Machine Learning Systems Design

Modeling Pipeline

Lecture 11: Model Development and Training

CE 40959 Spring 2023
Ali Zarezade
SharifMLSD.github.io

Agenda

1. Before Start
2. End-to-End vs Pipeline
3. Error Analysis
4. **Bias-Variance Tradeoff**
5. **Learning Curves**
6. **Human Performance Level**
7. **Learning under Distribution Mismatch**
8. **Debugging Inference Algorithms**
9. **Ensembles**

4. Bias-Variance Tradeoff

Bias and Variance: The two big sources of error

Suppose your training, dev and test sets all come from the same distribution.

More training data to improve performance?

Bias and Variance: The two big sources of error

More data can’t hurt.

But could be a waste of time.

How do you decide when to add data, and when not to bother?

Bias and Variance: The two big sources of error

Suppose:

- Hope to build a cat recognizer that has 5% error.
- Training set error rate: 15%, dev set error rate: 16%

The first problem to solve:

- Improve the performance on the training set.

Bias and Variance: The two big sources of error

Suppose

- Hope to build a cat recognizer that has 5% error.
- Training set error rate: 15%, dev set error rate: 16%

Dev/test set performance is usually worse than the training set performance.

If accuracy on the examples your algorithm has seen is 85%,

There’s no way getting 95% accuracy on examples the algorithm hasn’t even seen.

Bias and Variance: The two big sources of error

- **Bias:** the algorithm's error rate on the training set. 15%
- **Variance:** how much worse the algorithm does on the dev (or test) set than the training set. 1% (Generalization)

Bias and Variance: The two big sources of error

- **Bias:** Error between average model prediction and ground truth
- **Variance:** Average variability in the model prediction for the given dataset

$$Bias = \mathbb{E}_D[\hat{f}(x)] - f(x)$$

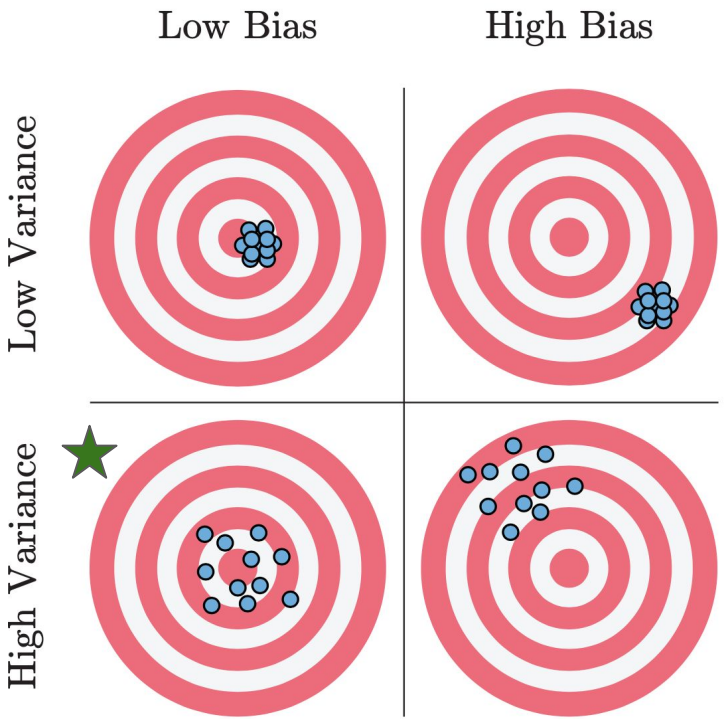
$$Variance = \mathbb{E}_D[(\hat{f}(x) - E[\hat{f}(x)])^2]$$

Examples of Bias and Variance

Training error = 1%, Dev error = 11%

Bias = 1%, Variance = 10%

- High Variance
- Fails to generalize
- Overfitting

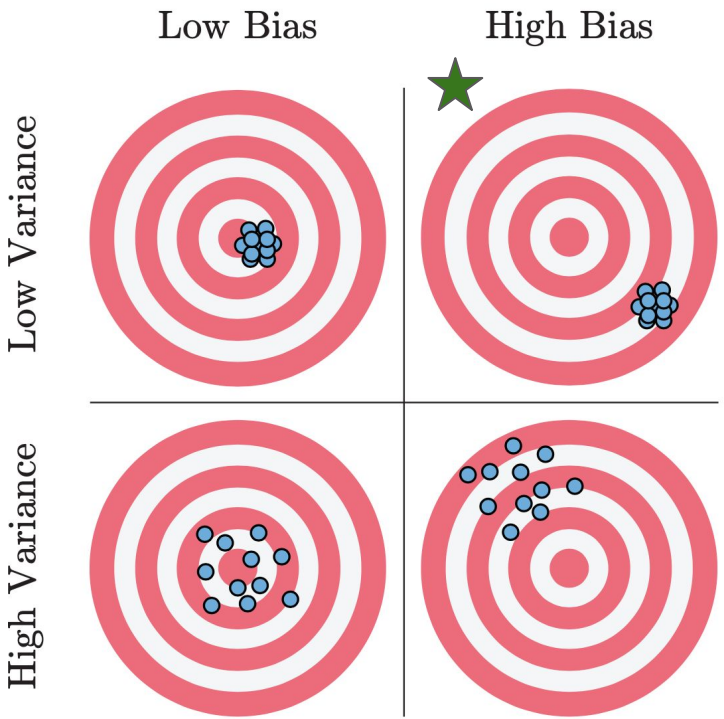


Examples of Bias and Variance

Training error = 15%, Dev error = 16%

Bias = 15%, Variance = 1%

- High Bias
- Underfitting

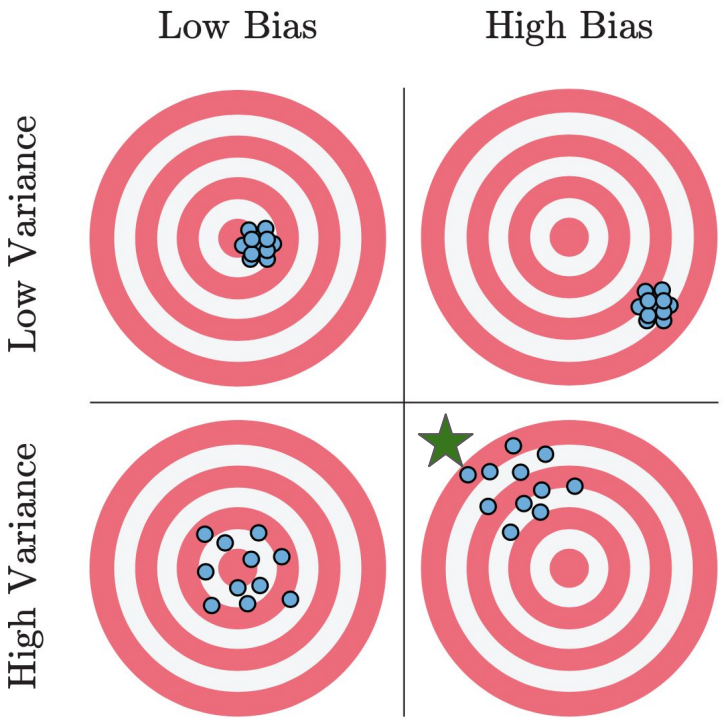


Examples of Bias and Variance

Training error = 15%, Dev error = 30%

Bias = 15%, Variance = 15%

- High Bias
- High Variance
- Underfitting
- Overfitting?

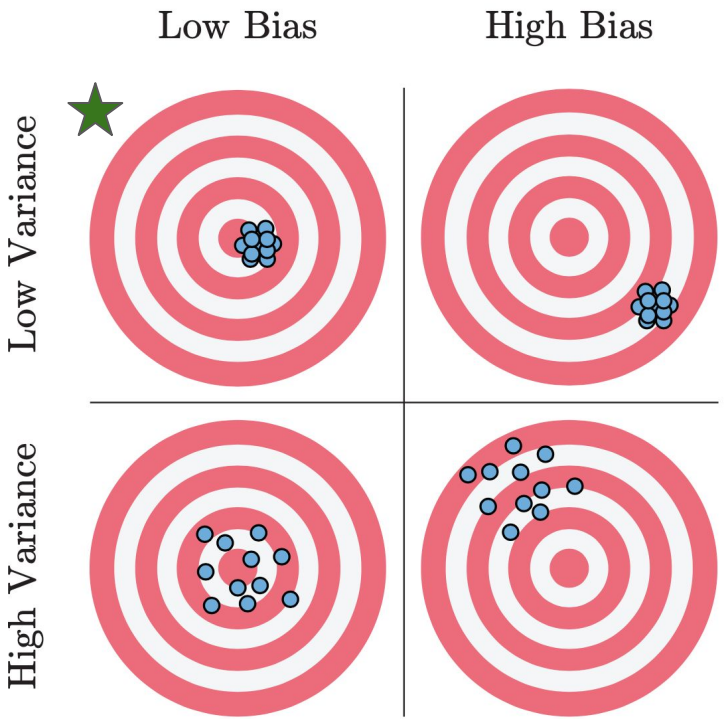


Examples of Bias and Variance

Training error = 0.5%, Dev error = 1%

Bias = 0.5%, Variance = 0.5%

- Low Bias
- Low Variance



Comparing to the optimal error rate

- Cat recognition example:
  - The “ideal” error rate is nearly 0%.

- Speech recognition system with 14% of the audio clips have so much background noise or are so unintelligible even a for human:
  - The most “optimal” speech recognition system might have error around 14%.

Comparing to the optimal error rate

Optimal error rate = 14%, Training error = 15%, Dev error = 30%

Bias = ?, Variance = 15%

- Not much room for bias or training set performance.
- This algorithm is not generalizing well to the dev set
- Thus there is ample room for the errors due to variance.

Optimal error rate is also called **Bayes error rate**

Comparing to the optimal error rate

Optimal error rate = 14%, Training error = 15%, Dev error = 30%

Unavoidable bias = 14%, Avoidable bias = 1%, Variance = 15%

Focus on variance!

- All variance is “avoidable” with a sufficiently large dataset.
- Optimal error rate is also called Bayes error rate , or Bayes rate.

Comparing to the optimal error rate

How to estimate the optimal error rate:

- For tasks that humans are reasonably good
  - Human-Level Performance
- Problem that even humans have a hard time solving (e.g., movie or ad recommender)
  - Hard to estimate.

Addressing Bias and Variance

How to solve these issues in models:

- High avoidable bias
- High variance

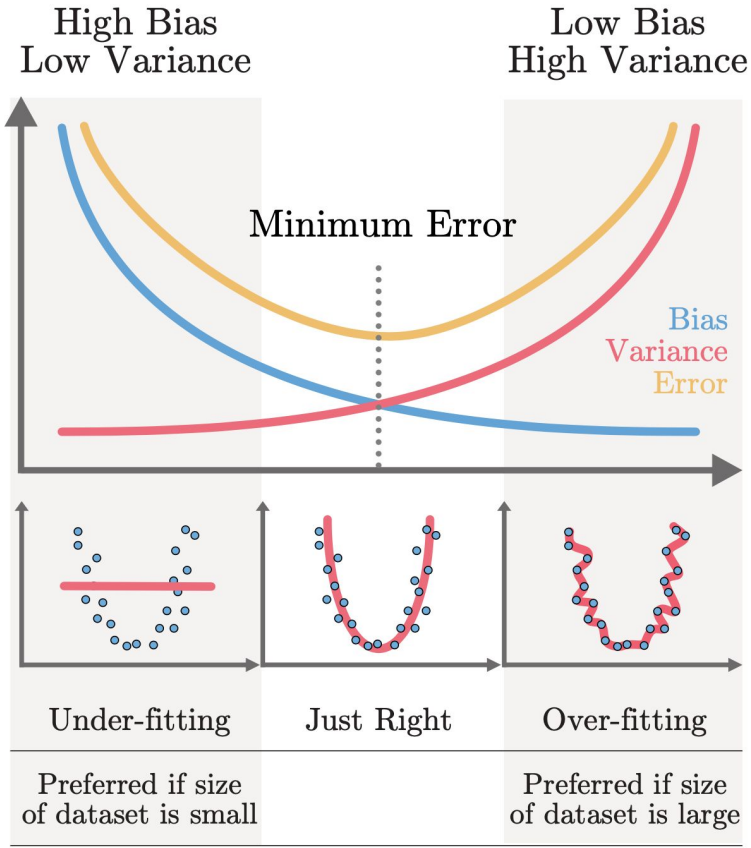
Addressing Bias and Variance

Simplest formula:

- High avoidable bias:
  - Increase the size of your neural network by adding layers/neurons.
  - **but** you may have computational problems in training and inference.
  - **but** could increase variance

- High variance:
  - add data to your training set.
  - **but** you may have a finite number of cat pictures!
  - add regularizer
  - **but** may increase biase

Bias and variance tradeoff



Bias and variance tradeoff

In the modern era (deep learning + big data), there is less of a tradeoff, and there are now more options for reducing bias without hurting variance, and vice versa.

For example, you can usually **increase a neural network size** and tune the **regularization** method to reduce bias without noticeably increasing variance. By **adding training data**, you can also usually reduce variance without affecting bias.

Techniques for reducing avoidable bias

- Increase the model size.
  - If variance increased, then use regularization.
- Modify input features based on insights from error analysis.
  - Error analysis inspires you to create additional features that help the algorithm eliminate a particular category of errors.
  - adding more features could increase the variance; but if you find this to be the case, then use regularization, which will usually eliminate the increase in variance.
- Reduce or eliminate regularization
  - L2, L1, dropout
- Modify model architecture
  - More suitable for the problem.
  - Can affect both bias and variance.
- Add more training data.
  - Usually not working!

Techniques for reducing variance

- Add more training data
  - The simplest and most reliable.
  - If you have access to significantly more data and enough computational power.
- Add regularization
  - L2, L1, dropout
  - But Increases bias.
- Add early stopping
  - But Increases bias.
  - Is like regularization methods.

Techniques for reducing variance

- Feature selection to decrease number/type of input features
  - Might increase bias.
  - From 1,000 features to 100—a 10x reduction can be effective. So long as too many useful features are not excluded.
  - In modern deep learning, with plentiful data, there is a shift away from feature selection
  - But when your training set is small, feature selection can be very useful.
- Decrease the model size
  - Use with caution, possibly increasing bias.
  - I don’t recommend for addressing variance! Regularization is better.
  - Good when computational cost and quick training is important.

Techniques for reducing variance

Additional tactics

- Modify input features based on insights from error analysis
  - Additional features that help the algorithm to eliminate a particular category of errors.
  - Can affect both bias and variance.
- Modify model architecture
  - So that it is more suitable for your problem.
  - Can affect both bias and variance.

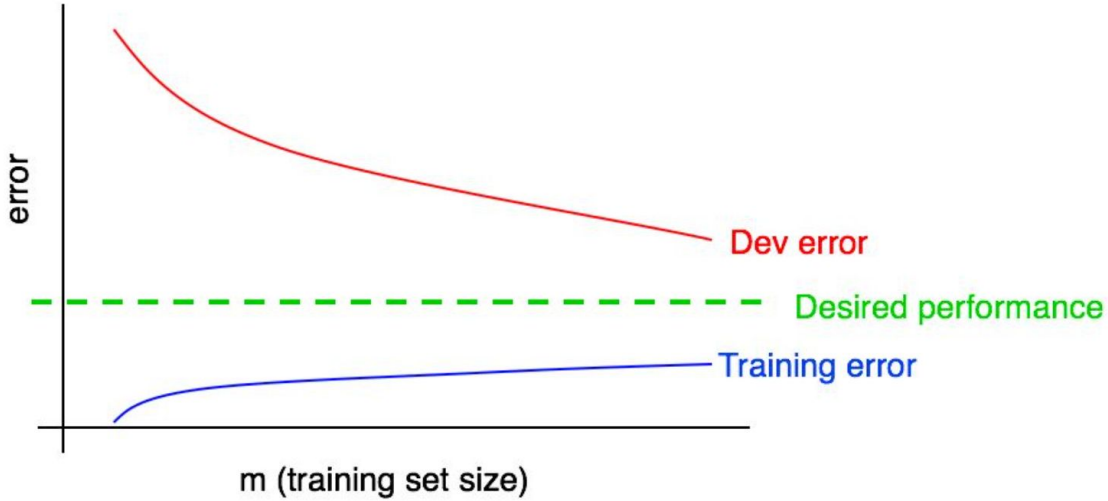
5. Learning Curves

Training and dev curves

In a typical ML model, how do you expect the error on the train and dev sets to change as you increase the size of the train data for a typical ML model?

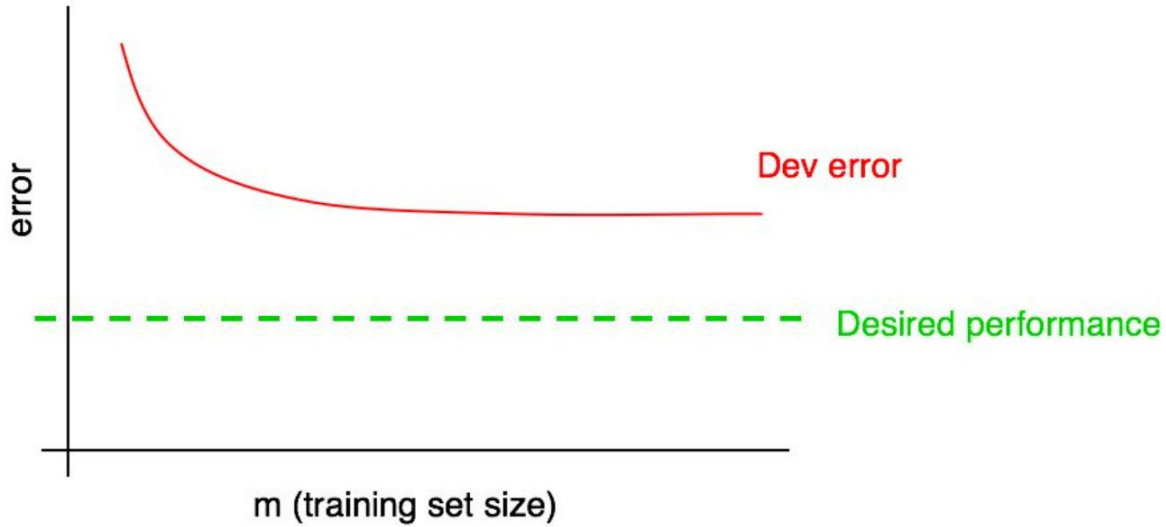
Training and dev curves

In a typical ML model, how do you expect the error on the train and dev sets to change as you increase the size of the train data for a typical ML model?



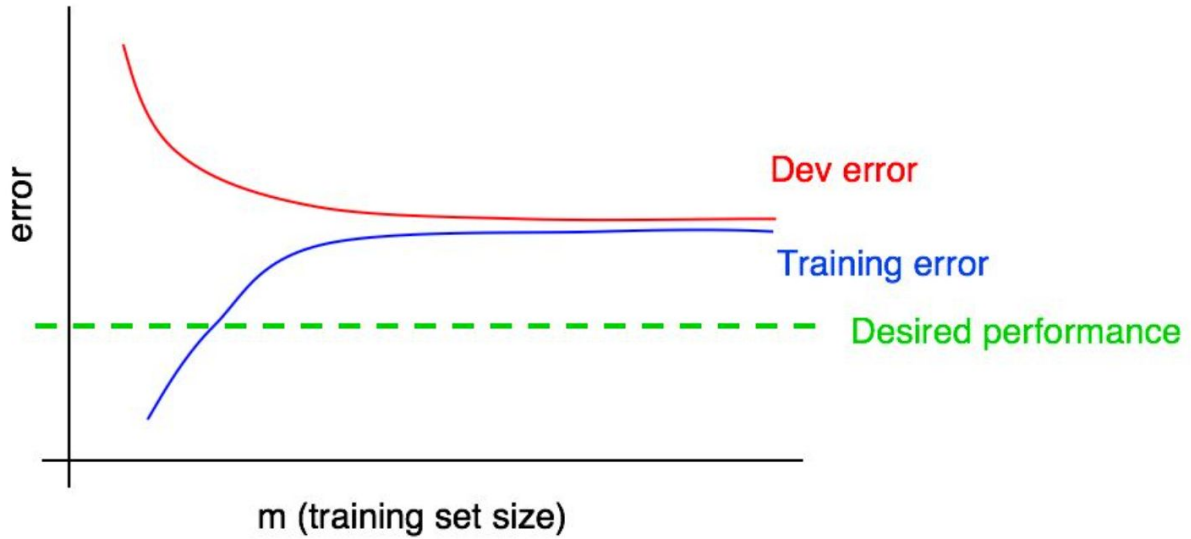
Interpreting learning curves

Interpret this curve:



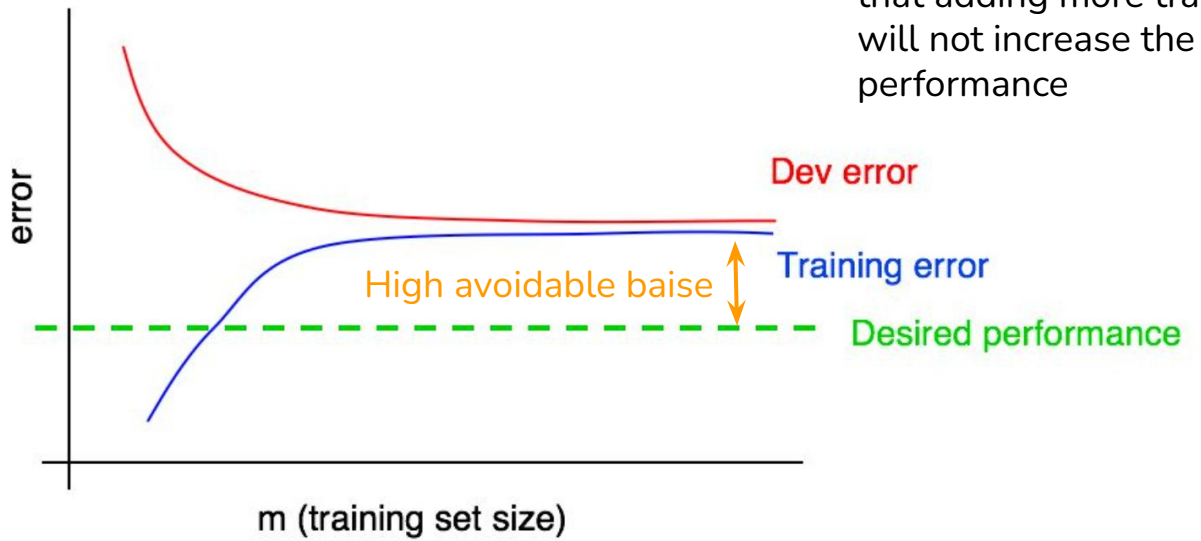
Interpreting learning curves

Interpret this curve:



Interpreting learning curves

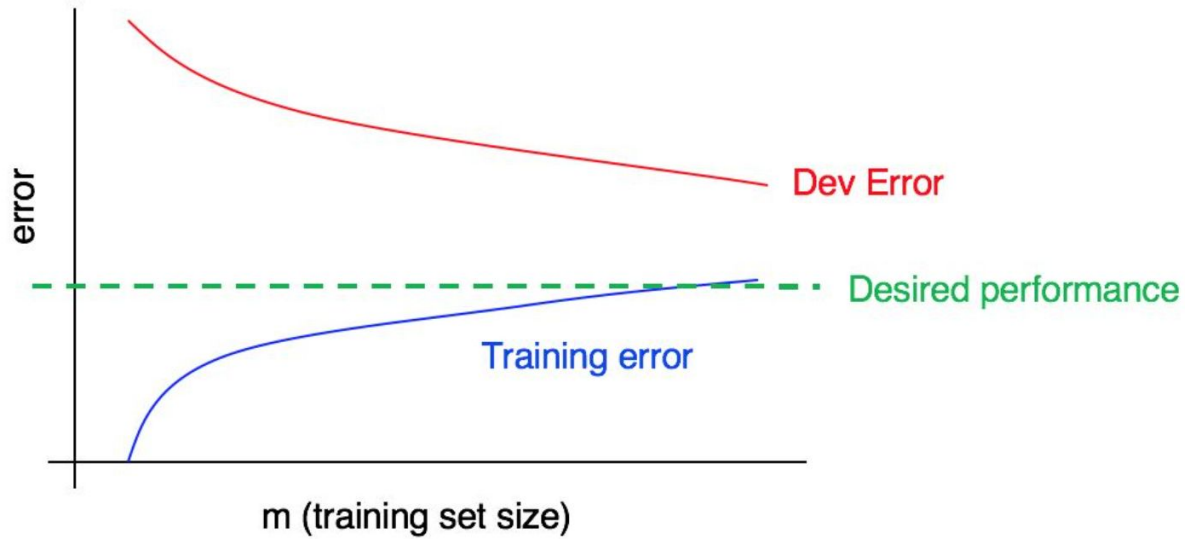
Interpret this curve:



Now we are more certain that adding more train data will not increase the model performance

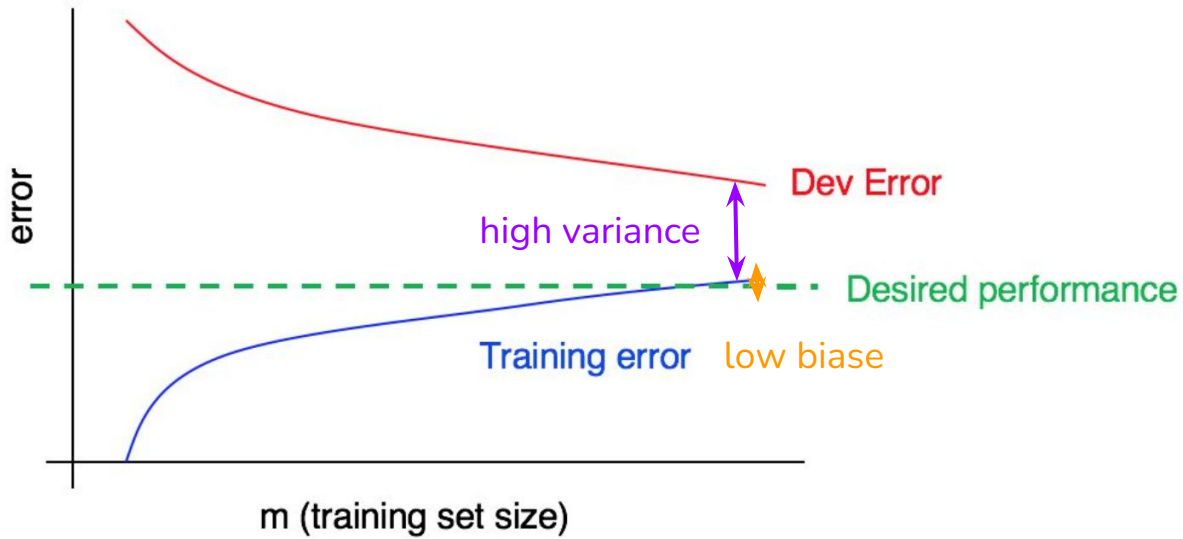
Interpreting learning curves

Consider this learning curve:



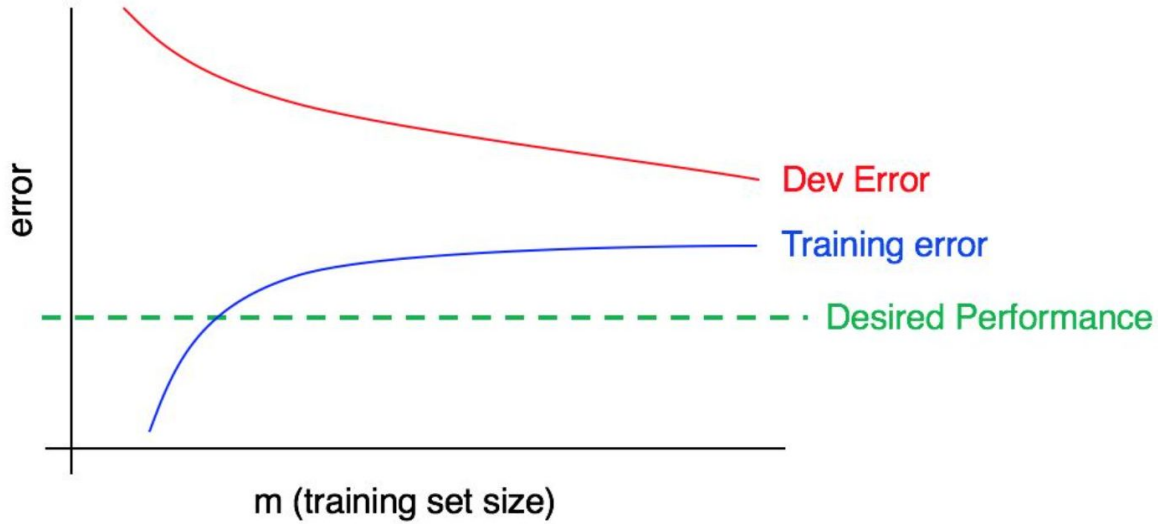
Interpreting learning curves

Consider this learning curve:



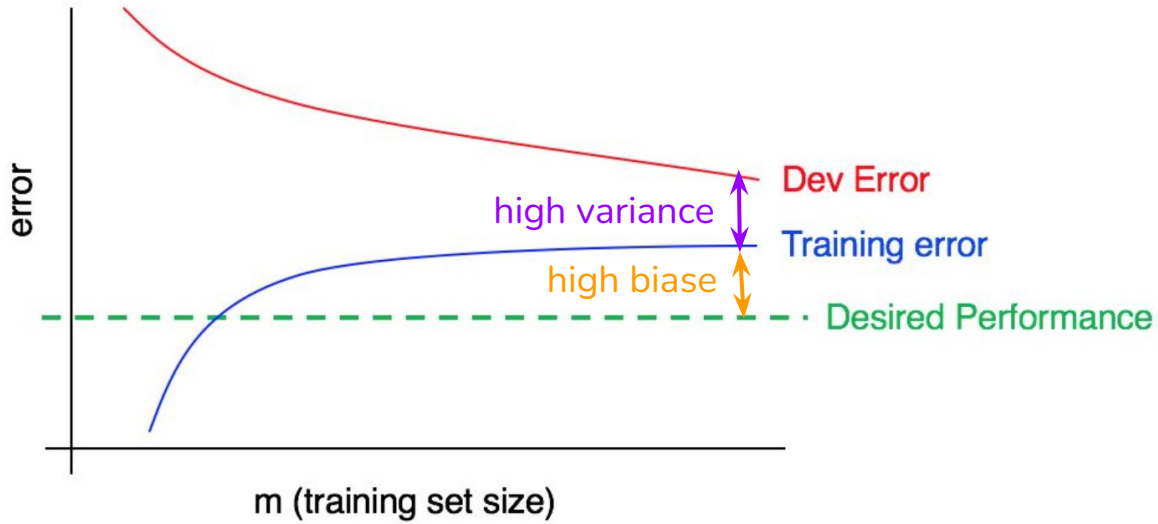
Interpreting learning curves

Now we add more training data:



Interpreting learning curves

Now we add more training data:

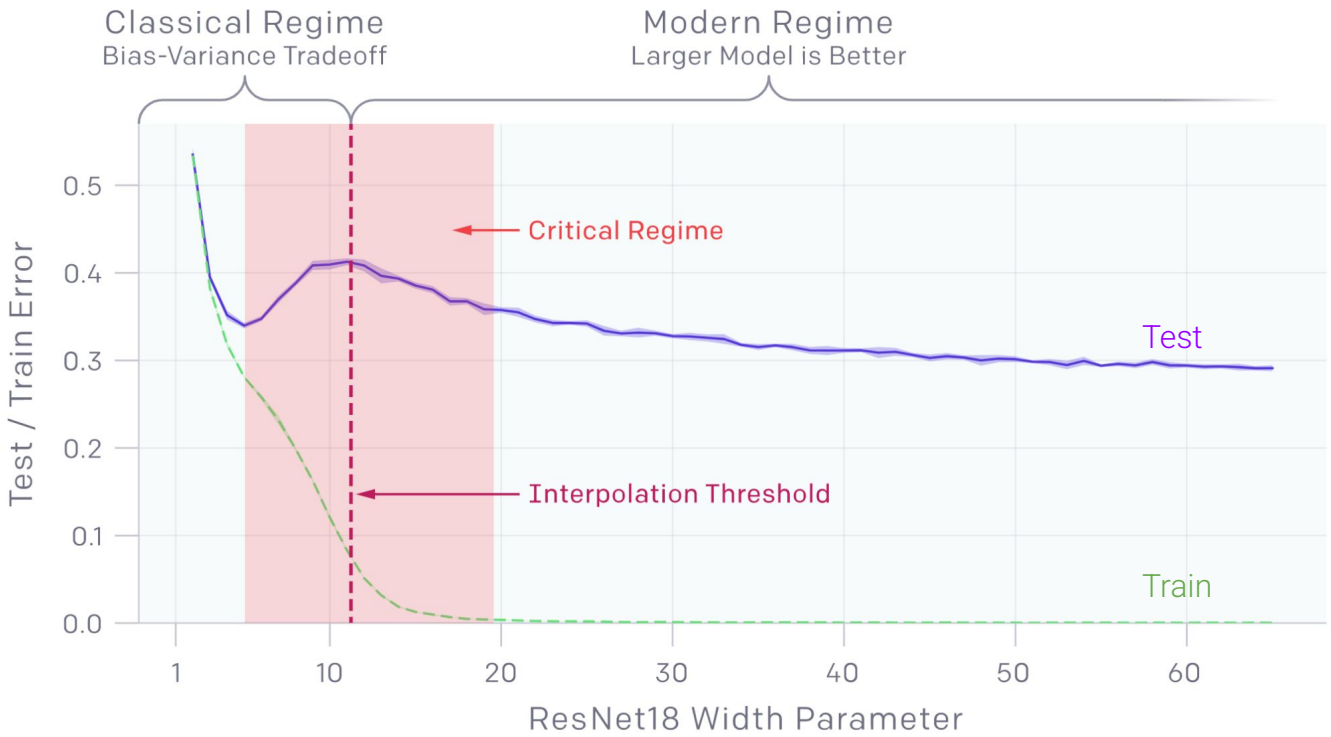


Interpreting learning curves

If the noise in the training curve makes it hard to see the true trends, you can:

- Train multiple models on different subsets of data and plot their average errors on the training and dev sets.
- If your training set is skewed towards one class, or if it has many classes, choose a “balanced” subset instead of 10 training examples at random out of the set of 100.

Double descent phenomenon



Check OpenAI [blog post]

6. Human Performance Level

Why compare to HPL?

Use human-level performance to estimate the optimal error rate and also set a “desired error rate.”

How to define HPL?

Suppose you are working on a medical imaging application that automatically makes diagnoses from x-ray images.

- typical person: 15% error
- junior doctor: 10% error
- experienced doctor: 5% error

What is the HPL?

Surpassing HPL

You are working on speech recognition and have a dataset of audio clips. Suppose HPL is 10% error and your system already achieves 8% error.

Can you continue the progress!?

Surpassing HPL

Yes! If you can identify a subset of data in which humans significantly surpass your system.

For example, your system may be much better than people at recognizing speech in noisy audio, but humans are still better at transcribing very rapidly spoken speech.

7. Learning under Distribution Mismatch

When you should train and test on different dists?

Users of your cat pictures **app** have uploaded **10,000** images, which you have manually labeled.

You also have a larger set of **200,000** images that you downloaded off the **internet**.

How should you define train/dev/test sets?

When you should train and test on different dists?

Most of the **academic** literature on machine learning assumes that the **training** set, **dev** set and **test** set all come from the **same distribution**.

It was okay in the early days of machine learning, where data was scarce. But in the **era of big data**, we now have access to huge training sets, such as cat internet images.

But, the **dev** and **test** set distribution should come from the **same distribution** of of your app users.

So, for example, use 5K (app) samples for the dev/test set and 5K (app) + 200K (internet) samples for the train set.

How to decide whether to use all your data

Should we discard the 20,000 internet images for fear of it biasing your learning algorithm?

How to decide whether to use all your data

Probably not! In the modern era of powerful, flexible learning algorithms (deep learning), as long as there is some x → y mapping that works well for both types of data, there is no need for separate models.

The effects of using all data

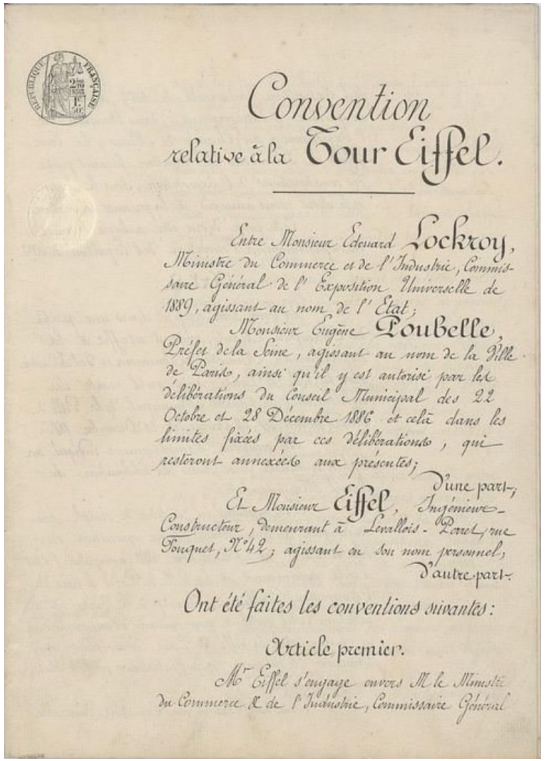
- **Transfer learning**: Your neural network can apply some of the knowledge acquired from internet images to mobile app images.
- **Biase**: if there are some properties in internet images that differ greatly from mobile app images, your model may use some of the representational capacity for it. Thus there is less capacity for recognizing data drawn from the distribution of mobile app images.

"*For every addition of knowledge, you forget something that you knew before. It is of the highest importance, therefore, not to have useless facts elbowing out the useful ones*."

Sherlock Holmes

The effects of using all data

- **Computation cost**: Even when two types of data do not compete for capacity and your algorithm's "brain" is big enough that you don't have to worry about running out of "attic space", you should just leave out that data for computational reasons if you think it has no benefit.



Convention
relative à la Tour Eiffel.

Entre Monsieur Edouard Lockroy, Ministre du Commerce et de l'Industrie, Commissaire Général de l'Exposition Universelle de 1889, agissant au nom de l'Etat;
Monsieur Eugène Poubelle, Préfet de la Seine, agissant au nom de la Ville de Paris, ainsi qu'il y est autorisé par les délibérations du Conseil Municipal des 22 Octobre et 28 Décembre 1886 et celle dans les limites fixés par ces délibérations, qui resteront annexées aux présentes;
D'une part,
Et Monsieur Eiffel, Ingénieur-Constructeur, demeurant à Levallois-Perret, rue Fouquet, N°42, agissant en son nom personnel,
D'autre part.

Ont été faites les conventions suivantes:

Article premier.

M. Eiffel s'engage envers M. le Ministre du Commerce & de l'Industrie, Commissaire Général

Do not include inconsistent data

Suppose you want to learn to predict housing prices in New York City. Given the size of a house (input feature x), you want to predict the price (target label y).

Suppose you have a second dataset of housing prices in Detroit, Michigan, where housing prices are much lower than NYC.

Should you include this data in your training set?

Do not include inconsistent data

No! In contrast to cat images example, New York City and Detroit, Michigan data are not consistent. Given the same x (size of house), the price is very different depending on where the house is.

But maybe we can still leverage Michigan data, how?

Weighting data

Remember the 40:1 ratio between the size of internet and app images datatest.

The regular learning algorithm tries to optimize:

$$\min_{\theta} \sum_{(x,y)\in \text{MobileImg}} \left(h_\theta\left(x\right) - y\right)^2 + \sum_{(x,y)\in \text{InternetImg}} \left(h_\theta\left(x\right) - y\right)^2$$

You can instead optimize with an additional parameter β (e.g 1/40 or less, **why?**):

$$\min_{\theta} \sum_{(x,y)\in \text{MobileImg}} \left(h_\theta\left(x\right) - y\right)^2 + \beta \sum_{(x,y)\in \text{InternetImg}} \left(h_\theta\left(x\right) - y\right)^2$$

Generalizing from the training set to the dev set

Suppose your training set contains Internet images + Mobile images, and the dev/test sets contain only Mobile images. However, the algorithm is not working well.

What might be wrong?

Generalizing from the training set to the dev set

Here are some possibilities of what might be wrong:

- High bias: poor performance on training set.
- High variance: good performance on training set, poor performance on unseen data from same distribution.
- Data mismatch: good performance on unseen data from same distribution, poor performance on dev/test set from different distribution.

Generalizing from the training set to the dev set

For example, suppose that humans achieve near perfect performance on the cat recognition task. Your algorithm achieves this:

- 1% error on the training set
- 1.5% error on data drawn from the same distribution as the training set that the algorithm
  has not seen
- 10% error on the dev set

What is the problem here?

Data mismatch problem

In order to diagnose to what extent an algorithm suffers from each of the problems 1-3 above, it will be useful to have another dataset, called **train dev set.**

Data mismatch problem

You now have four subsets of data:

- **Training set**: data for learning (e.g., Internet + Mobile images).
- **Training dev set**: data for tracking learning progress from same distribution as training set (e.g., Internet + Mobile images).
- **Dev set**: data for tuning model from same distribution as test set and ultimate goal (e.g., Mobile images).
- **Test set**: data for evaluating model from same distribution as dev set and ultimate goal (e.g., Mobile images).

Data mismatch problem

Armed with these four separate datasets, you can now evaluate:

- Training error, by evaluating on the training set.
- The algorithm’s ability to generalize to new data drawn from the training set distribution, by evaluating on the training dev set.
- The algorithm’s performance on the task you care about, by evaluating on the dev and/or test sets.

Identifying bias, variance, and data mismatch errors

- 1% error on the training set.
- 5% error on training dev set.
- 5% error on the dev set.

- 10% error on the training set.
- 11% error on training dev set.
- 12% error on the dev set.

- 10% error on the training set.
- 11% error on training dev set.
- 20% error on the dev set.

Identifying bias, variance, and data mismatch errors

- 1% error on the training set.
- 5% error on training dev set.
- 5% error on the dev set.

high variance

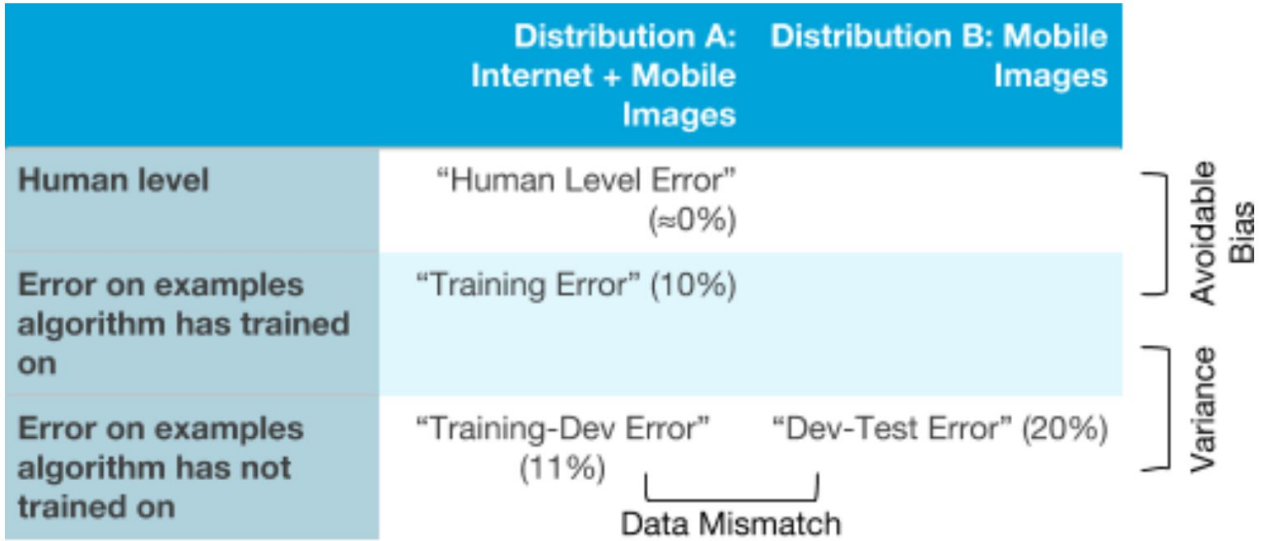
- 10% error on the training set.
- 11% error on training dev set.
- 12% error on the dev set.

high bias

- 10% error on the training set.
- 11% error on training dev set.
- 20% error on the dev set.

high bias & Data mismatch

Identifying bias, variance, and data mismatch errors



It is possible for an algorithm to suffer from any subset of these errors.

Addressing data mismatch

Suppose you have developed a speech recognition system that does very well on the training set and on the training dev set. But, it does poorly on your dev set.

What can you do with data mismatch problem?

Addressing data mismatch

You can:

1. Try to understand what properties of the data differ between the training and the dev set distributions (error analysis).
2. Try to find more training data that better matches the dev set examples that your algorithm has trouble with.

Addressing data mismatch

Or use:

- **Artificial data synthesis**: for example, obtain a large quantity of car/road noise audio clips and *synthesize* huge amounts of train data that sound as if it were collected inside a car. Another example, is to make images blurry in cat app to reduce error on blurred images.

But be careful to not introduce bias to your model!

Try to avoid giving the synthesized data properties that makes it possible for a learning algorithm to distinguish synthesized from non-synthesized examples

8. Debugging Inference Algorithms

The Optimization Verification test

Suppose you are building a speech recognition system. Your system takes an audio clip A and computes $Score_A(S)$ for each possible output sentence S. For example, you might estimate $Score_A(S) = P(S|A)$.

To find the output sentence, you need to maximize:

$$Output = \arg\max_S Score_A(S)$$

The Optimization Verification test

To search for S, If the English language has 50,000 words, then there are 50000^N possible sentences of length N.

So, one example (approximate) search algorithm is “*beam search*,” which keeps only K top candidates during the search process.

The Optimization Verification test

Suppose that an audio clip A records someone saying “I love machine learning.” But instead of outputting the correct transcription, your system outputs the incorrect “I love robots.” There are now two possibilities for what went wrong:

- **Search algorithm problem**. The approximate search algorithm (beam search) failed to find the value of S that maximizes $Score_A(S)$.
- **Objective (scoring function) problem**. Our estimates for $Score_A(S) = P(S|A)$ were inaccurate. In particular, our choice of $Score_A(S)$ failed to recognize that “I love machine learning” is the correct transcription.

The Optimization Verification test

Depending on which of these was the cause of the failure, you should prioritize your efforts very differently.

How can you decide what to work on?

The Optimization Verification test

Optimization Verification test:

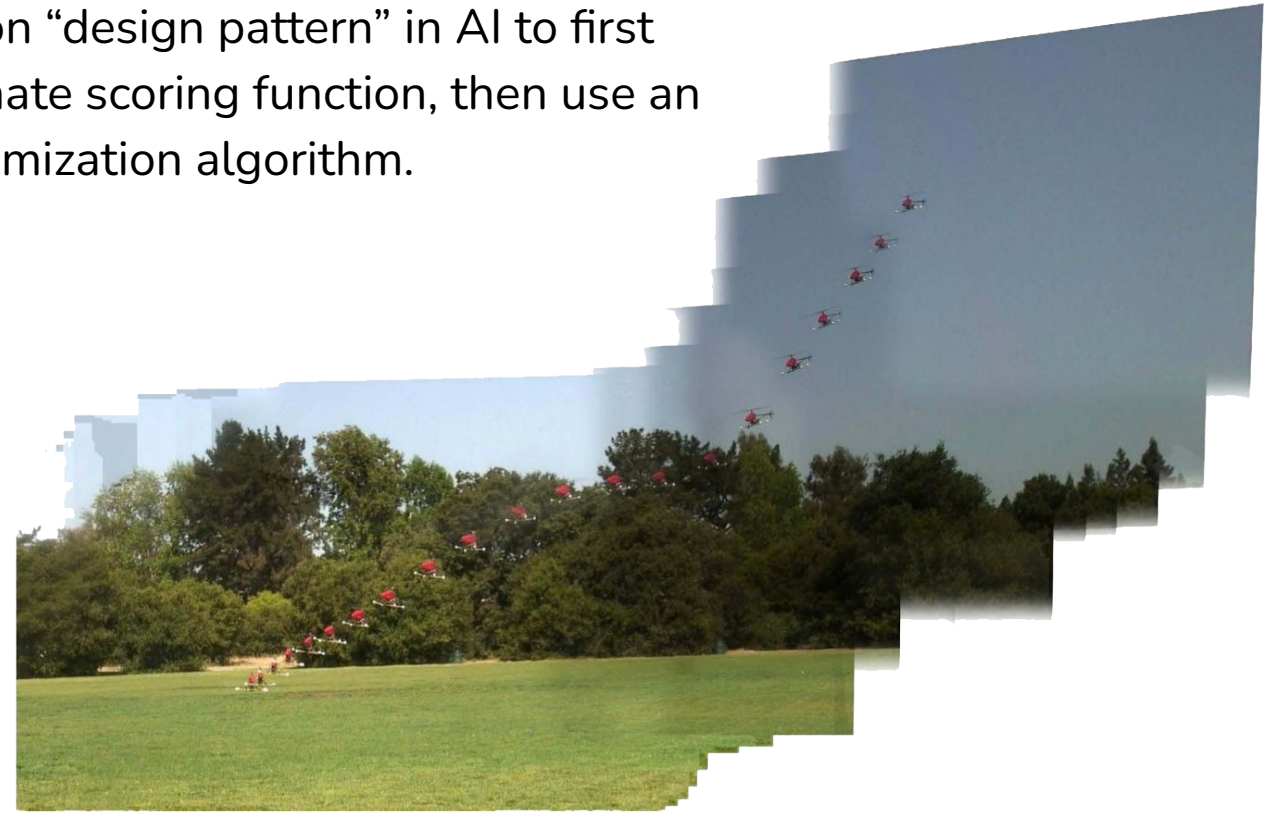
$$Score_A(S^*) > Score_A(S_{out}) \longrightarrow \quad \text{Improve search algorithm}$$

$$Score_A(S^*) \leq Score_A(S_{out}) \longrightarrow \quad \text{Improve score function}$$

in practice, you should examine the errors in your dev set, and decide based on error percentages.

General form of Optimization Verification test

It is a very common “design pattern” in AI to first learn an approximate scoring function, then use an approximate maximization algorithm.



9. Ensembles

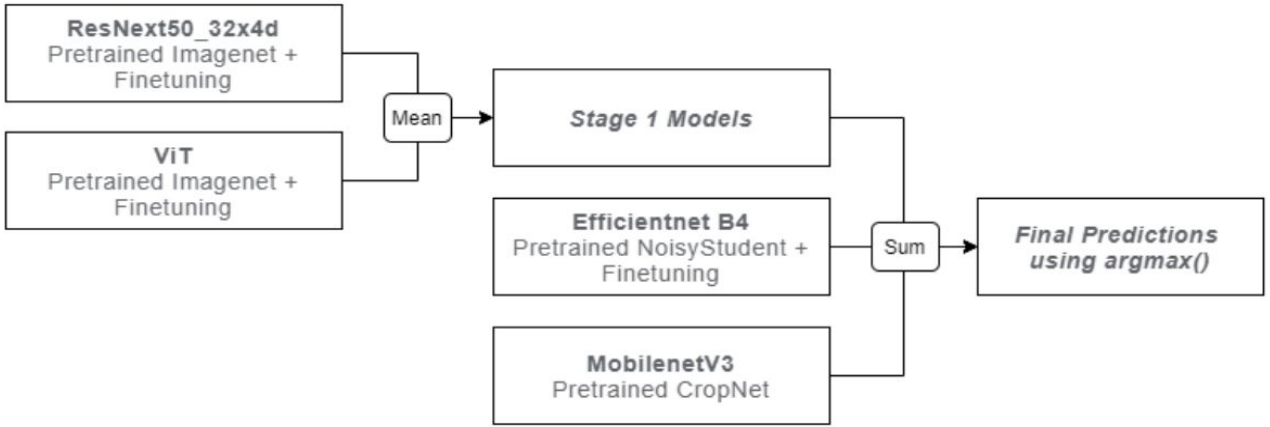
Ensemble

- Creating a strong model from an ensemble of weak models

Base learners

Ensembles: extremely common in leaderboard style projects

- 20/22 winning solutions on Kaggle in Jan - Aug 2021 use ensembles
- One solution uses 33 models!



https://farid.one/kaggle-solutions

Why does ensembling work?

- Task: email classification (SPAM / NOT SPAM)
- 3 uncorrelated models, each with accuracy of 70%
- Ensemble: majority vote of these 3 models
    - Ensemble is correct is at least 2 models are correct

Why does ensembling work?

- 3 models, each with 70% accuracy
- Ensemble is correct if at least 2 models are correct
- Probability at least 2 models are correct: 34.3% + 44.1% = 78.4%

Outputs of 3 models	**Probability**	**Ensemble's output**
All 3 are correct	0.7 * 0.7 * 0.7 = 0.343	Correct
Only 2 are correct	(0.7 * 0.7 * 0.3) * 3 = 0.441	Correct
Only 1 is correct	(0.3 * 0.3 * 0.7) * 3 = 0.189	Wrong
None is correct	0.3 * 0.3 * 0.3 = 0.027	Wrong

Why does ensembling work?

- 3 models, each with 70% accuracy
- Ensemble is correct if at least 2 models are correct
- Probability at least 2 models are correct: 34.3% + 44.1% = 78.4%

- The less correlation among base learners, the better
- Common for base learners to have different architectures

Ensemble

- Bagging
- Boosting
- Stacking

Bagging

- Sample with replacement to create different datasets
- Train a classifier with each dataset
- Aggregate predictions from classifiers
  - e.g. average, majority vote

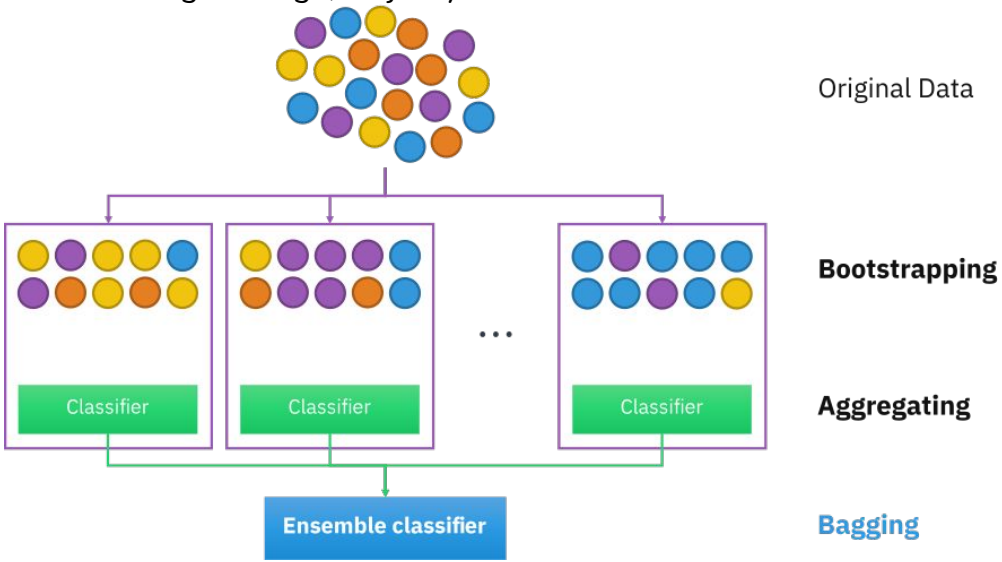
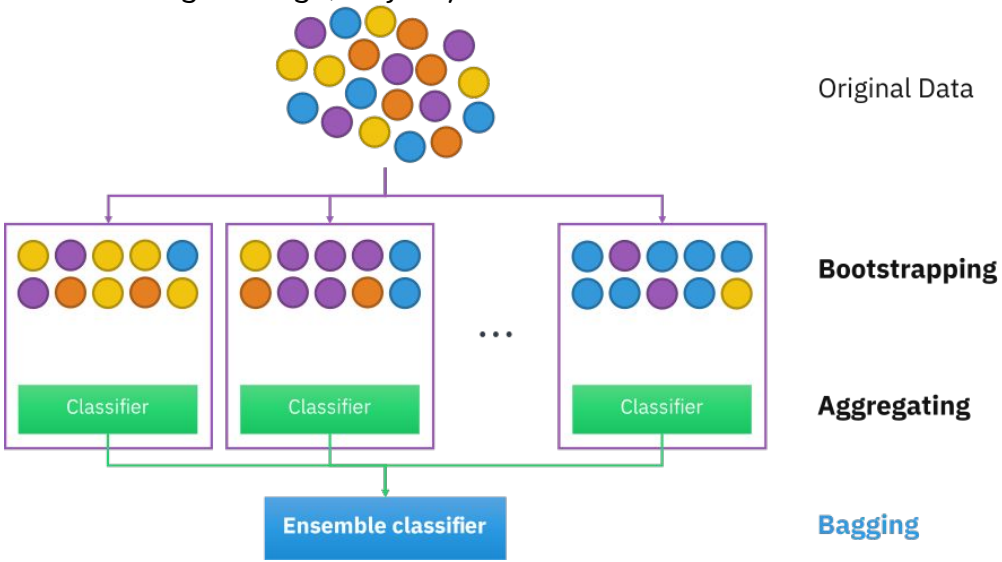


Illustration by Sirakorn

Bagging

- Sample with replacement to create different datasets
- Train a classifier with each dataset
- Aggregate predictions from classifiers
  - e.g. average, majority vote



- Generally improves unstable methods e.g. neural networks, trees
- Can degrade stable methods e.g. kNN

Bagging Predictors (Leo Breiman, 1996)

Illustration by Sirakorn

Boosting

1. Train a weak classifier
2. Give samples misclassified by weak classifier higher weight
3. Repeat (1) on this reweighted data as many iterations as needed
4. Final strong classifier: weighted combination of existing classifiers
    a. classifiers with smaller training errors have higher weights

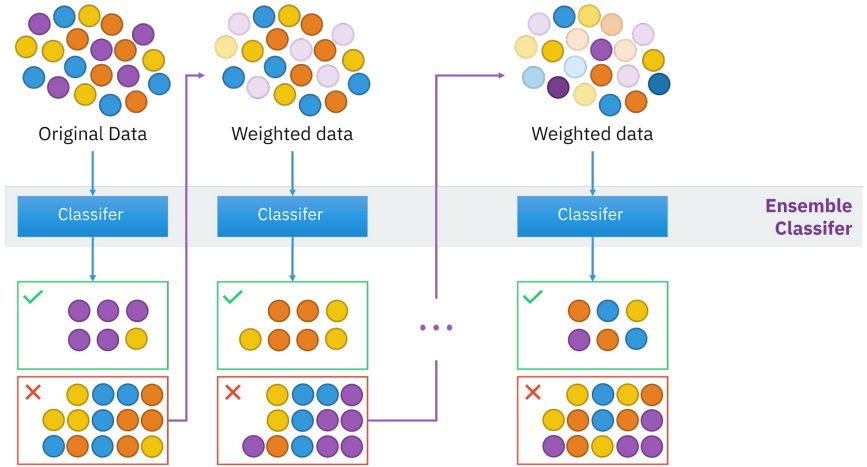
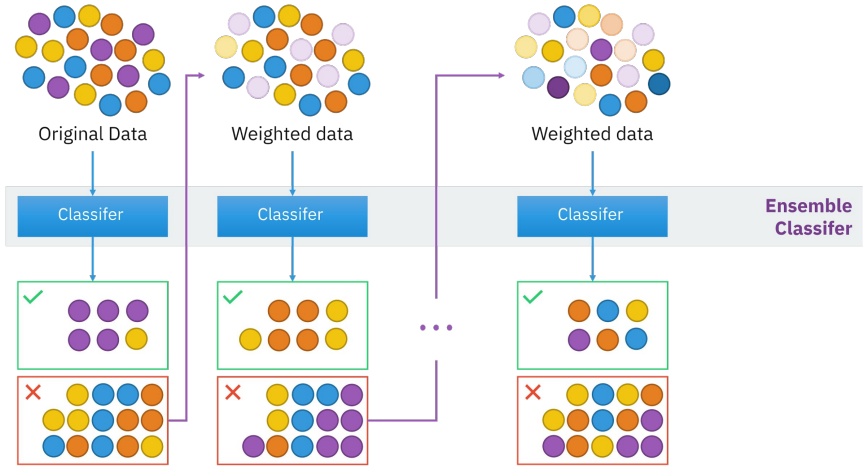


Illustration by Sirakorn

Boosting

1. Train a weak classifier
2. Give samples misclassified by weak classifier higher weight
3. Repeat (1) on this reweighted data as many iterations as needed
4. Final strong classifier: weighted combination of existing classifiers
    a. classifiers with smaller training errors have higher weights

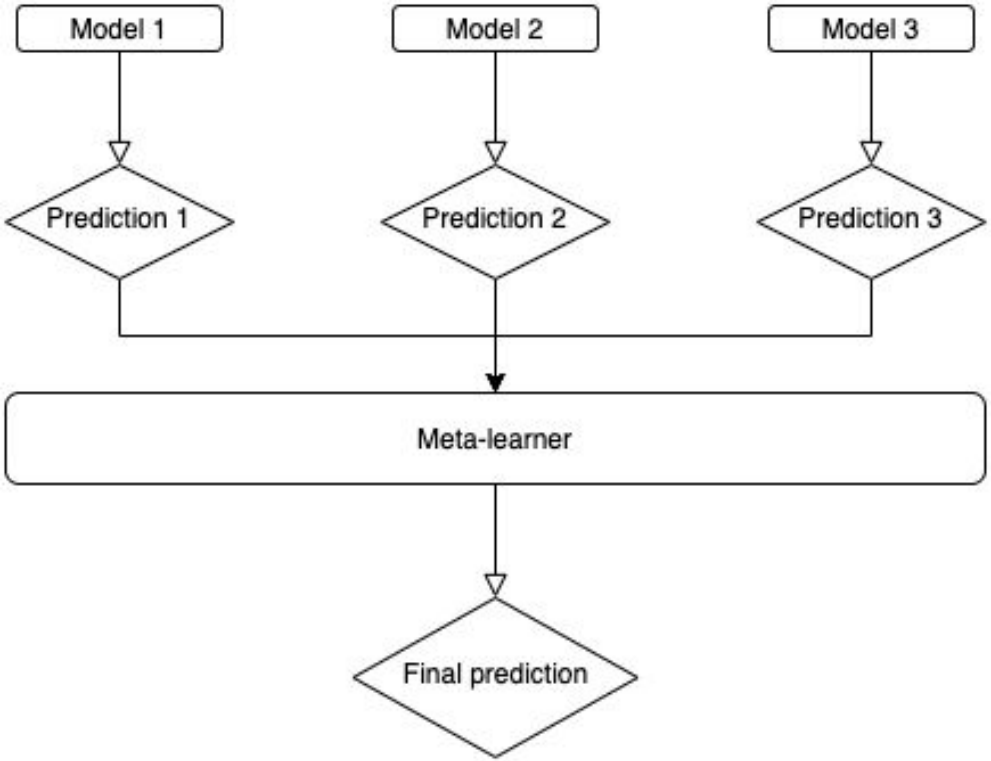


Extremely popular:

- XGBoost
- LightGBM

Illustration by Sirakorn

Stacking



Majority vote, logistic regression, simple NN

Machine Learning Systems Design

Modeling Pipeline

Next Lecture: Model Performance Analysis

CE 40959 Spring 2023
Ali Zarezade
SharifMLSD.github.io