PlanetS

UNIVERSITÄT BERN

Building Machine Learning Algorithms

A Practical Guide

Paolo Favaro

Workshop on Machine Learning - Observatoire de Geneve

Contents

- Practical rules with examples to build machine learning algorithms

Building a Machine Learning Algorithm

1. Build a dataset
2. Define a model
3. Define a cost function
4. Define an optimization procedure

Example

- **Dataset** of cars and non-cars

x



$y = 1$ (cars)

$y = 0$ (non-cars)

Example

- **Model**

$$p(y = 1|x; \theta) = \sigma(\theta^\top x) \quad \text{with} \quad \sigma(z) = \frac{1}{1 + e^{-z}}$$

logistic regression

Example

- **Cost function**

$$
\begin{aligned}
\sum_{i=1}^{m} \log p(y^i | x^i; \theta) &= \sum_{i=1}^{m} \log \sigma(\theta^\top x^i)^{y^i} \left(1 - \sigma(\theta^\top x^i)\right)^{1-y^i} \\
&= \sum_{i=1}^{m} y^i \log \sigma(\theta^\top x^i) + (1 - y^i) \log \left(1 - \sigma(\theta^\top x^i)\right) \\
&= \sum_{i=1}^{m} y^i \theta^\top x^i - \log \left[e^{+\theta^\top x^i} \left(1 + e^{-\theta^\top x^i}\right) \right]
\end{aligned}
$$

maximum likelihood

Example

- **Optimization procedure**

$$\theta_{t+1} = \theta_t + \alpha \sum_{i=1}^{m} \left(y^i - \sigma(\theta_t^\top x^i)\right) x^i$$

(batch) gradient ascent

Stochastic Gradient Descent

- A variation of (batch) gradient descent introduced to handle large training sets

- Applies to cost functions written as a sum of training samples (i.e., in the form of an expectation)

Stochastic Gradient Descent

- **Idea**: approximate gradient by using one (or a few) sample(s)

- Previous example: $\theta_{t+1} = \theta_t + \alpha \sum_{i \in Z_t \subset [1,m]} \left(y^i - \sigma(\theta_t^\top x^i)\right) x^i$

- **stochastic gradient descent** when Z_t singleton; **minibatch gradient descent** when Z_t larger; **incremental gradient method** (Bertsekas and Tsitsiklis 2000)

Stochastic Gradient Descent

- SGD will converge to a local minimum under some smoothness conditions on the cost

- Is the **fundamental optimization method** used in deep learning

Example (revisited)

- **Dataset** of cars and non-cars (>1M samples)

x

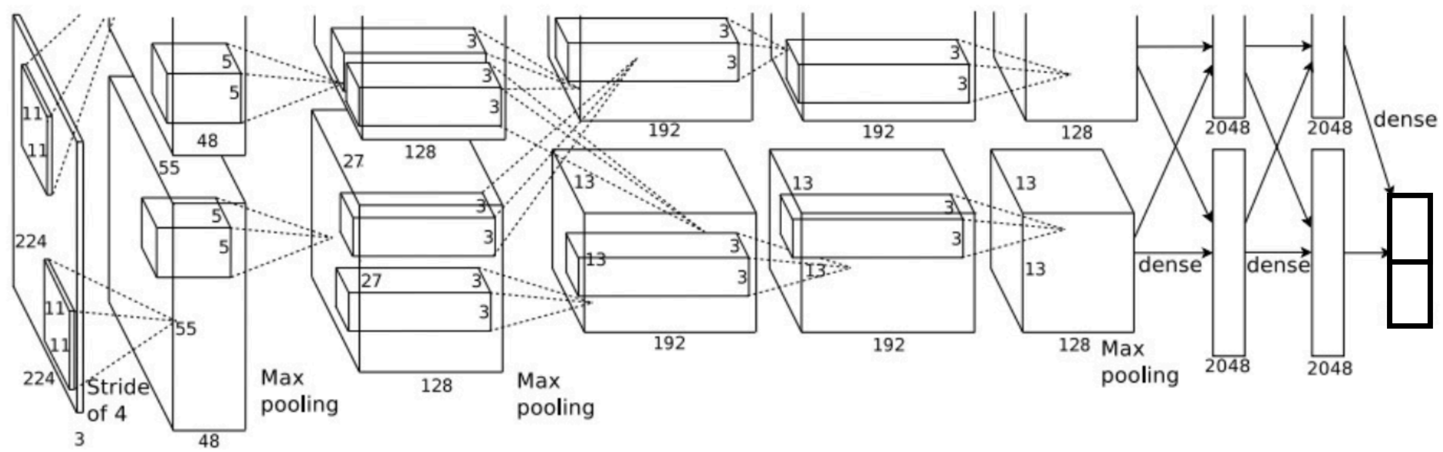
$y = 1$ (cars)

$y = 0$ (non-cars)

Example (revisited)

- **Model**

$$p(y|x;\theta) =$$



binary output

convolutional neural network

Example (revisited)

- **Cost function**

$$\sum_{i=1}^{m} \log p(y^i | x^i; \theta)$$

negative cross entropy (maximum likelihood)

Example (revisited)

- **Optimization procedure**

$$\theta_{t+1} = \theta_t + \alpha_t \frac{\nabla_\theta p(y^i | x^i; \theta_t)}{p(y^i | x^i; \theta_t)} \qquad i \sim U[1, m]$$

stochastic gradient ascent

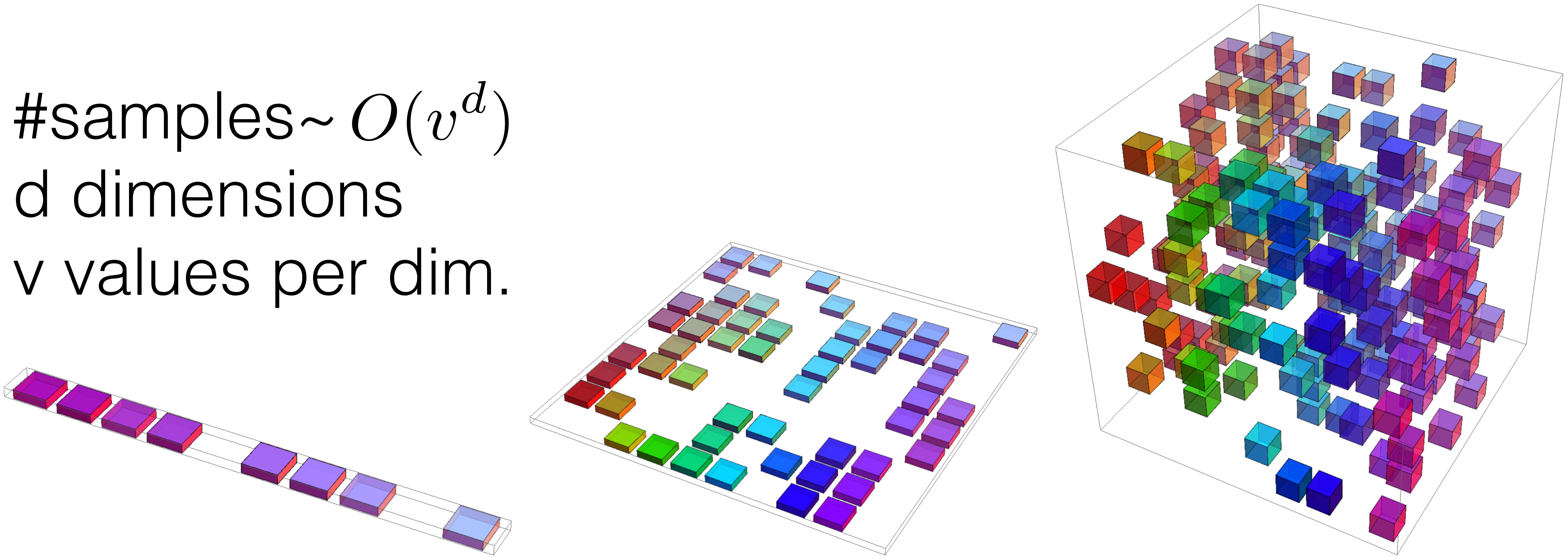
Challenges in Machine Learning

- Prior to deep learning, methods in ML could solve AI problems (e.g., object recognition) with limited success
- Difficulties
  - High-dimensional data
  - Generalization with high-capacity functions
  - High computational costs

Curse of Dimensionality

- Number of possible configurations grows exponentially with number of (data) dimensions
- We need an exponentially growing number of data samples to cover the configurations of interest

#samples~ $O(v^d)$
d dimensions
v values per dim.



Locality and Smoothness

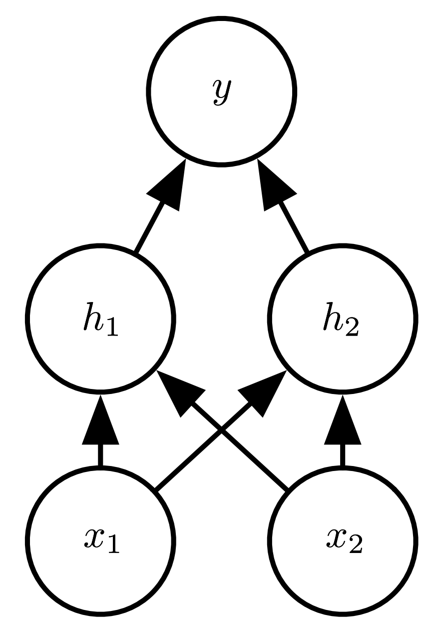
- Most ML algorithms make use of local constancy or smoothness
- This allows limited generalization
- For example, in the case of nearest neighbor, one needs at least one example per region of interest (the prediction function is as complex as the data)

Locality and Smoothness

- We need to generalize well complex functions with few examples

- Deep learning does so by introducing assumptions on the data generating distribution

- The key assumption is that data is generated by a **composition of factors** (typically, in a hierarchical structure)



Practical Methodology

- We have seen a recipe for building a ML algorithm
- In practice, is the implementation so straightforward?
- What can we do when we have a limited time and resource budget?

credits to Andrew Ng - University of Stanford

Strategy

1. **Look at/Plot the data**

2. Start with the simplest (but non trivial) implementation

3. Test it on validation data

   (i) **Diagnostics**: Plot learning curves to decide changes (e.g., more training data, better features etc)

   (ii) **Error/Ablative analysis**: Manually examine samples in the validation data where your algorithm makes mistakes; see if there are any patterns/trends; determine what works and what does not.

Debugging

- Two separate tasks
  - Debugging code: Code does not implement correctly the algorithm
  - Debugging algorithms: The algorithm does not perform as desired

Debugging

- Two separate tasks
  - Debugging code: Code does not implement correctly the algorithm
  - Debugging algorithms: The algorithm does not perform as desired

Diagnostics

- **Machine Learning Diagnostics**
  A test that you can run to gain insight on what is and isn’t working with a learning algorithm and to gain insight on how to best improve its performance

Debugging Learning Algorithms

- Example: Spam/Non-spam classification

- You carefully chose features (100 words out of 50K in English)

- Bayesian logistic regression implemented with gradient descent gets 20% error, which is too high

$$\max_{\theta} \sum_{i=1}^{m} \log p(y^i | x^i; \theta) - \lambda |\theta|^2$$

- What to do next?

Debugging Learning Algorithms

- Common attempts (trial and error)
  (a) Try getting more training examples
  (b) Try a smaller set of features
  (c) Try a larger set of features
  (d) Try changing the features: Email header vs. email body features
  (e) Run gradient descent for more iterations
  (f) Try Newton's method
  (g) Use a different regularization parameter
  (h) Try using an SVM
- This approach might work, but it's **very time-consuming**, and largely a matter of luck whether you end up fixing what the problem really is

Debugging Learning Algorithms

- Better approach

  1. Run **diagnostics** to find the problem

  2. Fix whatever the problem is

Debugging Learning Algorithms

1. Start with stating the problem: Bayesian logistic regression's test error is 20% (unacceptably high)

2. Make hypotheses for what the problem could be. For example,

   - Overfitting (high variance)

   - Too few features to classify spam (high bias)

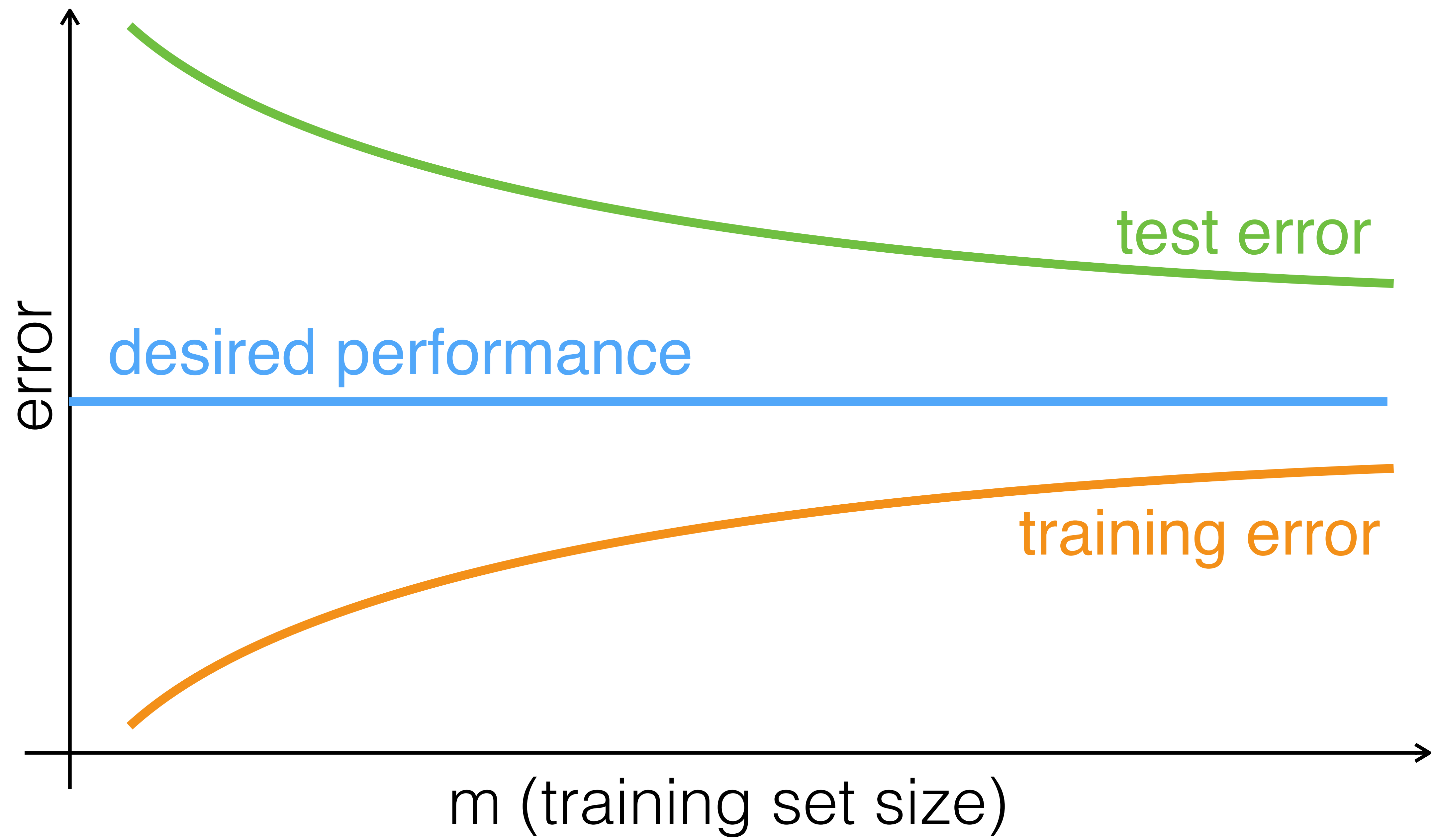
3. Run diagnostics. For example,

   - Variance: Training error will be much lower than test error

   - Bias: Training error will also be high

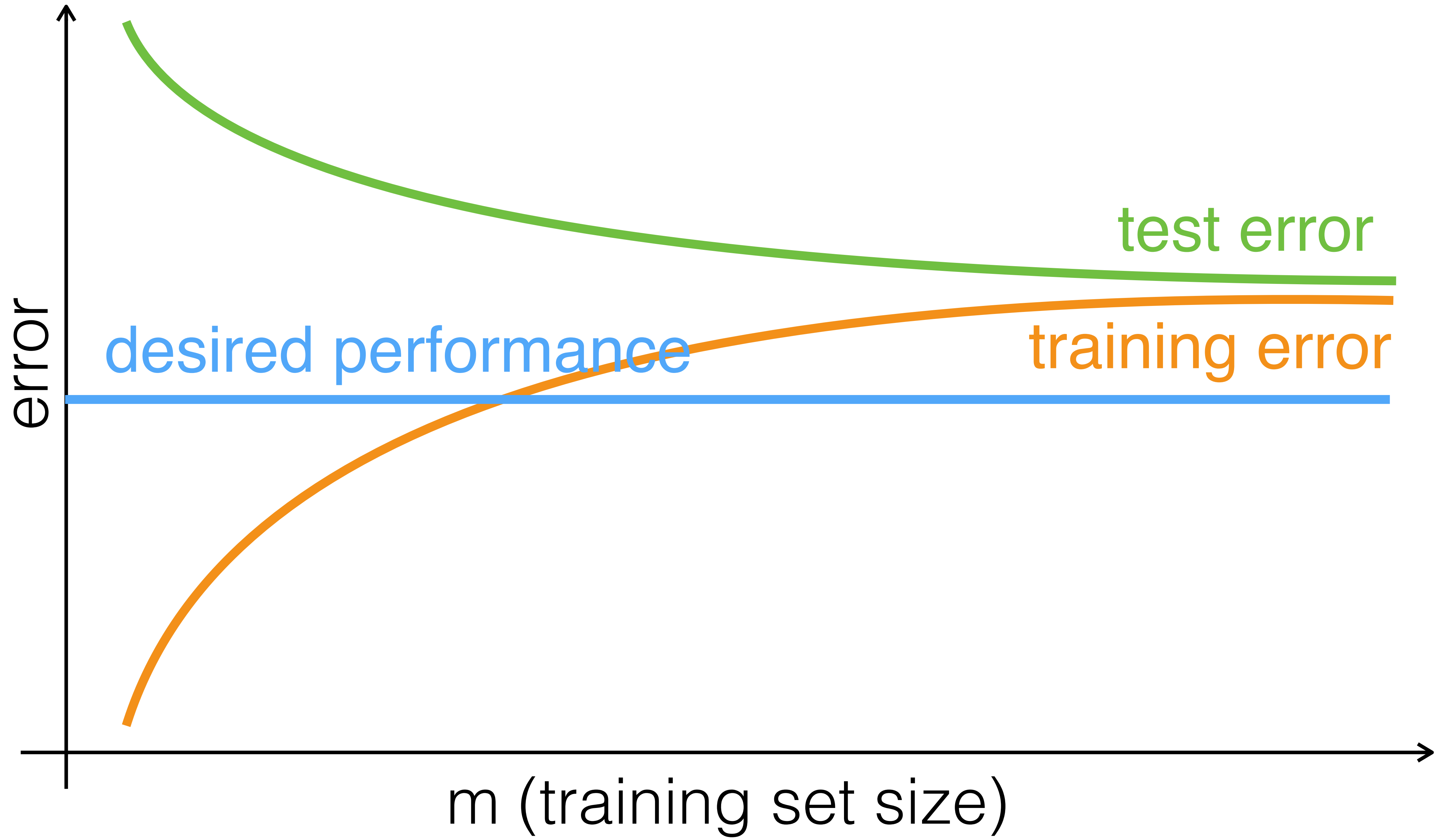
Bias vs Variance

typical learning curve for high variance



Bias vs Variance

typical learning curve for high bias



Fix Based on Diagnostics

- Fixes (and diagnostics)

  (a) Try getting more training examples → fixes high variance

  (b) Try a smaller set of features → fixes high variance

  (c) Try a larger set of features → fixes high bias

  (d) Try changing the features → fixes high bias

  (e) Run gradient descent for more iterations

  (f) Try Newton's method

  (g) Reg. parameter (lower -> fixes high bias, higher -> fixes high var.)

  (h) Try using an SVM

Other Diagnostics

- Bias vs variance is one common diagnostic
- In general, you need to construct your own diagnostics based on the specific problem at hand
- Other examples
  - Is the algorithm converging?
  - Are you optimizing the right function?
  - Is the model correct?
  - What is the best regularization value?

Optimization Diagnostics

- Example
  - Bayesian logistic regression gets 2% error on spam and 2% error on non-spam (too high)
  - SVM with a linear kernel gets 10% error on spam and 0.01% error on non-spam (acceptable)
  - You want to use logistic regression because of efficiency
- What to do next?

Optimization Diagnostics

- SVM outperforms Bayesian logistic regression (BLR) but you want to deploy BLR

- You care about the weighted accuracy

$$a(\theta) = \max_{\theta} \sum_i w^i \delta[h_\theta(x^i) = y^i]$$

and have $a(\theta_{SVM}) > a(\theta_{BLR})$

- BLR maximizes $J(\theta) = \sum_i \log p(y^i | x^i; \theta) - \lambda |\theta|^2$

- Diagnostic $J(\theta_{SVM}) > J(\theta_{BLR})$?

Optimization Diagnostics

- Case 1: $J(\theta_{SVM}) > J(\theta_{BLR})$

  BLR fails to maximize J (problem is with convergence/optimization algorithm)

- Case 2: $J(\theta_{SVM}) < J(\theta_{BLR})$

  BLR succeeds at maximizing J. Then, J is the wrong objective if we care about a

Fix Based on Diagnostics

- Fixes (and diagnostics)
  - (a) Try getting more training examples → fixes high variance
  - (b) Try a smaller set of features → fixes high variance
  - (c) Try a larger set of features → fixes high bias
  - (d) Try changing the features → fixes high bias
  - (e) Run gradient descent for more iterations → fixes opt. algorithm
  - (f) Try Newton's method → fixes opt. algorithm
  - (g) Reg. parameter → fixes opt. objective
  - (h) Try using an SVM → fixes opt. objective

Error analysis

- Explain the difference between current performance and perfect performance
- Understand what the sources of error are
- How much error is attributable to each component in the algorithm?
- Plug in ground truth in each component (if possible) and see how accuracy changes

Ablation Analysis

- Explain the difference between current performance and some baseline performance (much poorer)

- Which component of the algorithm really helps?

- Remove components one at a time and see how the algorithm breaks

More Tools

- Consider
  - Toy examples
  - Simple data first
  - Conditions where the result is known
  - Cases where analytic solutions are available
  - One source (input or intermediate) at a time

More Tools

- Consider
  - Set most items to known working settings
  - Simulating mentally the behavior of the algorithm
  - Worst-case scenario and conditions leading to it
  - Input data and settings that break the algorithm
  - Extreme conditions (meaningful behavior?)