
  - Features are noisy / uncorrelated to concept
  - Modeling process very sensitive (powerful)
  - Too much search

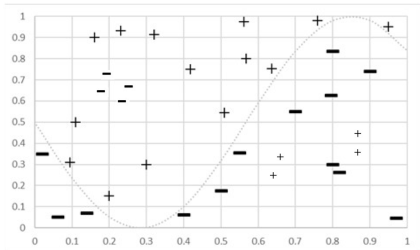


Underfitting

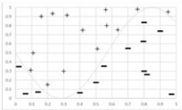
- Learning too little of the true concept
  - Features don’t capture concept
  - Too much bias in model
  - Too little search to fit model

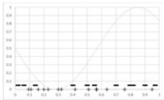


The Effect of Noise



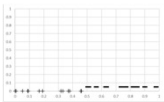
The Effect of Features





Throw out x_2

- Not much info
- Won't learn well
- Powerful -> high variance



New x_3

- Captures concept
- Simple model -> low bias
- Powerful -> low variance

The Power of a Model Building Process

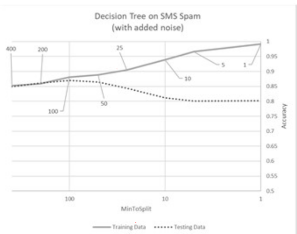
Weaker Modeling Process (higher bias)

- Simple Model (e.g. linear)
- Fixed sized Model (e.g. fixed # weights)
- Small Feature Set (e.g. top 10 tokens)
- Constrained Search (e.g. few iterations of gradient descent)

More Powerful Modeling Process (higher variance)

- Complex Model (e.g. high order polynomial)
- Scalable Model (e.g. decision tree)
- Large Feature Set (e.g. every token in data)
- Unconstrained Search (e.g. exhaustive search)

Example of Under/Over-fitting



Ways to Control Decision Tree Learning

- Increase minToSplit
- Increase minGainToSplit
- Limit total number of Nodes
- Penalize complexity

$$Loss(S) = \sum_{i}^{n} Loss(\hat{y_i}, y_i) + \alpha\, Log_2(\#\, Nodes)$$

Ways to Control Logistic Regression

- Adjust Step Size
- Adjust Iterations / stopping criteria of Gradient Descent
- Regularization

$$Loss(S) = \sum_{i}^{n} Loss(\hat{y_i}, y_i) + \alpha \sum_{j}^{\#\,Weights} |w_j|$$

Modeling to Balance Under & Overfitting

- Data
- Learning Algorithms
- Feature Sets
- Complexity of Concept
- Search and Computation
- Parameter sweeps!

Parameter Sweep

```
# optimize first parameter
for p in [ setting_certain_to_underfit, …, setting_certain_to_overfit]:
    # do cross validation to estimate accuracy

    # find the setting that balances overfitting & underfitting

# optimize second parameter
    # etc…

# examine the parameters that seem best and adjust whatever you can…
```

Types of Parameter Sweeps

- Optimize one parameter at a time
  - Optimize one, update, move on
  - Iterate a few times
- Gradient descent on meta-parameters
  - Start somewhere ‘reasonable’
  - Computationally calculate gradient wrt change in parameters
- Grid
  - Try every combination of every parameter
- Quick vs Full runs
  - Expensive parameter sweep on ‘small’ sample of data (e.g. grid)
  - A bit of iteration on full data to refine
- Intuition & Experience
  - Learn your tools
  - Learn your problem

Summary of Overfitting and Underfitting

- Bias / Variance tradeoff a primary challenge in machine learning
- Internalize: More powerful modeling is not always better
- Learn to identify overfitting and underfitting
- Tuning parameters & interpreting output correctly is key