EE628A: Machine Learning Basics

Rensheng Wang

Stevens Institute of Technology

rwang1@stevens.edu

September 14, 2017

Learning Algorithms

- A machine learning algorithm is an algorithm able to learn from data.
- What is learning?

  "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."
- Task T
  - Classification:

$$f: \quad \mathcal{R}^n \rightarrow \{1, 2, \cdots, k\}$$

  - Classification with missing inputs

Learning Algorithms

- Task T
  - Regression:

$$f: \quad \mathcal{R}^n \rightarrow \mathcal{R}$$

  - Machine translation
  - Anomaly detection

Learning Algorithms

- The Performance Measure, P
  - accuracy vs. error rate
  - validation from test data
  - The choice of performance measure may seem straightforward and objective, but it is often difficult to choose
- The experience, E
  - unsupervised

    Unsupervised learning algorithms experience a dataset containing many features, then learn useful properties of the structure of this dataset.
  - supervised

    Supervised learning algorithms experience a dataset containing features, but each example is also associated with a label or target.
  - reinforcement learning

    Reinforcement learning algorithms interact with an environment, a feedback loop between the learning system and its experiences.

Example: Linear Regression

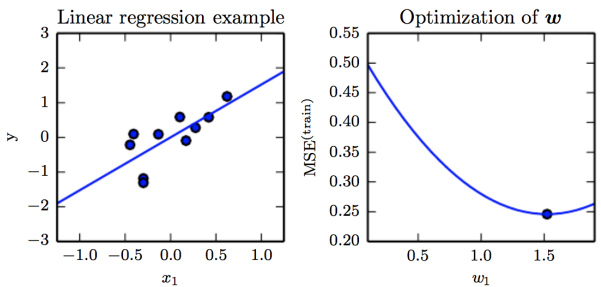
- Linear regression is to build a system that can take a vector $\mathbf{x} \in \mathcal{R}^n$ as input and predict the value of a scalar $y \in \mathcal{R}$ as its output.
- Let $\hat{y}$ be the value that our model predicts y should take on

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

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

- Performance measure of model: Mean Squared Errors

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^{m} (\hat{\mathbf{y}} - \mathbf{y})_i^2$$



Capacity, Overtting and Undertting

- The learning should be generalizations of experiences from data
- Measure model performance on a test set of examples separated from training
- One assumption: Training set & testing set identically distributed
- How to evaluate a good machine learning algorithm
  - Make the training error small
  - Make the gap between training and test error small
- Underfitting

  Undertting occurs when the model is not able to obtain a sufficiently low error value on the training set
- Overfitting

  Overtting occurs when the gap between the training error and test error is too large

Capacity, Overtting and Undertting

- A models capacity is its ability to fit a wide variety of functions.
- Models with low capacity may struggle to fit the training set.
- Models with high capacity can overt by memorizing properties of the training set that do not serve them well on the test set.
- One way to control the capacity of a learning algorithm is by choosing its hypothesis space.
- We can generalize linear regression to include polynomials

  from

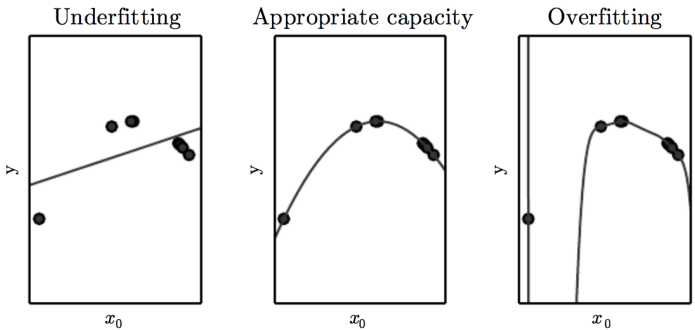
$$\hat{y} = b + wx$$

  to

$$\hat{y} = b + w_1 x + w_2 x^2$$

Capacity, Overtting and Undertting

- Model fitting



Regularization

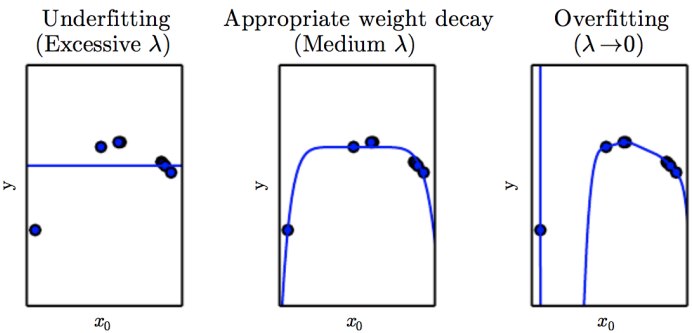
- We build a set of preferences into the learning algorithm.
- For instance, we can increase or decrease the models representational capacity by adding or removing functions from the hypothesis space of solutions the learning algorithm is able to choose.
- The behavior of our algorithm is strongly affected not just by how large we make the set of functions allowed in its hypothesis space, but by the specific identity of those functions.
- Learning algorithm with preference means, when both functions are eligible, but one is preferred.
- For example, we can modify the training criterion for linear regression to include **weight decay**.
- To perform linear regression with weight decay, we minimize a sum comprising both the mean squared error on the training and a criterion $J(\mathbf{w})$ that expresses a preference for the weights to have smaller squared L^2 norm.

$$J(\mathbf{w}) = \mathsf{MSE}_{\mathsf{train}} + \lambda \mathbf{w}^T \mathbf{w}$$

where λ is a value chosen ahead of time that controls the strength of our preference for smaller weights.

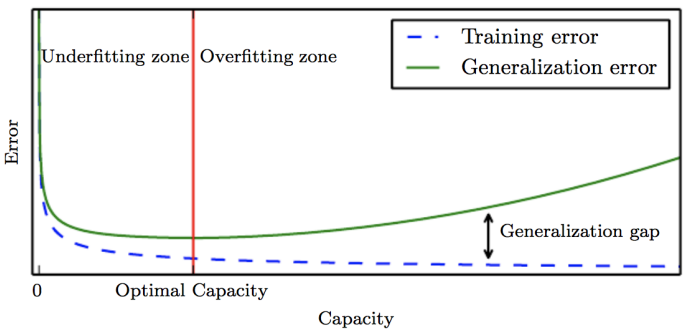
Regularization

- $\lambda = 0$, we impose no preference.
- Larger λ forces the weights to become smaller.
- Minimizing $J(\mathbf{w})$ results in a choice of weights that make a tradeoff between fitting the training data and being small.
- This gives us solutions that have a smaller slope, or put weight on fewer of the features.
- As an example of how we can control a models tendency to overt or undert via weight decay, we can train a high-degree polynomial regression model with different values of λ.



Regularization

- More generally, we can regularize a model that learns a function $f(\mathbf{x}; \boldsymbol{\theta})$ by adding a penalty called a regularizer to the cost function.
- In aforementioned example, the regularizer is $\Omega(\mathbf{w}) = \mathbf{w}^T\mathbf{w}$
- Regularization is any modication we make to a learning algorithm that is intended to reduce its generalization error but not its training error.



Validation

- To overcome the overfitting problem, we need a validation set of examples that training algorithm does not observe.
- We can hold-out some training data set for validation test, for example, 80% of training samples for training and the remaining 20% for validation.
- It is important that the test examples are not used in any way to make choices about the model.
- When data set is not large enough, we can use k-fold cross-validation procedure.
- Randomly split the dataset into k non-overlapping subsets. On trail i, the $i-$th subset of the data is used as the test set and the rest is used as the training set.

Bayesian Statistics

- The Bayesian analysis uses probability to reflect degrees of certainty of states of knowledge.
- The dataset is directly observed and so is not random.
- On the other hand, the true parameter $\boldsymbol{\theta}$ we are interested is unknown or uncertain and thus is represented as a random variable.
- Before observing the data, we represent our knowledge of $\boldsymbol{\theta}$ using the prior probability distribution, $p(\boldsymbol{\theta})$.
- Let us assume we have a set of data observations, or samples $\{x^{(1)}, \cdots, x^{(m)}\}$. We can recover the effect of data on our belief about $\boldsymbol{\theta}$ by combining the data likelihood $p(x^{(1)}, \cdots, x^{(m)}|\boldsymbol{\theta})$ with the prior via Bayes' rule:

$$p(\boldsymbol{\theta}|x^{(1)}, \cdots, x^{(m)}) = \frac{p(x^{(1)}, \cdots, x^{(m)}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(x^{(1)}, \cdots, x^{(m)})}$$

Bayesian Linear Regression

- In linear regression, we learn a linear mapping from an input vector $\mathbf{x} \in \mathcal{R}^n$ to predict the value of a scalar $y \in \mathcal{R}$
- The prediction is parametrized by the vector $\mathbf{w} \in \mathcal{R}^n$

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

- Given a set of m training samples $\left(\mathbf{X}^{(\text{train})}, \mathbf{y}^{(\text{train})}\right)$, we can express the prediction of y over the entire training set as:

$$\hat{\mathbf{y}}^{(\text{train})} = \mathbf{X}^{(\text{train})} \mathbf{w}$$

- Expressed as a Gaussian conditional distribution on $\mathbf{y}^{(\text{train})}$, we have

$$p(\mathbf{y}^{(\text{train})} | \mathbf{X}^{(\text{train})}, \mathbf{w}) = \mathcal{N}(\mathbf{y}^{(\text{train})};\ \mathbf{X}^{(\text{train})} \mathbf{w}, \mathbf{I})$$
$$\propto \exp\left(-\frac{1}{2}\left(\mathbf{y}^{(\text{train})} - \mathbf{X}^{(\text{train})} \mathbf{w}\right)^T \left(\mathbf{y}^{(\text{train})} = \mathbf{X}^{(\text{train})} \mathbf{w}\right)\right)$$

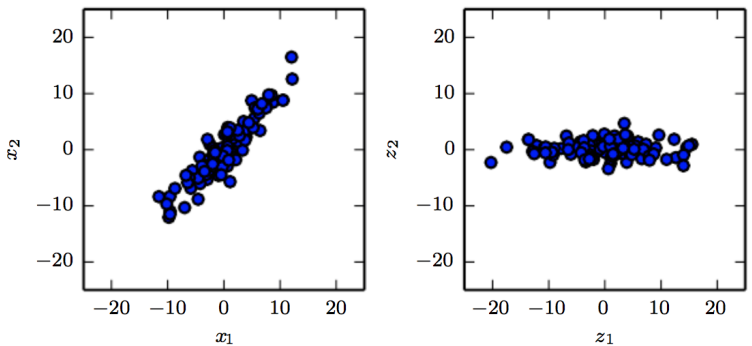
where we assume the Gaussian variance on y is $\mathbf{1}$.

Supervised Learning Algorithms

- Probabilistic Supervised Learning:
- Support Vector Machines
- Decision Tree

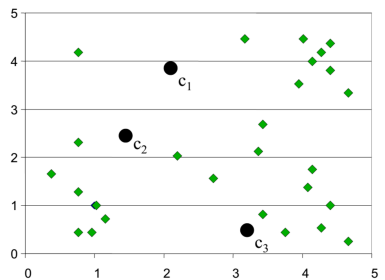
Unsupervised Learning Algorithms

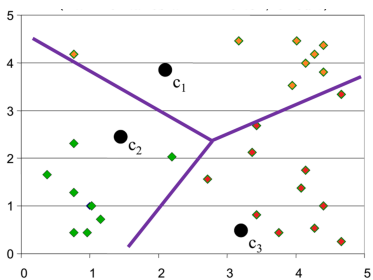
- **Principal Components Analysis**
- In below plot, PCA learns a linear projection that aligns the direction of greatest variance with the axes of the new space.



Unsupervised Learning Algorithms

- **k-means Clustering**
- The k-means clustering algorithm divides the training set into k different clusters of examples that are near each other.
- The k-means algorithm works by initializing k different centroids $\{\mathbf{c}^{(1)}, \cdots, \mathbf{c}^{(k)}\}$ to different values, then alternating between two different steps until convergence.
- In one step, each training example is assigned to cluster i, where i is the index of the nearest centroid $\mathbf{c}^{(i)}$.
- In the other step, each centroid $\mathbf{c}^{(i)}$ is updated to the mean of all training examples $\mathbf{x}^{(j)}$ assigned to cluster i.





Stochastic Gradient Descent (SGD)

- Why SGD?

  A recurring problem in machine learning is that large training sets are necessary for good generalization, but large training sets are also more computationally expensive.

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

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

- When m is huge, like, millions or billions of samples, the time to take a single gradient step becomes prohibitively long.

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

- Specifically, on each step of the algorithm, we can sample a **mini-batch** of examples $\mathcal{B} = \{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(m')}\}$ drawn uniformly from the training set, where $m' \ll m$ chosen to be a relatively small number of examples.

Stochastic Gradient Descent (SGD)

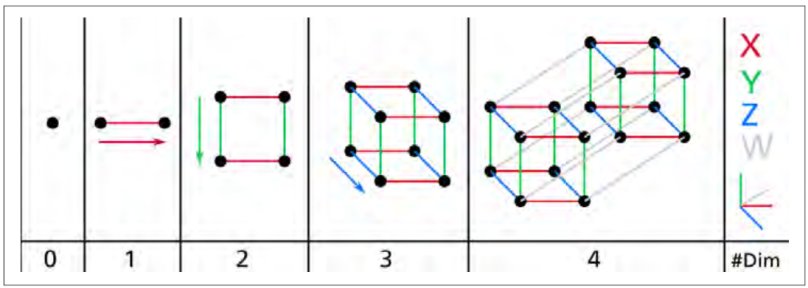
- Stochastic gradient descent has many important uses outside the context of deep learning.
- It is the main way to train large linear models on very large data sets. For a fixed model size, the cost per SGD update does not depend on the training set size m.
- Prior to the advent of deep learning, the main way to learn nonlinear models was to use the kernel trick in combination with a linear model.
- Deep learning garnered additional interest in industry, because it provided a scalable way of training nonlinear models on large data sets.

Challenges Motivating Deep Learning

- Conventional machine learning algorithms have not succeeded in solving the central problems in AI, such as recognizing speech or recognizing objects in images.
- The development of deep learning was motivated in part by the failure of traditional algorithms to generalize well on such AI tasks.
- **The Curse of Dimensionality:** Many machine learning problems become exceedingly difficult when the number of dimensions in the data is high.
- **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 of these implicit "priors" is the smoothness prior or local constancy prior. This prior states that the function we learn should not change very much within a small region.

The Curse of Dimensionality

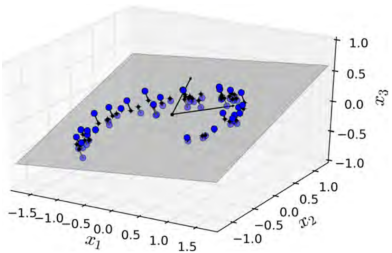
- We are familiar with space with low-dimensions. However, it turns out that many things behave very differently in high-dimensional space.



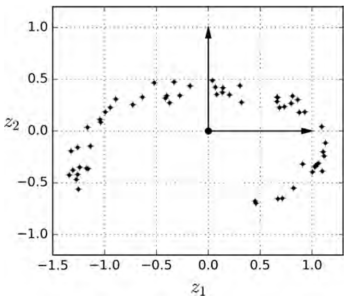
- For example, if you pick a random point in a unit square (a 1×1 square), it will have only about a 0.4% chance of being located less than 0.001 from a border.
- But in a 10,000-dimensional unit hypercube (a $1 \times 1 \times \cdots \times 1$ cube, with ten thousand 1s), this probability is greater than 99.999999%.

Main Approaches for Dimensionality Reductiony

- Projection



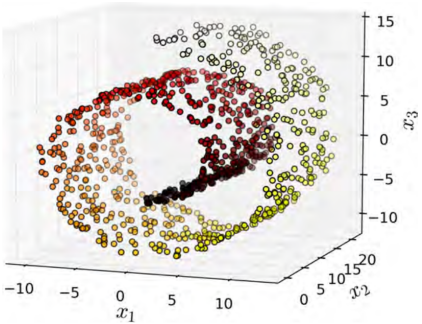
(a)before projection - 3D



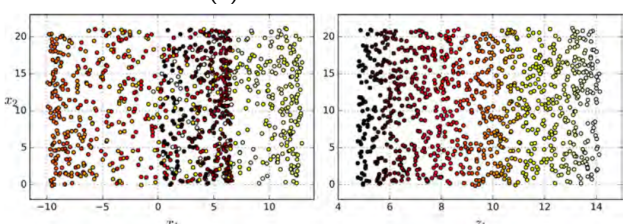
(b) after projection - 2D.

Main Approaches for Dimensionality Reductiony

- Manifold Learning



(a) swiss roll dataset



(b) projection (c) unrolling

The Curse of Dimensionality

- What are the main motivations for reducing a datasets dimensionality?
- What is the curse of dimensionality?
- Once a datasets dimensionality has been reduced, is it possible to reverse the operation?
- How can you evaluate the performance of a dimensionality reduction algorithm on your dataset?