Lecture 26: Practical Advice

COMP 343, Spring 2022
Victoria Manfredi

WESLEYAN
UNIVERSITY

Acknowledgements: These slides are based primarily on content from the book "Machine Learning" by Tom Mitchell, and on slides created by Vivek Srikumar (Utah)

Today's Topics

Project checkpoint 2

- due today

ML and the world

- Debugging machine learning
- Adding machine learning into your favorite task
- Course retrospective

Practical Advice

DEBUGGING MACHINE LEARNING

Debugging machine learning

Suppose you train a classifier for spam detection

You *obviously* follow best practices for finding hyper-parameters (such as cross-validation)

Your classifier is only 75% accurate

What can you do to improve it? (assuming that there are no bugs in the code)

Different ways to improve your model

More training data

Features

1. Use more features
2. Use fewer features
3. Use other features

Better training

1. Run for more iterations
2. Use a different algorithm
3. Use a different classifier
4. Play with regularization

Tedious!

And prone to errors, dependence on luck

Let us try to make this process more methodical

First, diagnostics

Easier to fix a problem if you know where it is

Some possible problems:

1. Over-fitting (high variance)

2. Under-fitting (high bias)

3. Your learning algorithm does not converge

4. Are you measuring the right thing?

Detecting over- or under-fitting

Over-fitting: Training accuracy is much higher than test accuracy

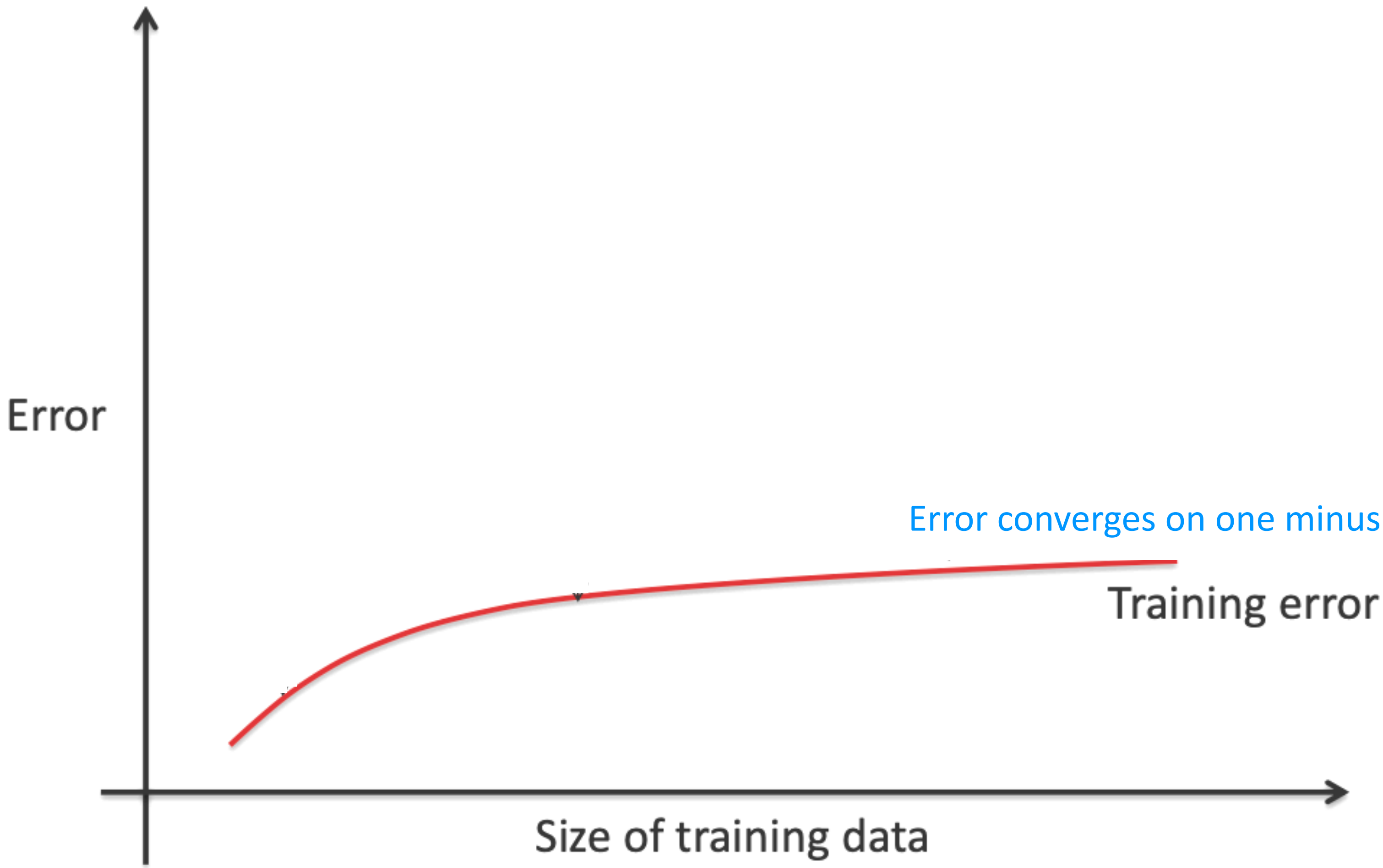
- Model explains training data very well, but poor generalization

Under-fitting: Both training and test accuracies are unacceptably low

- Model cannot represent concept well enough because hypothesis space is not big enough

How to detect? Plot training error with increasing amounts of training data

Detecting high variance using learning curves

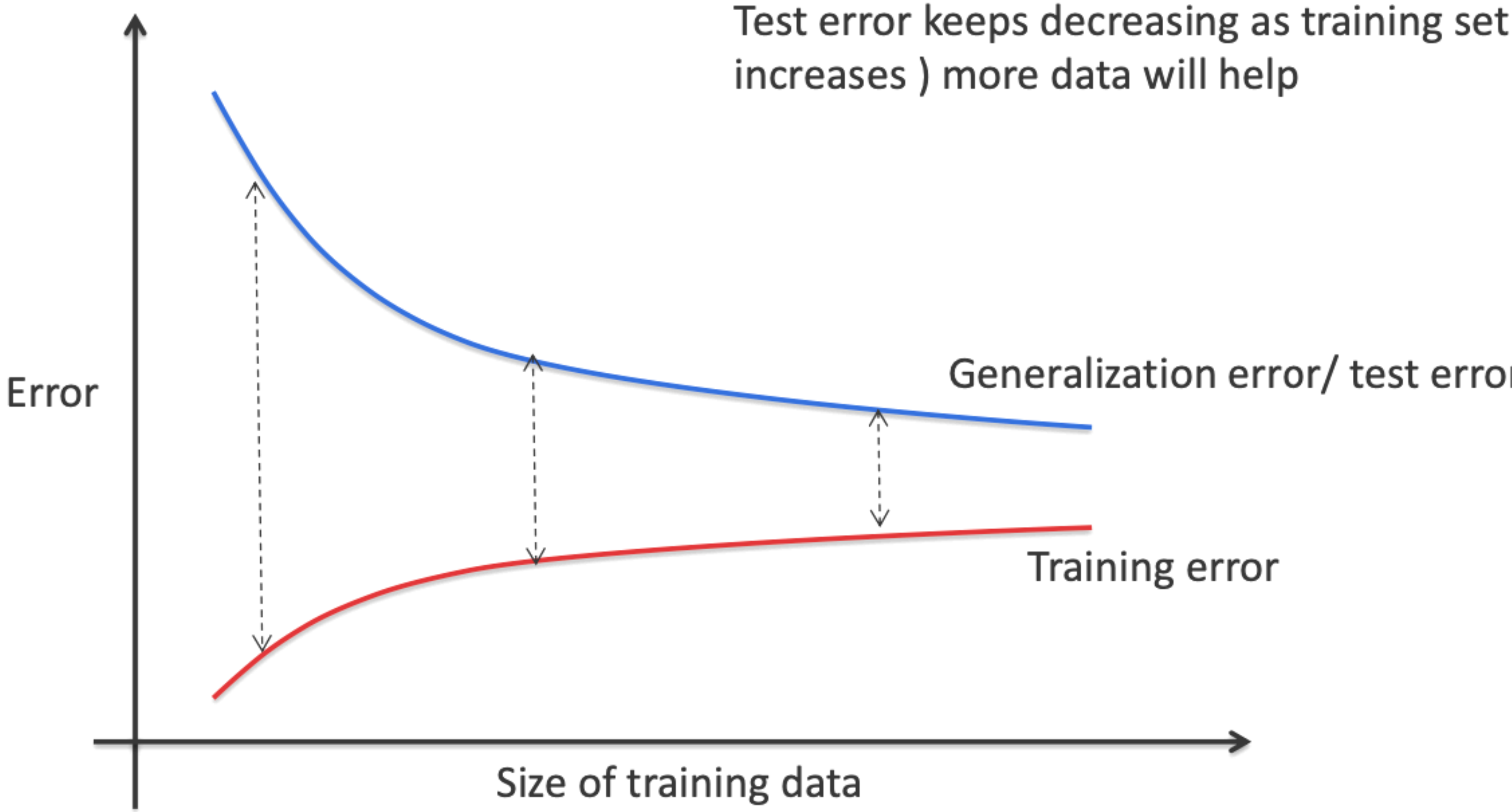


Error converges on one minus noise in data

Error is low at beginning: why?

Suppose one example: classifier
100% accurate on training set

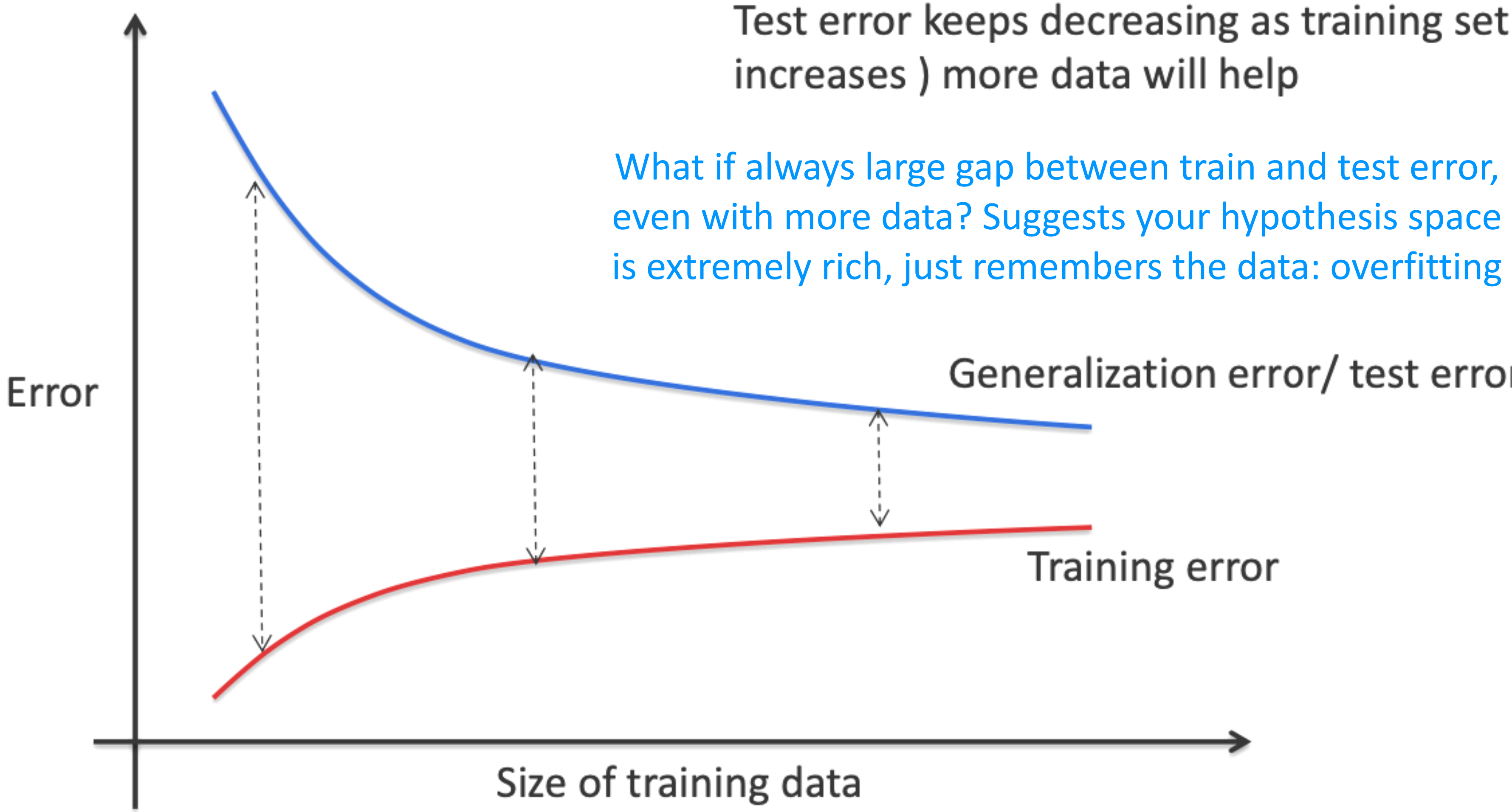
Detecting high variance using learning curves



Detecting high variance using learning curves

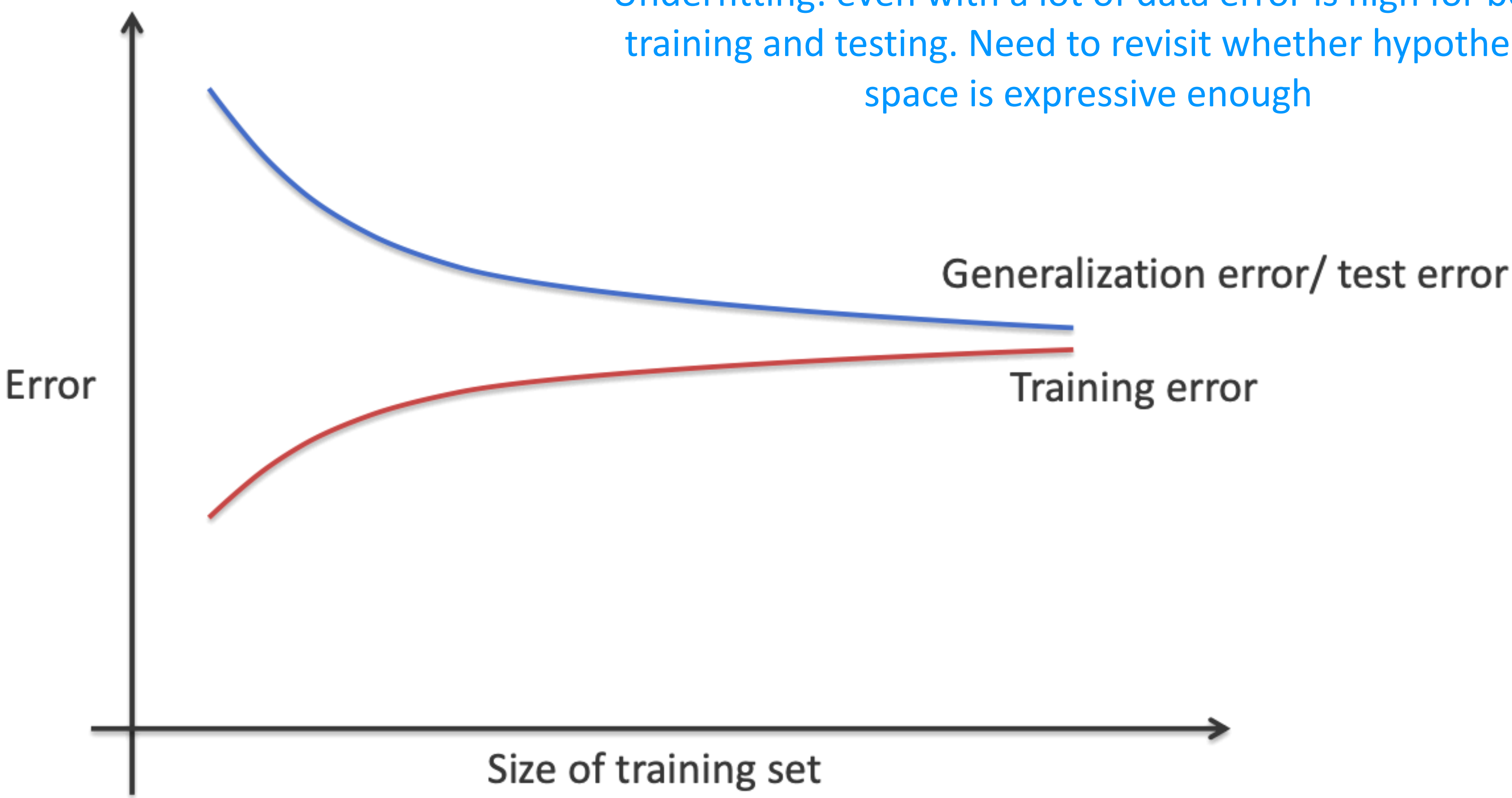
Test error keeps decreasing as training set increases) more data will help

What if always large gap between train and test error, even with more data? Suggests your hypothesis space is extremely rich, just remembers the data: overfitting



Detecting high bias using learning curves

Underfitting: even with a lot of data error is high for both training and testing. Need to revisit whether hypothesis space is expressive enough



Different ways to improve your model

More training data Helps with over-fitting

Features

1. Use more (informative) features Helps with under-fitting
2. Use fewer features (reduces expressiveness of classifier) Helps with over-fitting
3. Use other features Could help with over-fitting and under-fitting

Better training

1. Run for more iterations
2. Use a different algorithm
3. Use a different classifier
4. Play with regularization

Could help with over-fitting and under-fitting

Over-regularization can lead to underfitting
Under-regularization can lead to overfitting

First, diagnostics

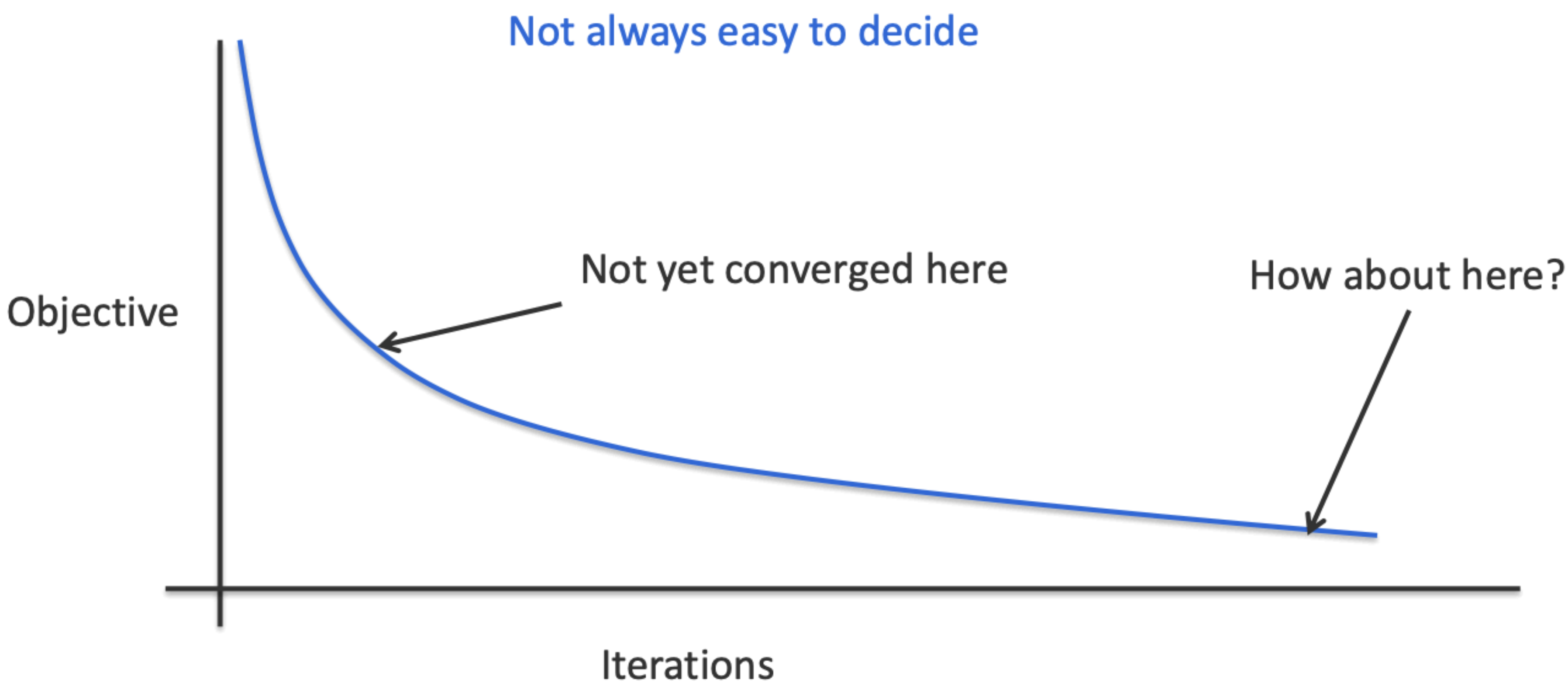
Easier to fix a problem if you know where it is

Some possible problems:

1. Over-fitting (high variance)
2. Under-fitting (high bias)
3. Your learning algorithm does not converge
4. Are you measuring the right thing?

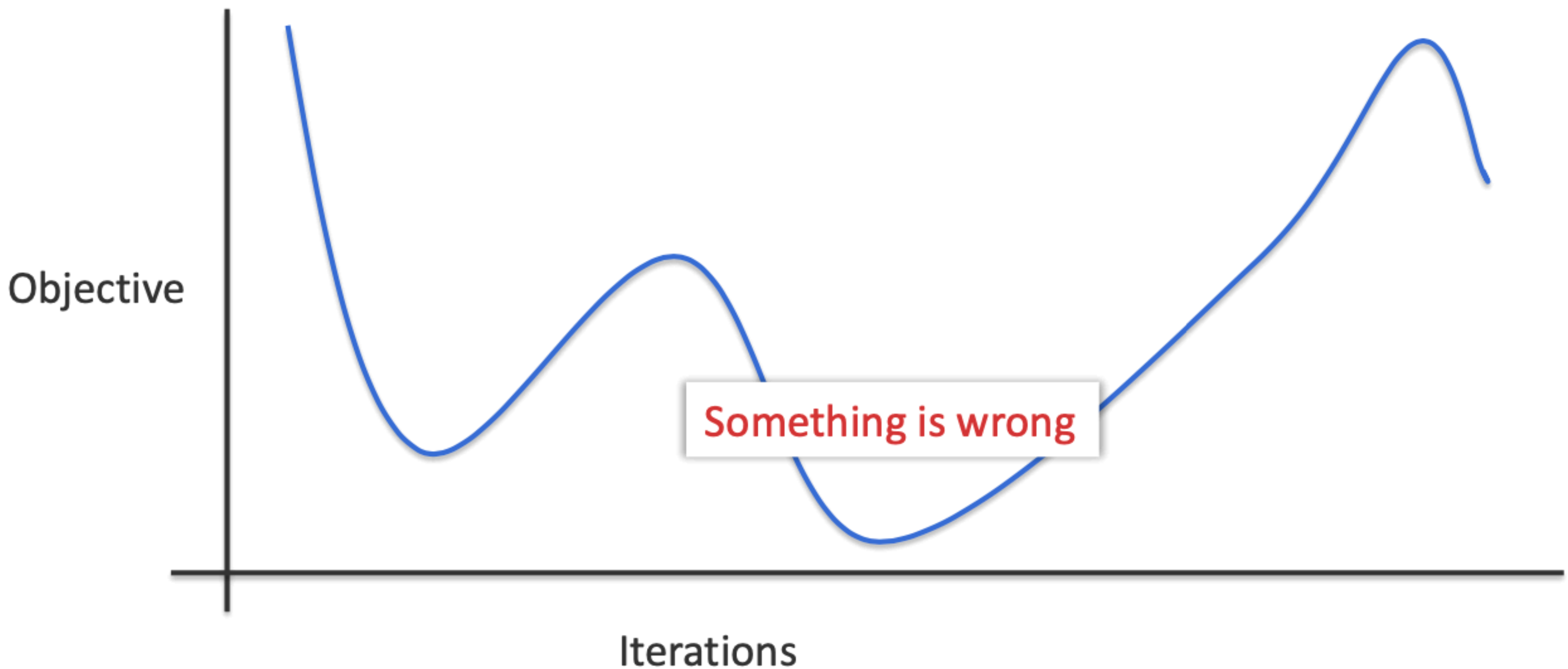
Does your learning algorithm converge?

Most of the algorithms we saw were framed as an optimization problem and minimize some loss function (i.e., objective): track the objective by plotting as a learning curve



Does your learning algorithm converge?

Most of the algorithms we saw were framed as an optimization problem and minimize some loss function (i.e., objective): track the objective by plotting as a learning curve



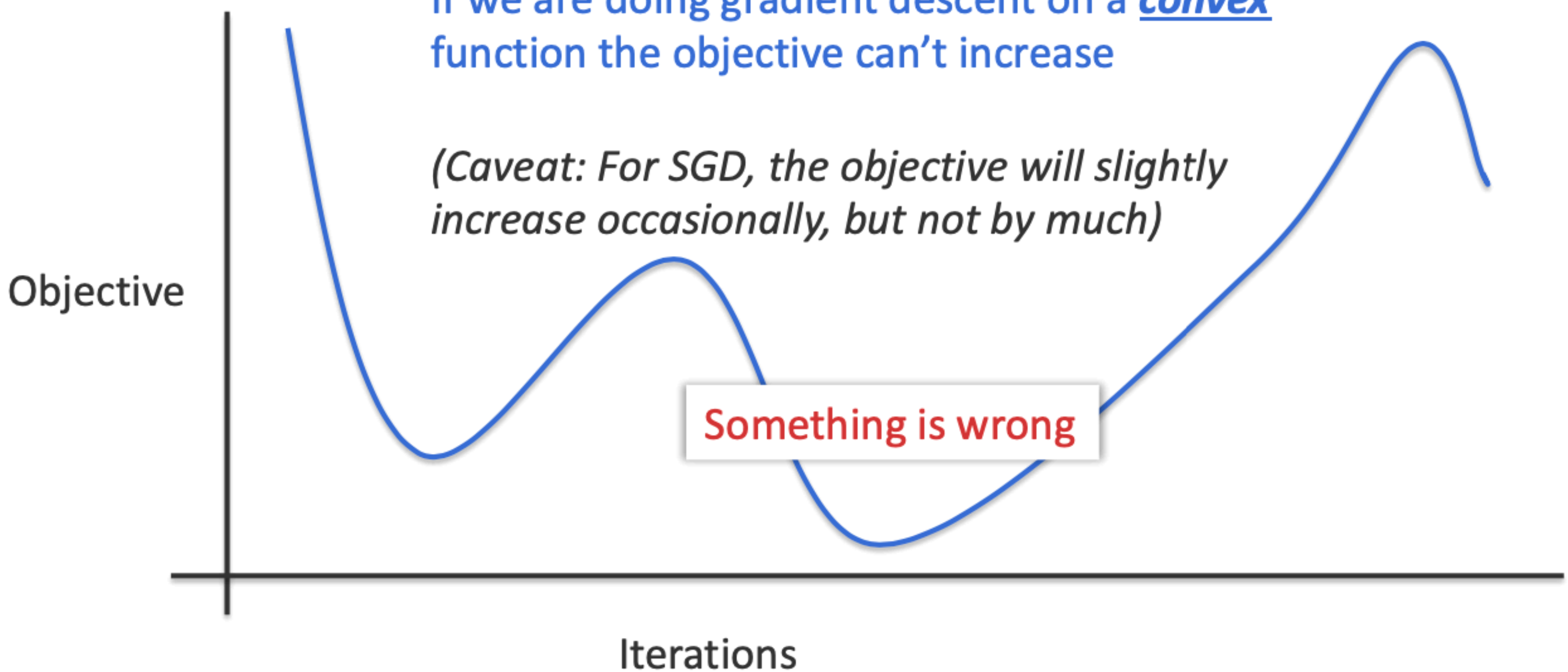
Does your learning algorithm converge?

Most of the algorithms we saw were framed as an optimization problem and minimize some loss function (i.e., objective): track the objective by plotting as a learning curve

Helps to debug

If we are doing gradient descent on a ***convex*** function the objective can't increase

(Caveat: For SGD, the objective will slightly increase occasionally, but not by much)



Different ways to improve your model

More training data Helps with over-fitting

Features

1. Use more (informative) features Helps with under-fitting
2. Use fewer features (reduces expressiveness of classifier) Helps with over-fitting
3. Use other features Could help with over-fitting and under-fitting

Better training

1. Run for more iterations
2. Use a different algorithm
3. Use a different classifier
4. Play with regularization

Track the objective for convergence: if not converging, gradient descent algorithm may not be good enough, maybe need to use Adam, etc.

Could help with over-fitting and under-fitting

Over-regularization can lead to underfitting
Under-regularization can lead to overfitting

First, diagnostics

Easier to fix a problem if you know where it is

Some possible problems:

1. Over-fitting (high variance)
2. Under-fitting (high bias)
3. Your learning algorithm does not converge
4. Are you measuring the right thing?

What to measure (for classification)

Accuracy of prediction is the most common measurement

But if your data set is unbalanced, accuracy may be misleading

- 1000 positive examples, 1 negative example
- A classifier that always predicts positive will get 99.9% accuracy. Has it really learned anything?

Unbalanced labels → measure label specific precision, recall and F- score

- Precision for a label: Among examples that are predicted with label, what fraction are correct
- Recall for a label: Among the examples with given ground truth label, what fraction are correct
- F-score: Harmonic mean of precision and recall

First, diagnostics

Easier to fix a problem if you know where it is

Some possible problems:

1. Over-fitting (high variance)
2. Under-fitting (high bias)
3. Your learning algorithm does not converge
4. Are you measuring the right thing?

Practical Advice

ADDING MACHINE LEARNING TO YOUR FAVORITE TASK

What to watch out for

Do you have the right evaluation metric?

- And does your loss function reflect it?

Beware of contamination: ensure that your training data is not contaminated with the test set

- Learning = generalization to new examples
- Do not look at your test set either. You may inadvertently contaminate the model
- Beware of contaminating your features with the label!
- (Be suspicious of perfect predictors)

What to watch out for

Be aware of bias vs. variance tradeoff

- Or over-fitting vs. under-fitting

Be aware that intuitions may not work in high dimensions

- No proof by picture
- Curse of dimensionality

A theoretical guarantee may only be theoretical

- May make invalid assumptions (e.g.,: the data is linearly separable)
- May only be legitimate with infinite data (e.g., estimating probabilities) or infinite time
- Experiments on real data are equally important

Big data is not enough

But more data is always better

- Cleaner data is even better

Remember that learning is impossible without some bias that simplifies the search

- Otherwise, no generalization

Learning requires knowledge to guide the learner

- *Machine learning is <u>not</u> a magic wand*

What knowledge?

Which model is the right one for this task?

- Linear models, decision trees, deep neural networks, etc

Which learning algorithm?

- Does the data violate any crucial assumptions that were used to define the learning algorithm or the model?
- Does that matter?

Feature engineering is crucial but intrinsic to application domain

Implicitly, these are all assumptions about the nature of the problem

Miscellaneous advice

Learn simpler models first

- If nothing, at least they form a baseline that you can improve upon

Ensembles seem to work better

Think about whether your problem is learnable at all

- Maybe there is no informative signal in your data, e.g., what if labels are randomly generated
- Learning = generalization

A retrospective look at the course

Learning = generalization

Tom Mitchell, Machine Learning book (1997)

"A computer program is said to learn from **experience E** with respect to some **class of tasks T** and **performance measure P**, if its performance at tasks in T, as measured by P, improves with experience E."

We saw different "models"

Or: what kind of a function should a learner learn

- Peceptron: linear classifiers
- Decision trees: non-linear classifiers/regressors
- Neural networks: non-linear classifiers/regressors

Different learning paradigms

Supervised learning: learn with a teacher

We did not see

- Unsupervised learning: learn without a teacher
- Semi-supervised learning: learn with and without a teacher
- Active learning: learner and teacher interact with each other
- Reinforcement learning: learn by interacting in environment

Learning algorithms

Online algorithms: learner can access only one labeled example a a time

- Perceptron

Batch algorithms: learner can access the entire dataset

- Decision trees
- Neural networks

Representing data

What is the best way to represent data for a particular task

- Features
- But what features are best?

Feature engineering is one of key things you will need to do

- Algorithms are already implemented …

Many things we did not see …

Focus of this course was on the underlying concepts and algorithmic ideas in the field of machine learning

1. A broad theoretical and practical understanding of machine learning paradigms and algorithms

2. Ability to implement learning algorithms

3. Identify where machine learning can be applied and make the most appropriate decisions

Thank you!

Have a wonderful summer :-)