Lecture 1: Machine Learning Basics

Ali Harakeh

University of Waterloo

WAVE Lab
ali.harakeh@uwaterloo.ca

May 1, 2017

Overview

Section 1

Learning Algorithms

- A machine learning algorithm is an algorithm that is able to **learn** from data.
- A machine is said to have **learned** from **Experience** E with respect to some **Task** T, as measured by a **Performance Measure** P, if its performance on T as measured by P, improves with E.

The Task T

- Example T: Vehicle Detection In Lidar Data.
- Approach 1: Hard code what a vehicle is in Lidar data based on Human experience.
- Approach 2: Learn what a vehicle is in Lidar data.
- Machine learning allows us to tackle tasks that are too difficult to be hard coded by humans.

The Task T

- Machine learning algorithms are usually described in terms of how the algorithm should process an **example** $\mathbf{x} \in \mathbb{R}^n$.
- Each entry x_j of $\mathbf{x}$ is called a **feature**.
- Example : Features in an image can be its pixel values.

Common Machine Learning Tasks

- **Classification**: Find $f(\mathbf{x}) : R^n \Rightarrow \{1, ..., k\}$ that maps examples $\mathbf{x}$ to one of k classes.
- **Regression**: Find $f(\mathbf{x}) : R^n \Rightarrow R$ that maps examples to the real line.

The Performance Measure P

- A quantitative measure of performance is required in order to evaluate a machine's ability to learn.
- P depends on task T.
- **Classification**: P is usually the **accuracy** of the model . Another equivalent measure is the **error rate** (also called the expected 0-1 loss).

The Experience E

- Machine learning algorithms can be classified into two classes: **supervised** and **unsupervised** based on what kind of experience they are allowed to have during the learning process.
- Machine learning algorithms are usually allowed to experience an entire **dataset**.

Categorizing Algorithms Based On E

- **Unsupervised learning algorithms** experience a dataset containing many features, then learn useful properties of the structure of this dataset.
- **Supervised learning algorithms** experience a dataset containing features, but each example is also associated with a **label** or **target**.

Dataset Splits

- We usually split our dataset to three subsets: **train, val, test**.
- E is usually experiencing **train** and **val** sets.
- P is usually evaluated on **test** set.

Section 2

Capacity, Overfitting, and Underfitting

- The main challenge in machine learning is that the algorithm must perform well on **new, unseen input data**.
- This ability is called **generalization**.
- We usually have access to the training set, and we try to minimize some error measure called the **training error**. This is standard optimization.
- What differentiates machine learning from standard optimization is that we care to minimize the **generalization error**, the error evaluated on the test set.

The Data Generating Distribution p_{data}

- Is minimizing over training set error guaranteed to provide parameters that minimize the test set error ?
- Under the **i.i.d** assumption on train and test examples, the answer is "Yes".

The factors that determine how well a machine learning algorithm performs is its ability to:

- Make the training error small.
- Make the gap between training and test error small.

Overfitting, Underfitting, and Capacity

- **Underfitting** occurs when the model is not able to obtain a sufficiently low error value on the training set.
- **Overfitting** occurs when the gap between the training error and test error is too large.
- **Capacity** is a model's ability to fit a wide variety of functions.

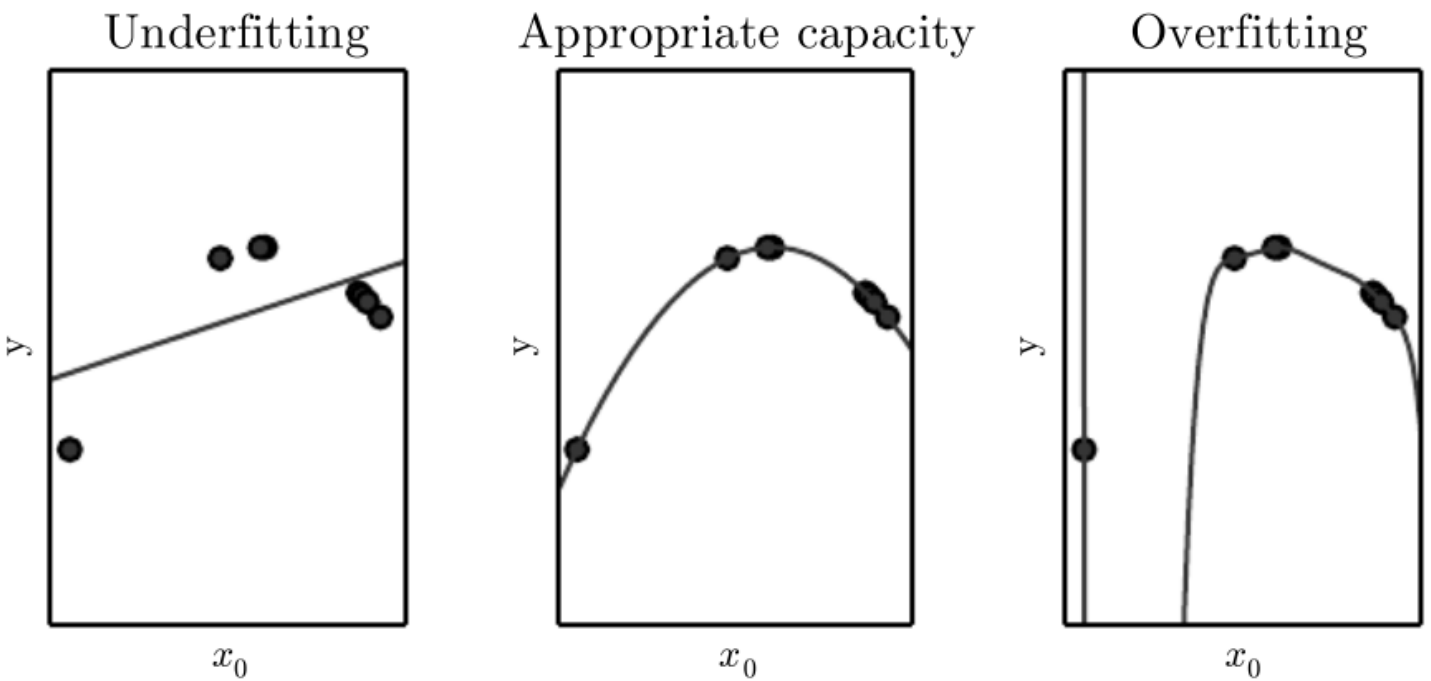
Overfitting, Underfitting, and Capacity

- There is a direct relation between the model's capacity and whether it will overfit or underfit.
- Models with low capacity may struggle to fit the training set.
- Models with high capacity can overfit by memorizing properties of the training set that do not serve them well on the test set.

Controlling Capacity: The **Hypothesis Space**

- Hypothesis Space : the set of functions that the learning algorithm is allowed to select as being the solution.
- Increase the model's capacity by expanding the hypothesis space.

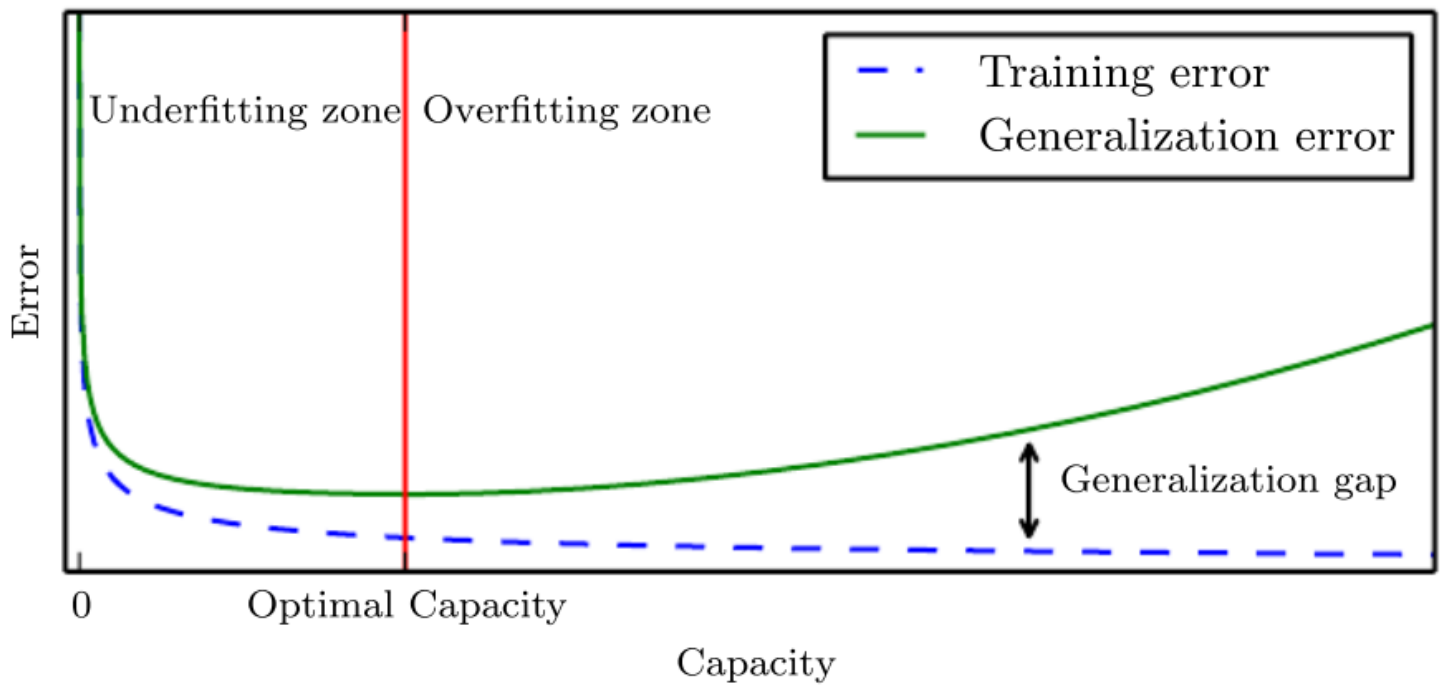
Controlling Capacity: The **Hypothesis Space**



Controlling Capacity: The **Hypothesis Space**

- From statistical learning theory: The discrepancy between training error and generalization error is bounded from above by a quantity that grows as the model capacity grows but shrinks as the number of training examples increases (Vapnik and Chervonenkis, 1971).
- Intellectual justification that machine learning algorithms **can work**!
- Note: We must remember that while simpler functions are more likely to generalize (to have a small gap between training and test error) we must still choose a sufficiently complex hypothesis to achieve low training error.

Controlling Capacity: The **Hypothesis Space**



Bayes Error

- The **ideal model** is an oracle that simply knows the true probability distribution that generates the data.
- The error incurred by an oracle making predictions from the true distribution $p(\mathbf{x}, y)$ is called the **Bayes error**.
- Example: In the case of supervised learning, the mapping from $\mathbf{x}$ to y may be inherently stochastic, or y may be a deterministic function that involves other variables besides those included in $\mathbf{x}$.

The No Free Lunch Theorem

- Averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.
- What are the consequences of this theorem?

Controlling The Capacity: **Regularization**

- The behavior of our algorithm is strongly affected not just by how large we make the set of functions allowed in its hypothesis space, but by the specific identity of those functions.
- Regularization can be used as a way to give preference to one solution in our hypothesis space (more general than restricting the space itself).
- Weight Decay: $\lambda w^T w$

Controlling The Capacity: **Regularization**

- More formally, **Regularization** is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.

Section 3

Hyperparameters and Validation Sets

Hyperparameters

- **Hyperparameters** are any variables that affect the behavior of the learning algorithm, but are not adapted by the algorithm itself.

Importance of the Validation Set

- In a **test-train-val** split, learning is performed on the **train** set. The choice of hyperparameters is done by evaluation on the **val** set.
- Construction of a **train-val-test** split: Split the **data set** to **train-test** at a $1:1$ ratio. Then, split the **train** set to **train-val** at a $4:1$ ratio.

- What happens when the **same test set** has been used repeatedly to evaluate performance of different algorithms over many years?

Section 4

Estimators, Bias and Variance

Point Estimation

- **Point estimation** is an attempt to provide the single "best" prediction $\hat{\theta}$ of some quantity of interest θ. This quantity might be a scalar, vector, matrix, or even a function.
- Usually, point estimation is done using a set of data points:

$$\hat{\theta} = g(x^{(1)}, ..., x^{(m)})$$

- Note that g does not need to return a value close to θ, it even might not have the same set of allowable values.

Bias

- The bias of an estimator is:

$$bias(\hat{\theta}) = \mathbb{E}(\hat{\theta}) - \theta$$

- Bias measures the expected deviation of the estimate from the true value of the function or parameter.
- We say an estimator is **unbiased** if its bias is 0.
- We say an estimator is **asymptotically unbiased** if $\lim_{m \to \infty} bias(\hat{\theta}) = 0$.

Variance

- The variance $Var(\hat{\theta})$ of an estimator provides a measure of how we would expect the estimate we compute from data to vary as we independently resample the dataset from the underlying data generating process.

The Bias-Variance Trade Off

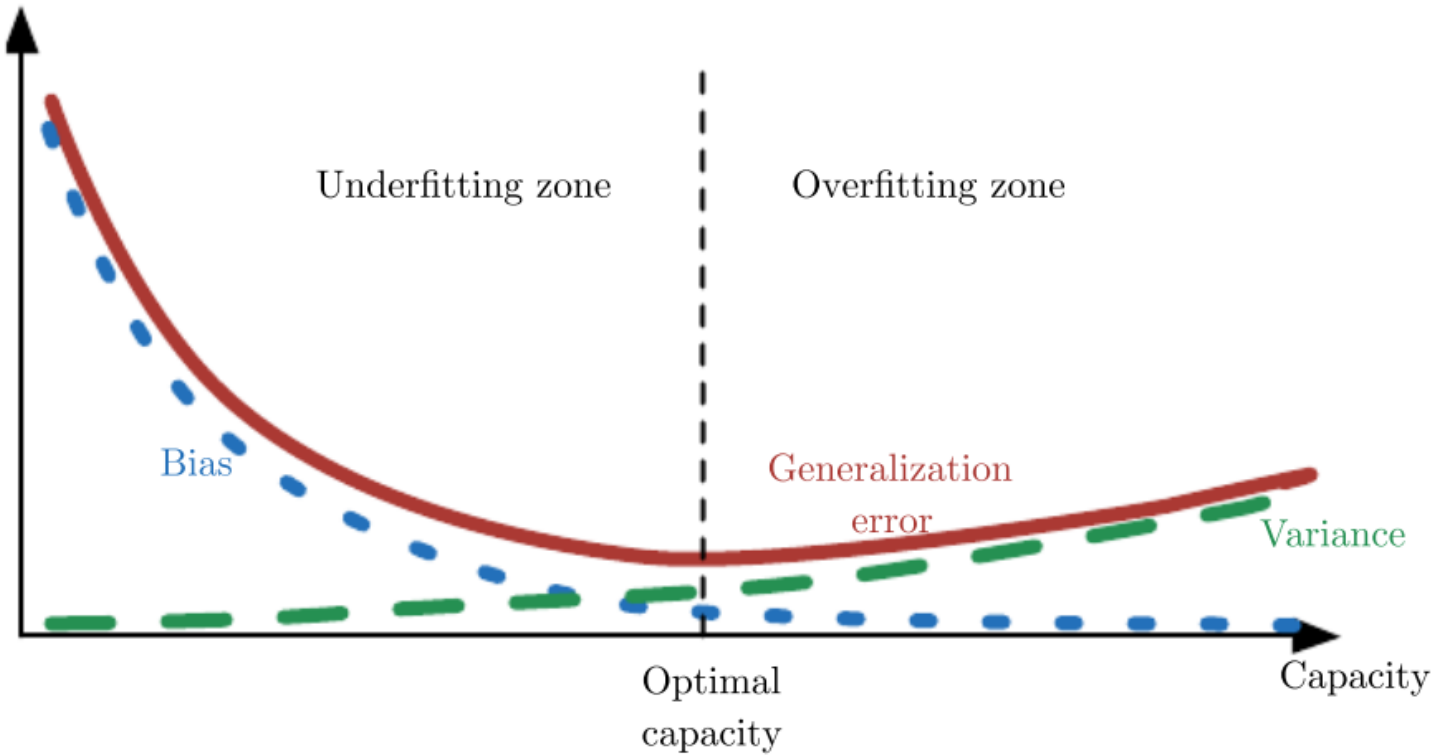
- How to choose between two estimators, one with large bias and the other with large variance ?
- Mean-Square Error of the estimates:

$$\begin{aligned} MSE &= \mathbb{E}[(\hat{\theta} - \theta)^2] \\ &= Bias(\hat{\theta})^2 + Var(\hat{\theta}) \end{aligned}$$

- MSE incorporates both bias and variance components.

Relation To Machine Learning

- The relationship between bias and variance is tightly linked to the machine learning concepts of capacity, underfitting and overfitting.
- **How ?**



Consistency

- **Consistency** is a desirable property of estimators. It insures that as the number of data points in our data set increase, our point estimate converges to the true value of θ.
- More formally, consistency states that:

$$\lim_{m \to \infty} \hat{\theta} \xrightarrow{p} \theta$$

- The convergence here is in probability.
- Consistency of an estimator ensures that the bias will diminish as our training data set grows.
- It is better to choose consistent estimators with large bias over estimators with small bias and large variance. Why ?

Section 5

ML and MAP Estimators

Maximum Likelihood Estimation

- **Maximum likelihood** (ML) is a principle used to derive estimators.
- Given m examples $\mathbb{X} = x^{(1)}, ..., x^{(m)}$ drawn **independently** form data generating distribution p_{data}:

$$\theta_{ML} = \underset{\theta}{\text{argmax}}\, p_{model}(\mathbb{X}; \theta)$$

- $p_{model}(\mathbf{x}; \theta)$ maps any configuration $\mathbf{x}$ to a real number, hence tries to estimate the true data distribution p_{data} .

Maximum Likelihood Estimation

- After some mathematical manipulation:

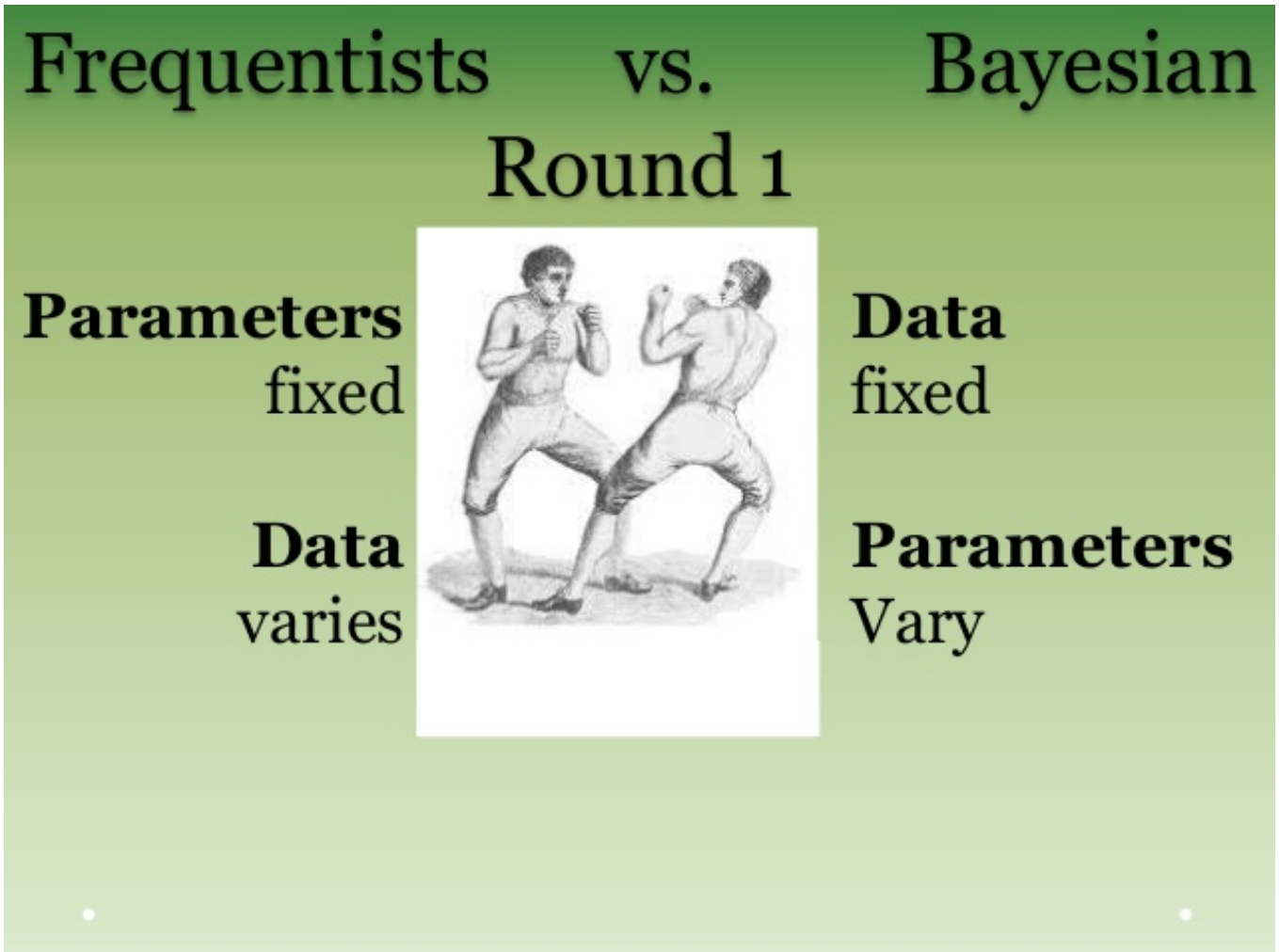
$$\theta_{ML} = \underset{\theta}{\text{argmax}}\, \mathbb{E}_{\mathbf{x}\sim\hat{p}_{data}} log\ p_{model}(\mathbf{x}, \theta)$$

- Ideally, we would like to have this expectation over p_{data}. Unfortunately, we only have access to the empirical distribution $\hat{p}_{data}$ from training data.
- Maximum likelihood can be viewed as a minimization of the dissimilarity between $\hat{p}_{data}$ and p_{model}. How ?

Maximum Likelihood Estimation

- Maximum likelihood can be shown to be the best estimator, asymptotically in terms of its rate of convergence as $m \to \infty$.
- The estimator derived by ML is **consistent**. However, certain conditions are required for consistency to hold:
  - The true distribution p_{data} must lie within the model family $p_{model}(.;\theta)$. Otherwise, no estimator can recover p_{data} even with infinite training examples.
  - There needs to exist a unique θ. Otherwise, ML will recover p_{data} but will not be able to determine the true value of θ used in the data generation process.
- Under these conditions, you are guaranteed to improve the performance of your estimator with more training data.

Maximum A Posteriori Estimation



Maximum A Posteriori Estimation

- **Bayesian Statistics**: The dataset is directly observed and so is not random. On the other hand, the true parameter θ is unknown or uncertain and thus is represented as a random variable.
- Before observing data, we represent our knowledge of θ using the prior probability distribution $p(\theta)$. After observing data, we use bayes rule to compute the posterior distribution $p(\theta|x^{(1)}...x^{(m)})$.

Maximum A Posteriori Estimation

- Usually, priors are chosen to be high entropy distributions such as **uniform** or **Gaussian** distributions. These distributions are described as **broad**.
- From Bayes rule we have:

$$p(\theta|x^{(1)}...x^{(m)}) = \frac{p(x^{(1)}...x^{(m)}|\theta)p(\theta)}{p(x^{(1)}...x^{(m)})}$$

Maximum A Posteriori Estimation

- To predict the distribution over new input data, marginalize over θ:

$$p(x_{new}|x^{(1)}...x^{(m)}) = \int p(x_{new}|\theta)p(\theta|x^{(1)}...x^{(m)})d\theta$$

- Example: Bayesian Linear Regression.

Maximum A Posteriori Estimation

- **Maximum a posteriori estimation** (MAP) tries to overcome the intractability of the full Bayesian treatment, by providing point estimates using the posterior probability:

$$\theta_{MAP} = \underset{\theta}{\text{argmax}}\, p(\theta|\mathbf{x}) \quad = \underset{\theta}{\text{argmax}}\, log\ p(\mathbf{x}|\theta) + log\ p(\theta)$$

- MAP Bayesian inference has the advantage of leveraging information that is brought by the prior and cannot be found in the training data.

Section 6

Gradient Based Optimization

Optimization

- **Optimization** refers to the task of either minimizing or maximizing some function $f(\mathbf{x})$ by altering the value of $\mathbf{x}$.
- $f(\mathbf{x})$ is called an **objective function**. In context of machine learning, it is also called the **loss**, **cost**, or **error** function.
- Notation: $\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} f(\mathbf{x})$ is the value of $\mathbf{x}$ that minimizes $f(\mathbf{x})$.

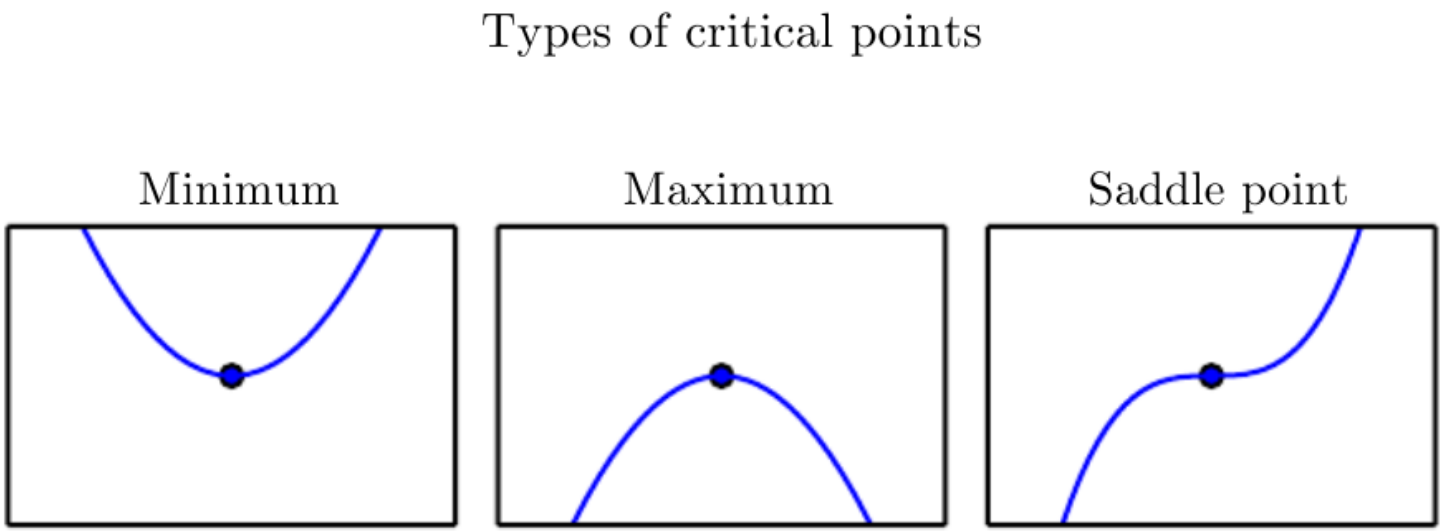
Using The Derivative For Optimization

- The derivative of a function specifies how to scale a small change in input in order to obtain the corresponding change in output.

$$f(\mathbf{x} + \epsilon) \approx f(\mathbf{x}) + \epsilon \nabla_{\mathbf{x}} f(\mathbf{x})$$

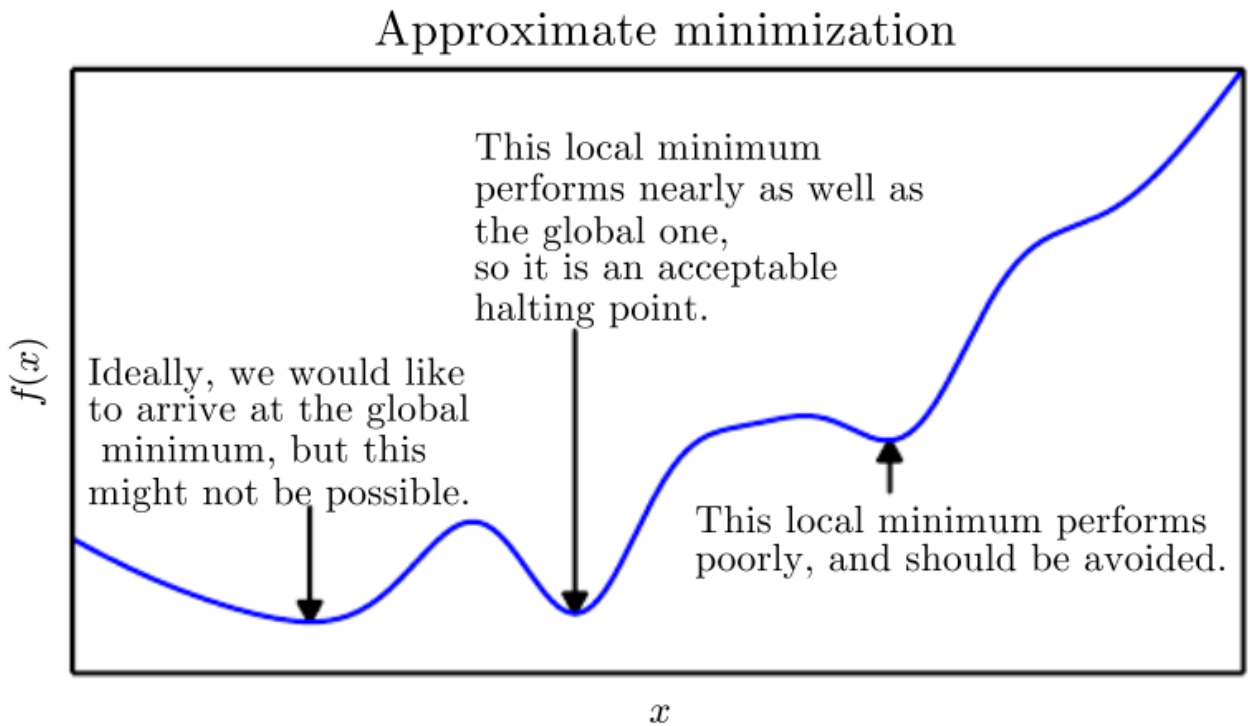
- The derivative is useful for optimization because it allows knowledge of how to change $\mathbf{x}$ to improve $f(\mathbf{x})$.
- Example: $f(\mathbf{x} - \epsilon \, sign(\nabla_{\mathbf{x}} f(\mathbf{x}))) \leq f(\mathbf{x})$ for small enough ϵ.

Critical Points



- A **critical** point or **stationary** point is a point $\mathbf{x}$ with $\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$.

Global vs Local Optimal Points



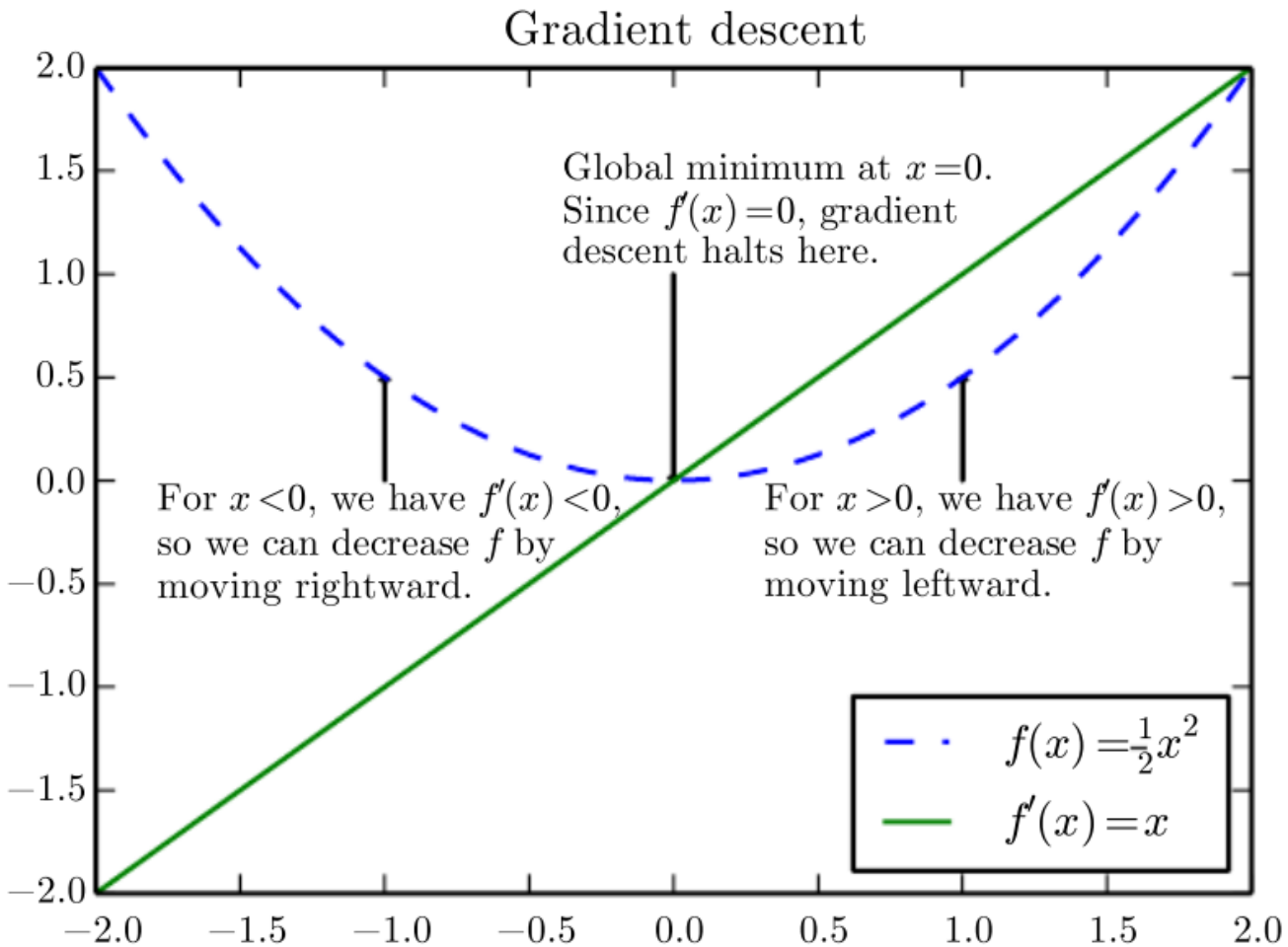
Gradient Descent

- Gradient descent proposes to update the parameter according to:

$$\mathbf{x} \leftarrow \mathbf{x} - \epsilon \nabla_{\mathbf{x}} f(\mathbf{x})$$

- ϵ is referred to as the learning rate.
- Gradient descent converges when all the elements in the gradient are almost equal to zero.

Gradient Descent



Stochastic Gradient Descent

- Nearly all of deep learning is powered by one optimization algorithm: SGD.
- Motivation behind SGD: The cost function used by a machine learning algorithm often decomposes as a sum over training examples of some per-example loss function:

$$J(\theta) = \mathbb{E}_{\mathbf{x},y\sim\hat{p}_{data}} L(\mathbf{x}, y, \theta)$$
$$= \frac{1}{m}\sum_{i=1}^{m} L(\mathbf{x}^{(i)}, y^{(i)}, \theta)$$

Stochastic Gradient Descent

- To minimize the loss over θ, the gradient needs to be computed.

$$\nabla_\theta J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta L(\mathbf{x}^{(i)}, y^{(i)}, \theta)$$

- What is the computational cost for computing the gradient above ?

Stochastic Gradient Descent

- SGD relies on the fact that the gradient is an **expectation**, hence can be approximated with a small set of samples.
- let m' be a **minibatch** uniformly drawn from our training data.

$$\nabla_\theta J(\theta) = \frac{1}{m'} \sum_{i=1}^{m'} \nabla_\theta L(\mathbf{x}^{(i)}, y^{(i)}, \theta)$$

- The SGD update rule becomes :

$$\theta \leftarrow \theta + \epsilon \nabla_\theta J(\theta)$$

Section 7

Challenges That Motivate Deep Learning

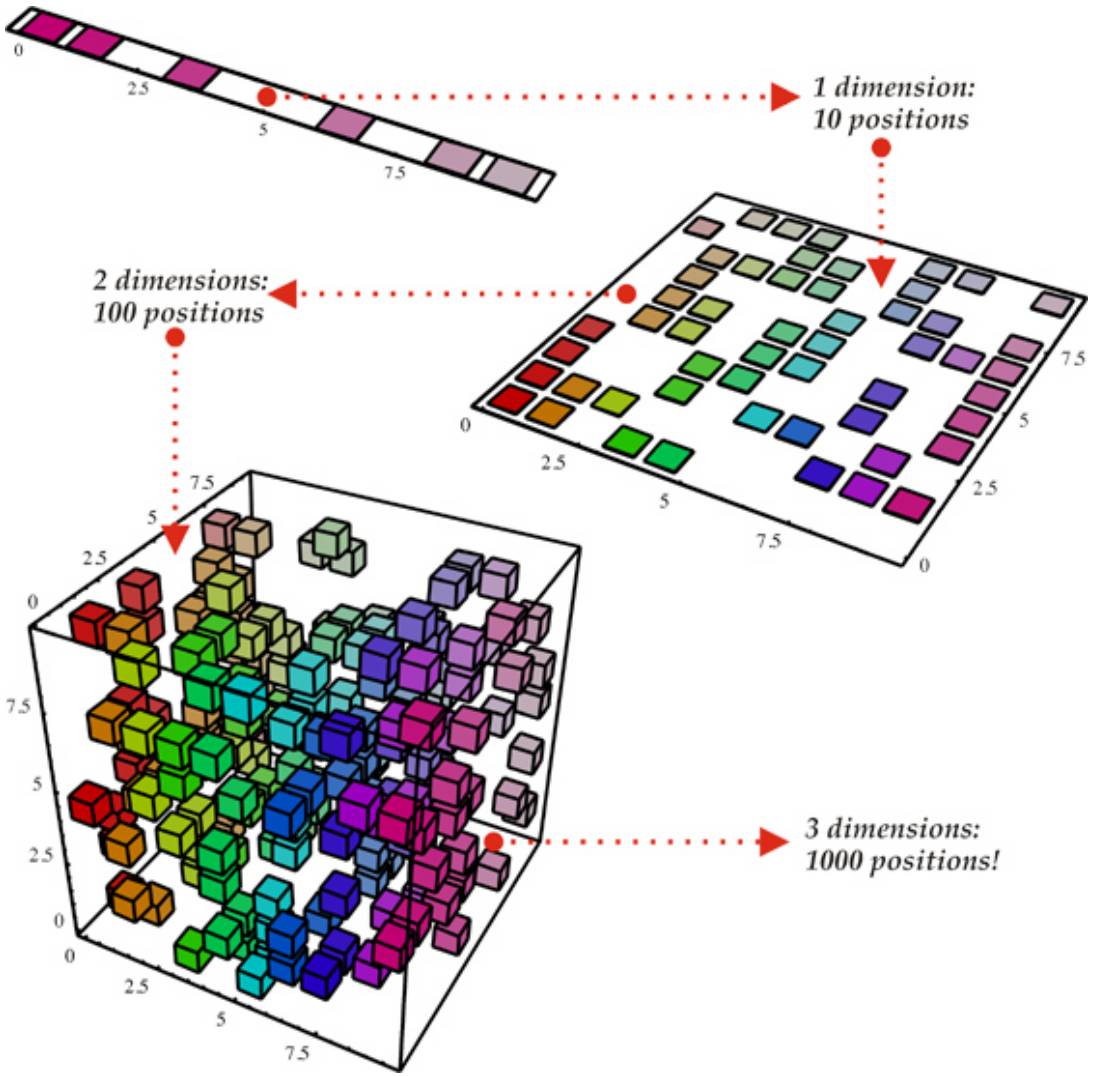
Major Obstacles For Traditional Machine Learning

- The development of deep learning was motivated by the failure of traditional ML algorithms when applied to central problems in AI due to:
  - The mechanisms used to achieve generalization in traditional machine learning are insufficient to learn complicated functions in high-dimensional spaces.
  - The challenge of generalizing to new examples becomes exponentially more difficult when working with high-dimensional data.

The Curse Of Dimensionality

- Many machine learning problems become exceedingly difficult when the number of dimensions in the data is high.
- This is because the number of distinct configurations of a set of variables increase exponentially as the number of variables increase.
- How does that affect ML algorithms ?

The Curse Of Dimensionality



Local Constancy And Smoothness Regularization

- In order to generalize well, machine learning algorithms need to be guided by prior beliefs about what kind of function they should learn.
- Among the most widely used priors is the **smoothness** or **local constancy** prior.
- A function is said to have local constancy if it does not change much within a small region of space.
- As the machine learning algorithm becomes simpler, it tends to rely extensively on this prior.
- Example: K nearest neighbors.

Local Constancy And Smoothness Regularization

- In general, traditional learning algorithms require $O(k)$ examples to distinguish $O(k)$ regions in space.
- Is there a way to represent a complex function that has many more regions to be distinguished than the number of training examples ?

Local Constancy And Smoothness Regularization

- **Key insight**: Even though the number of regions of a function can be very large, say $O(2^k)$, the function can be defined with $O(k)$ examples as long as we introduce additional dependencies between regions via generic assumptions.
- **Result**: Non local generalization is actually possible.

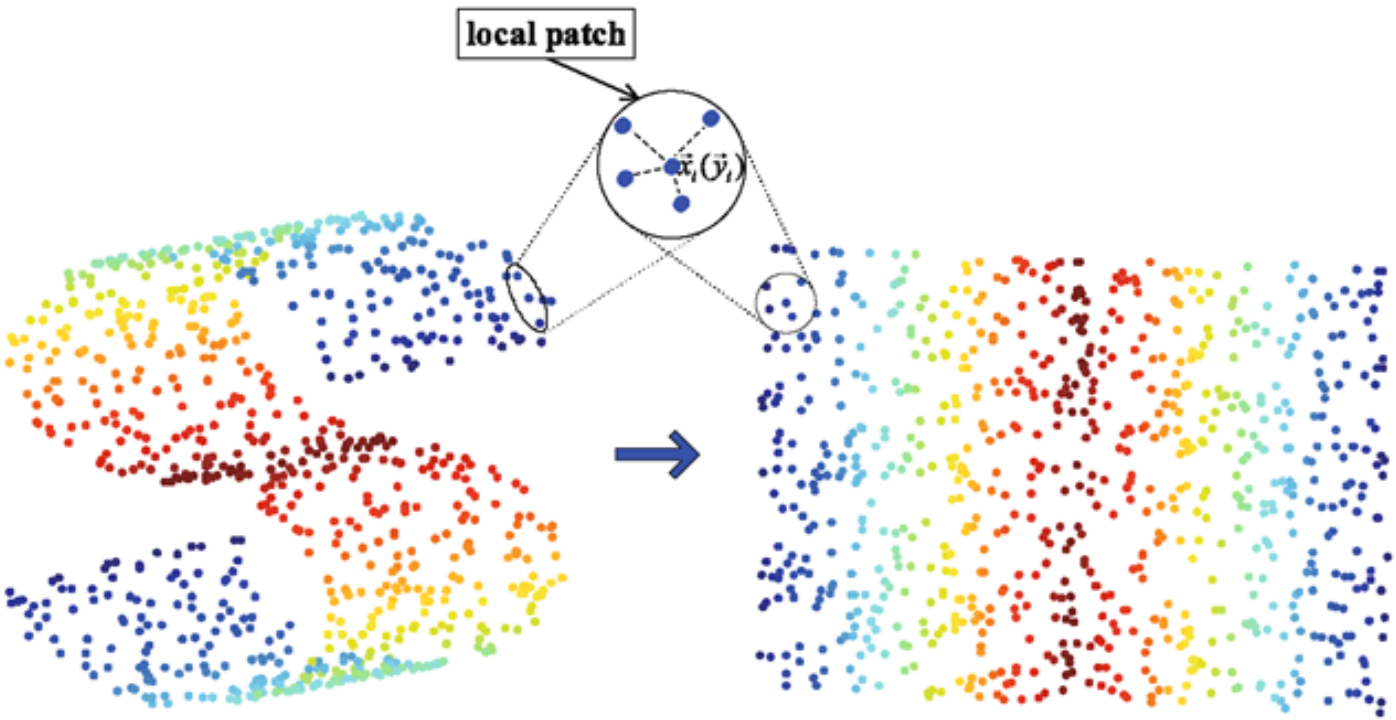
Local Constancy And Smoothness Regularization

- **Example assumption:** The data was generated by the composition of factors or features, potentially at multiple levels in a hierarchy. (core idea in deep learning)
- To a certain point, the exponential advantages conferred by the use of deep, distributed representations counter the exponential challenges posed by the curse of dimensionality.
- Many other generic mild assumptions allow an exponential gain in the relationship between the number of examples and the number of regions that can be distinguished.

Manifold Learning

- A **manifold** is a connected region in space. Mathematically, it is a set of points, associated with a neighborhood around each points.
- From any point, the surface of the manifold appears as a euclidean space.
- Example: We observe the world as a 2-D plane, whereas in fact it is a spherical manifold in 3-D space.

Manifold Learning



Manifold Learning

- Most AI problems seem hopeless if we expect algorithms to learn interesting variations over all of $\mathbb{R}^n$.
- **Manifold Learning**: Most of $\mathbb{R}^n$ consists of invalid input. Interesting input occurs only along a collection of manifolds embedded in $\mathbb{R}^n$.
- Conclusion: probability mass is highly concentrated.

Manifold Learning

- Fortunately, there is evidence to support the above assumptions.
- Observation 1: Probability distributions in natural data (images, text strings, and sound) is highly concentrated.
- Observation 2: Examples encountered in natural data are connected to each other by other examples, with each example being surrounded by similar data.

Manifold Learning

- Training examples from the QMULMultiview Face Dataset.

Conclusion

- Deep learning present a framework to solve tasks that cannot be solved by traditional ML algorithms.
- Next lecture: Feed Forward Neural Networks.