Chapter 5. Machine Learning Basics

Shansuo Liang

August 25th, 2018

Outline

Definition by T. Mitchell

- T. Mitchell: A computer program is said to learn from experience E with respect to some task T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Definition by T. Mitchell

- T. Mitchell: A computer program is said to learn from experience E with respect to some task T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

- The Task, T: Classification, Regression, Transcription, Structured output, Anomaly detection, Denoising, Density estimation, etc.
- The Performance Measure, P: Error rate (the expected 0-1 loss), MSE, etc.
- The Experience, E: Supervised learning, Unsupervised learning, Reinforcement learning (fixed dataset or not), etc.

An Example: Linear Regression

- T: To build a linear system that can take a vector $\mathbf{x} \in \mathbb{R}^n$ as input and predict the value of a scalar $y \in \mathbb{R}^n$ as its output as

$$\hat{y} = \mathbf{w}^T \mathbf{x}, \tag{1}$$

  where $\mathbf{w} \in \mathbb{R}^n$ is a vector of parameters.

- P: To compute MSE of the model on the test set:

$$\mathrm{MSE}_{\mathrm{test}} = \frac{1}{m} \|\hat{\mathbf{y}}^{\mathrm{test}} - \hat{\mathbf{y}}^{\mathrm{test}}\|_2^2. \tag{2}$$

- E: To design an algorithm that will improve the weights $\mathbf{w}$ in a way that reduce $\mathrm{MSE}_{\mathrm{test}}$ when the algorithm can observe a training set $\{\mathbf{X}^{\mathrm{test}}, \mathbf{y}^{\mathrm{test}}\}$. (Solved by the normal equations)

Outline

Capacity, Overfitting and Underfitting

- Training error & Test error (expected):
  1. IID assumptions: Training error $=$ Test error,
  2. The parameters are not fixed: Test error $\geq$ Training error.

Capacity, Overfitting and Underfitting

- Training error & Test error (expected):
  1. IID assumptions: Training error $=$ Test error,
  2. The parameters are not fixed: Test error $\geq$ Training error.

- Two factors correspond to underfitting and overfitting:
  1. Make the training error small,
  2. Make the gap between training and test error small.

Capacity, Overfitting and Underfitting

- Training error & Test error (expected):
  1. IID assumptions: Training error $=$ Test error,
  2. The parameters are not fixed: Test error $\geq$ Training error.
- Two factors correspond to underfitting and overfitting:
  1. Make the training error small,
  2. Make the gap between training and test error small.
- Capacity is defined by the ability to fit a wide variety of functions. We can control whether a model is more likely to be overfit or underfit by altering its capacity.

Capacity, Overfitting and Underfitting

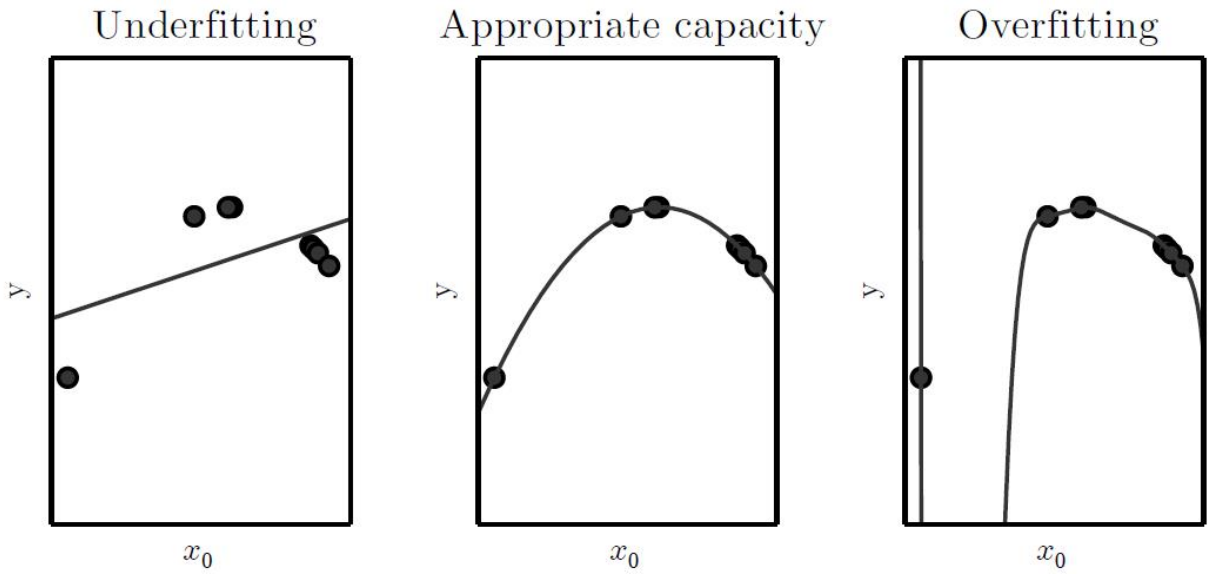


Figure: (Left)A linear function fit to the date suffers from underfitting. (Center)A quadratic function fit to the date generalizes well to unseen points. (Right)A polynomial of degree 9 fit to the data suffers from overfitting.

Capacity, Overfitting and Underfitting

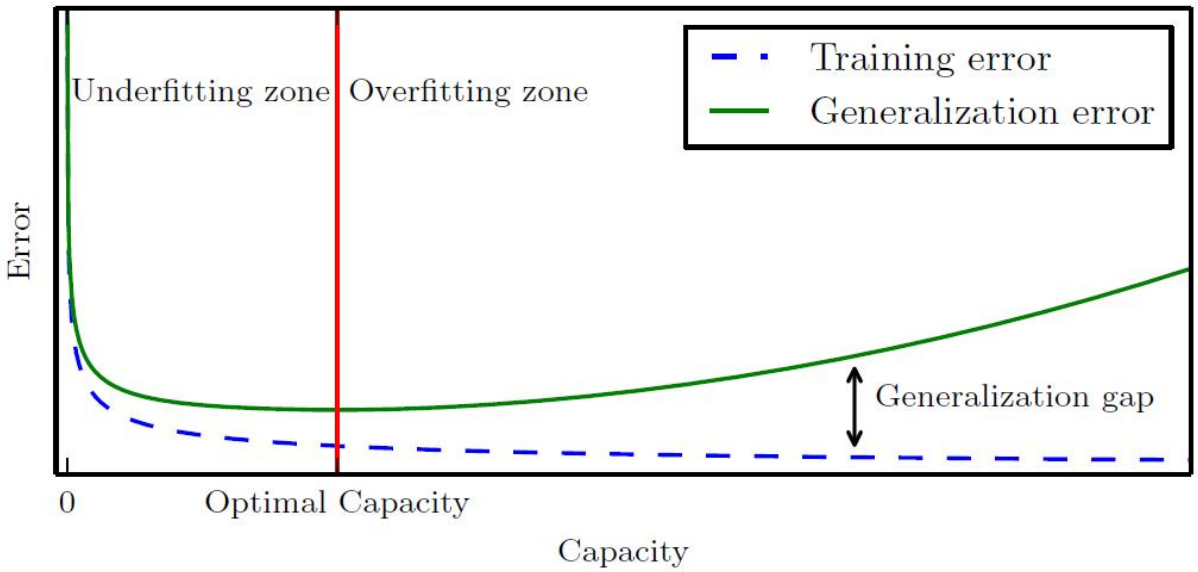


Figure: Typical relationship between capacity and error.

Outline

Hyperparameters

- Hyperparameters: Settings that we can use to control the algorithm's behavior. The values of hyperparameters are not adapted by the learning algorithm itself.

Hyperparameters

- Hyperparameters: Settings that we can use to control the algorithm's behavior. The values of hyperparameters are not adapted by the learning algorithm itself.

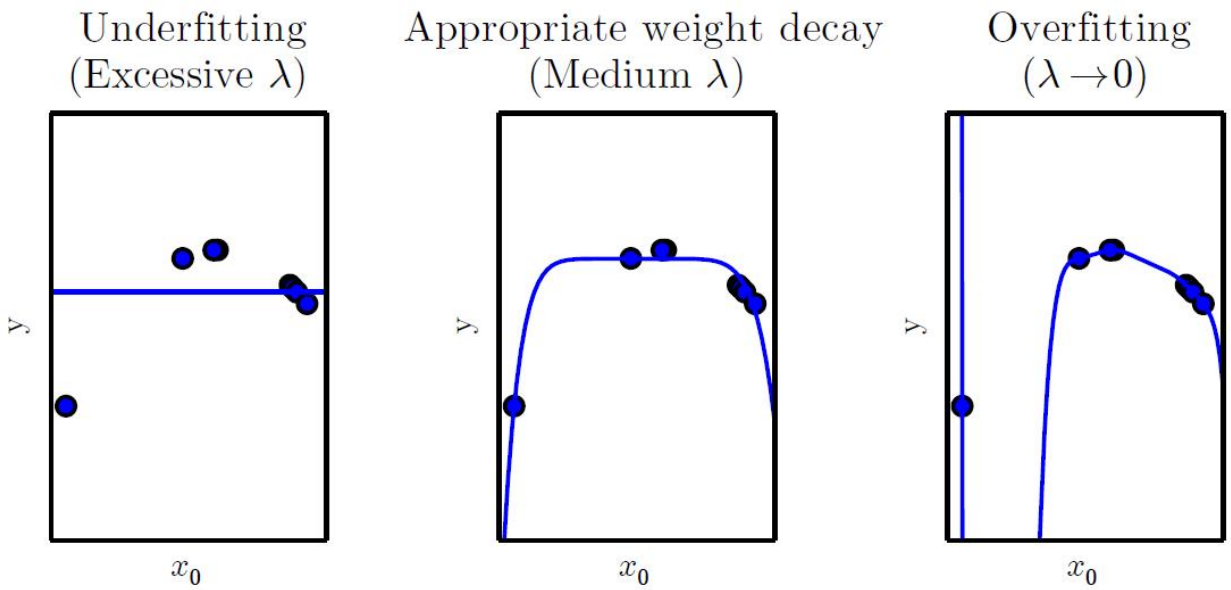


Figure: Example: Fit a high-degree polynomial regression model. The true function is quadratic, but here we use only models with degree 9. We vary the amount of weight decay to prevent these high-degree models from overfitting.

Validation Sets

- A setting is chosen to be a hyperparameter since it is not appropriate to learn that on the training set. If learned on the training set, such hyperparameters would always choose the maximum possible model capacity, resulting in overfitting.

Validation Sets

- A setting is chosen to be a hyperparameter since it is not appropriate to learn that on the training set. If learned on the training set, such hyperparameters would always choose the maximum possible model capacity, resulting in overfitting.
- Validation set: Split the training data into two disjoint subsets, and one of these subsets is used to learn the parameters.

Validation Sets

- A setting is chosen to be a hyperparameter since it is not appropriate to learn that on the training set. If learned on the training set, such hyperparameters would always choose the maximum possible model capacity, resulting in overfitting.
- Validation set: Split the training data into two disjoint subsets, and one of these subsets is used to learn the parameters.
- Cross-validation: When the dataset is small, repeating the training and testing computation on different randomly chosen subsets or splits of the original sets.

Outline

Supervised Learning

- Definition: Roughly speaking, learning algorithms that learn to associate some inputs **x** and some outputs **y**.
  In many cases, the outputs **y** may be difficult to collect and must be provided by a human "supervisor".

Supervised Learning

- Definition: Roughly speaking, learning algorithms that learn to associate some inputs **x** and some outputs **y**.
  In many cases, the outputs **y** may be difficult to collect and must be provided by a human "supervisor".

- Probabilistic interpretation: Most supervised learning algorithms are based on estimating a probability distribution $p(y|\mathbf{x})$. Using ML estimation to find the best parameter vector θ for a parameter family of distribution $p(y|\mathbf{x};\theta)$.

Supervised Learning

- Definition: Roughly speaking, learning algorithms that learn to associate some inputs **x** and some outputs **y**.
  In many cases, the outputs **y** may be difficult to collect and must be provided by a human "supervisor".

- Probabilistic interpretation: Most supervised learning algorithms are based on estimating a probability distribution $p(y|\mathbf{x})$. Using ML estimation to find the best parameter vector θ for a parameter family of distribution $p(y|\mathbf{x};\theta)$.

- Examples: Logistic Regression, Support Vector Machines, $k-NN$, Decision Tree.

Logistic Regression

- Logistic regression or classification: For those with the output $y \in \{0, 1\}$ or other discrete values.

Logistic Regression

- Logistic regression or classification: For those with the output $y \in \{0, 1\}$ or other discrete values.

- Probabilistic interpretation:

  Classes $y \in \{0, 1\}$;
  Function $z = \mathbf{w}^T \mathbf{x}$;
  Hypothesis logistic sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$.

- Decision rules:

$$\hat{y} = \begin{cases} 1, & z > 0 \\ 0, & z \leq 0 \end{cases}.$$

Logistic Regression

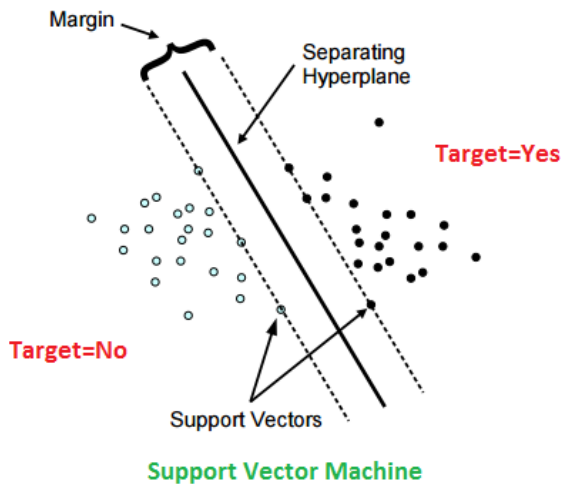
- Define $\pi_i = \sigma(\mathbf{w}^T \mathbf{x}_i)$, then $p(y_i|\mathbf{x}_i, \mathbf{w}) = \pi_i^{y_i} \cdot (1 - \pi_i)^{1-y_i}$.
- Loglikelihood of all data: $\ell(\mathbf{w}) = \sum_i \log p(y_i|\mathbf{x}_i, \mathbf{w})$.
- MLE: $\mathbf{w}^* = \text{argmax}_{\mathbf{w}}\, \ell(\mathbf{w})$.
  1. No close-form solution;
  2. Gradient ascent / SGD (first-order);
  3. Newton-Raphson method (second-order).

Support Vector Machines

- Non-probabilistic interpretation:

$$\begin{aligned} \text{Classes} \quad & y \in \{-1, 1\}; \\ \text{Function} \quad & z = \mathbf{w}^T \mathbf{x} + b; \\ \text{Decision rules} \quad & \hat{y} = sign(z). \end{aligned}$$

`http://www.saedsayad.com/support_vector_machine.htm`

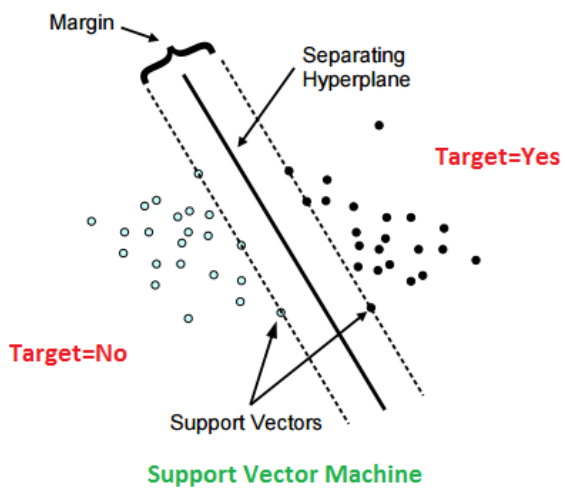


Support Vector Machines

- Non-probabilistic interpretation:

$$\begin{aligned} \text{Classes} \quad & y \in \{-1, 1\}; \\ \text{Function} \quad & z = \mathbf{w}^T \mathbf{x} + b; \\ \text{Decision rules} \quad & \hat{y} = sign(z). \end{aligned}$$

`http://www.saedsayad.com/support_vector_machine.htm`



- Assumption: Data set $\{(\mathbf{x}_i, y_i)\}$ are linearly separable.

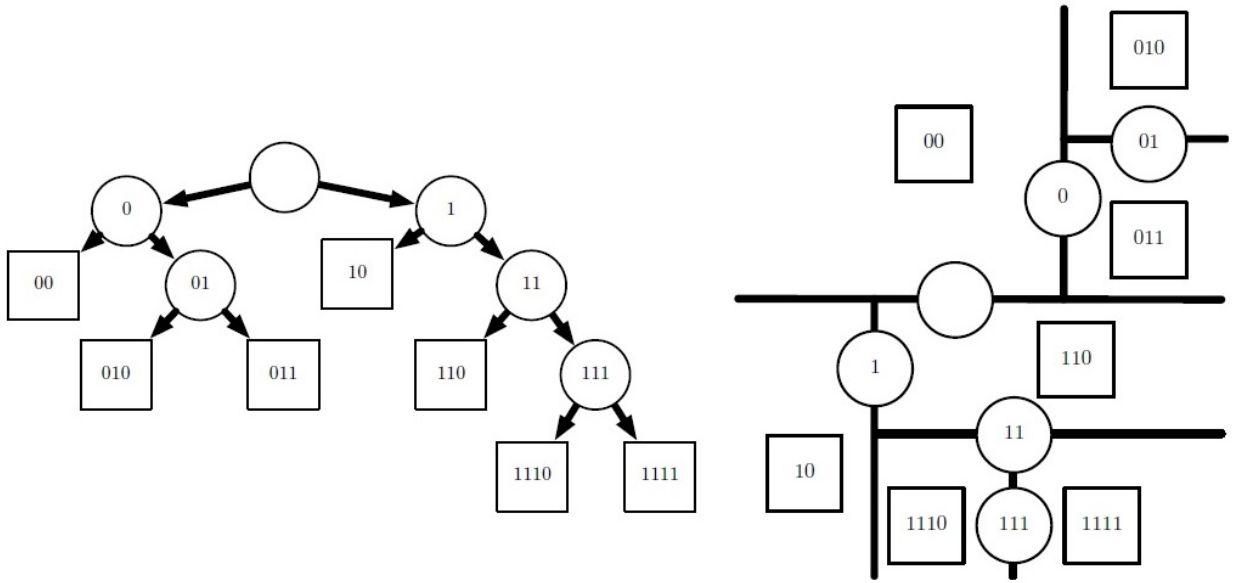
  Margin: Distance from boundary to the closet point in training sets as

$$\gamma = \min_i \frac{|\mathbf{w}^T \mathbf{x}_i + b|}{\|\mathbf{w}\|} = \min_i \frac{y_i(\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|}. \tag{3}$$

  Idea: To maximize the margin. Solved by Quadratic Programming.

k-NN and Decision Tree

- k-nearest neighbors (k-NN) algorithm: When we want to produce an output y for a new test input $\mathbf{x}$, we find the k-nearest neighbors to $\mathbf{x}$ in the training data $\mathbf{X}$. We then return the average of the corresponding y values in the training set.
- Decision Tree:



Unsupervised Learning

- Definition: Informally, unsupervised learning refers to most attempts to extract information from a distribution that do not require human labor to annotate example.

Unsupervised Learning

- Definition: Informally, unsupervised learning refers to most attempts to extract information from a distribution that do not require human labor to annotate example.

- A classic unsupervised learning task is to find the "best" representation of the data.

Unsupervised Learning

- Definition: Informally, unsupervised learning refers to most attempts to extract information from a distribution that do not require human labor to annotate example.
- A classic unsupervised learning task is to find the "best" representation of the data.
- Three common representations
  1. Low-dimensional representations: Compress as much information as possible;
  2. Sparse representations: Entries are mostly zeros;
  3. Independent representations: Disentangle the source of variation underlying the data distribution such that the dimensions of the representation are statistically independent.

Principle component analysis

- PCA learns an orthogonal, linear transformation of the data that projects an input $\boldsymbol{x}$ to a representation $\boldsymbol{z}$.

Principle component analysis

- PCA learns an orthogonal, linear transformation of the data that projects an input $\boldsymbol{x}$ to a representation $\boldsymbol{z}$.

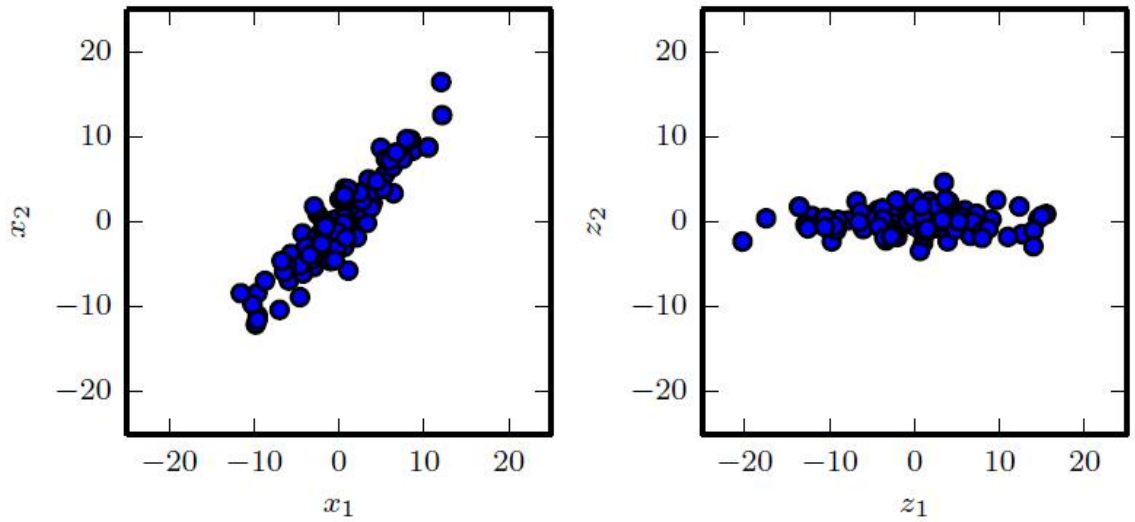


Figure: (Left)The original data consist of samples of $\boldsymbol{x}$. In this space, the variance might occur along directions that are not axis aligned. (Right) The transformed data $\boldsymbol{z} = \boldsymbol{x}^T \boldsymbol{W}$ now varies most along the axis z_1. The direction of second-most variance is now along z_2.

Principle Component Analysis

- Consider a zero-mean data matrx $\boldsymbol{X}$, the covariance matrix associated with $\boldsymbol{X}$ is given by

$$\mathrm{Var}[\boldsymbol{x}] = \frac{1}{m-1}\boldsymbol{X}^T\boldsymbol{X}. \tag{4}$$

  PCA finds a representation $\boldsymbol{z} = \boldsymbol{W}^T\boldsymbol{x}$, where $\mathrm{Var}[\boldsymbol{x}]$ is diagonal.

- The principle components of a design matrix $\boldsymbol{X}$ are given by the eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$.

k-means Clustering

- The $k-$means clustering algorithm divides the training set into k different clusters of examples that are near each other.

k-means Clustering

- The $k-$means clustering algorithm divides the training set into k different clusters of examples that are near each other.

- Initializing k different centroids $\{\boldsymbol{\mu}^{(1)}, ..., \boldsymbol{\mu}^{(k)}\}$, the k-means algorithm alternates between the following two different steps until convergence:

  1. Each training example is assigned to cluster i, where i is the index of the nearest centroid $\boldsymbol{\mu}^{(i)}$.
  2. Each centroid $\boldsymbol{\mu}^{(i)}$ is updated to the mean of all training examples $\boldsymbol{x}^{(i)}$ assigned to cluster i.

Outline

Stochastic Gradient Descent

- The cost function used by a machine learning algorithm often decomposes as a sum over training examples of some per-example loss function, such as

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x},y\sim\hat{p}_{data}} L(\mathbf{x}, y, \boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L(\mathbf{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}), \tag{5}$$

where L is the per-example loss $L(\mathbf{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}) = -\log p(y^{(i)}|\mathbf{x}^{(i)}, \boldsymbol{\theta})$.

Stochastic Gradient Descent

- The cost function used by a machine learning algorithm often decomposes as a sum over training examples of some per-example loss function, such as

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x},y\sim\hat{p}_{data}} L(\mathbf{x}, y, \boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L(\mathbf{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}), \tag{5}$$

where L is the per-example loss $L(\mathbf{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}) = -\log p(y^{(i)}|\mathbf{x}^{(i)}, \boldsymbol{\theta})$.

- Gradient descent requires computing

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L(\mathbf{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}). \tag{6}$$

As the size m grows to billions, a single gradient step becomes computational prohibitive.

Stochastic Gradient Descent

- The insight of SGD is that the gradient is an expectation that may be approximately estimated using a small set of samples.

Stochastic Gradient Descent

- The insight of SGD is that the gradient is an expectation that may be approximately estimated using a small set of samples.
- On each step of the algorithm, we can sample a minibatch of examples $\mathbb{B} = \{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(m')}\}$ drawn uniformly from the training set.
- The estimate of the gradient is formed as

$$\mathbf{g} = \frac{1}{m'} \sum_{i=1}^{m'} \nabla_{\boldsymbol{\theta}} L(\mathbf{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}) \tag{7}$$

The SGD algorithm then follows $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \mathbf{g}$ with ϵ being the learning rate.

Outline

Building a Machine Learning Algorithm

- Recipe: Combine a specification of a dataset, a cost function, an optimization procedure and a model.

Building a Machine Learning Algorithm

- Recipe: Combine a specification of a dataset, a cost function, an optimization procedure and a model.
- For example, the linear regression combines a dataset consisting of $\boldsymbol{X}$ and $\boldsymbol{y}$, the cost function

$$J(\boldsymbol{w}, b) = -\mathbb{E}_{\boldsymbol{x},y\sim\hat{p}_{\text{data}}} \log p_{\text{model}}(y|\boldsymbol{x}), \tag{8}$$

the model specification $p_{\text{model}}(y|\boldsymbol{x}) = \mathcal{N}(y; \boldsymbol{x}^T\boldsymbol{w} + b, 1)$, and the optimization algorithm. is solving the normal equations for the gradient of the cost being zero.

Challenges Motivating Deep Learning

- The Curse of Dimensionality

Challenges Motivating Deep Learning

- The Curse of Dimensionality
- Local Constancy and Smoothness Regularization

Challenges Motivating Deep Learning

- The Curse of Dimensionality
- Local Constancy and Smoothness Regularization
- Manifold Learning

Thanks!

Q&A

Aug 25, 2018