Large Scale Data Analysis Using Deep Learning

Machine Learning Basics - 2

U Kang
Seoul National University

In This Lecture

- Estimators, bias, and variance
- Maximum likelihood estimation
- Supervised learning algorithm
- Stochastic Gradient Descent (SGD)
- Challenges motivating deep learning

Estimators

- Point estimation: the attempt to provide the single "best" prediction of some quantity of interest

- Let $\{x^{(1)}, \ldots, x^{(m)}\}$ be a set of m independent and identically distributed (i.i.d.) data points. A point estimator or statistic is any function of the data: $\widehat{\theta_m} = g(x^{(1)}, \ldots, x^{(m)})$
  - E.g., estimate the average of a random variable

Bias

- Bias of an estimator is defined as $bias(\widehat{\theta_m}) = E(\widehat{\theta_m}) - \theta$
  - An estimator $\widehat{\theta_m}$ is unbiased if $bias(\widehat{\theta_m}) = 0$
- E.g., Bernoulli Distribution: $P(x^{(i)}; \theta) = \theta^{x^{(i)}}(1-\theta)^{(1-x^{(i)})}$
  - A common estimator for θ is given by $\widehat{\theta_m} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$
  - Is $\widehat{\theta_m} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$ biased?

Variance

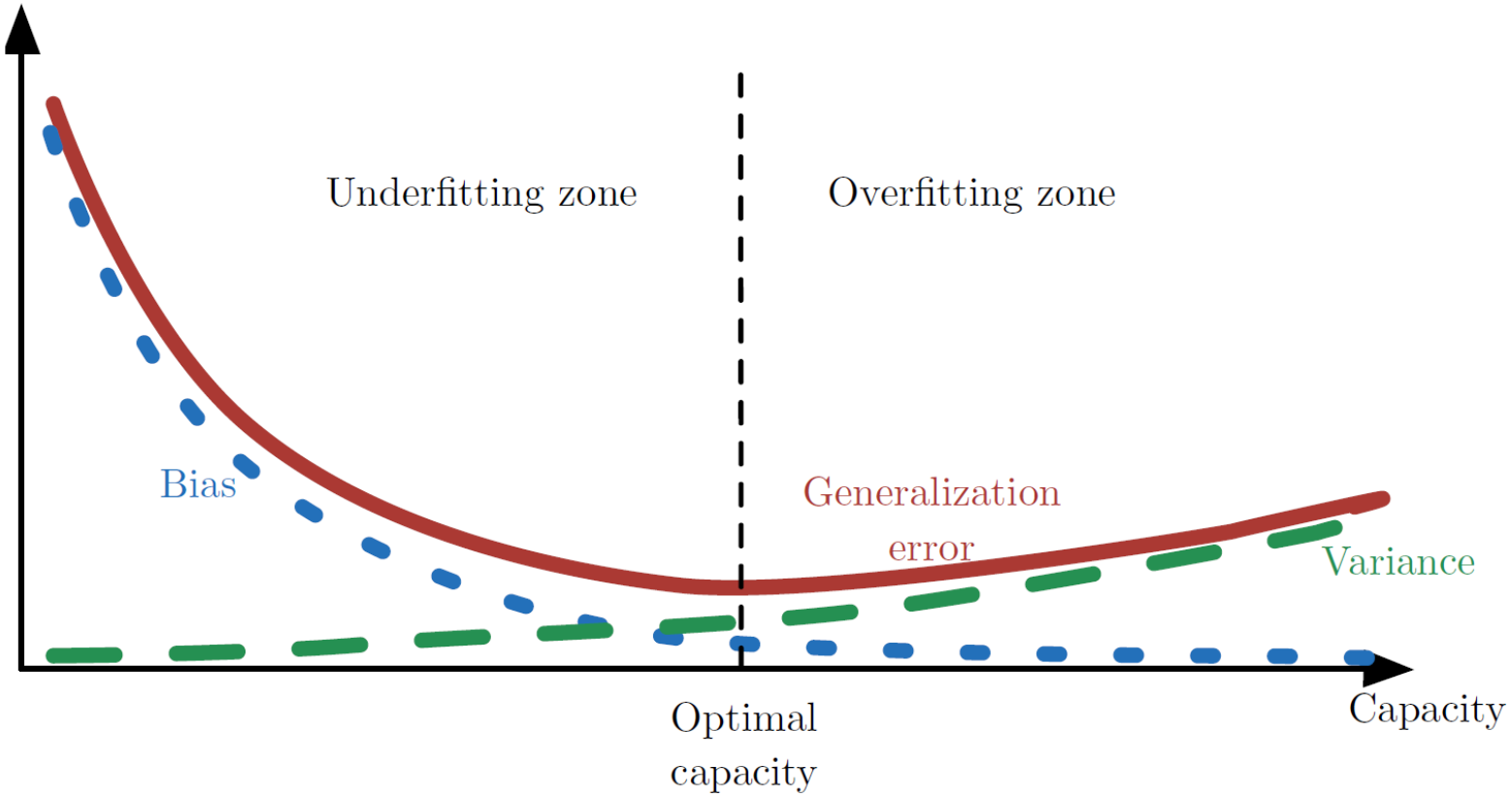
- Variance of an estimator is defined as in the variance of a variable
  - $Var(\hat{\theta}) = E\left[(\hat{\theta} - E[\hat{\theta}])^2\right]$
- Square root of the variance is called the standard error $\mathrm{SE}(\hat{\theta})$
- We want an estimator to have a low variance

Bias – Variance Tradeoff

- The mean squared error (MSE) is tightly connected to bias and variance (called bias-variance decomposition)
  - $\text{MSE} = E\left[(\widehat{\theta_m} - \theta)^2\right] = (bias(\widehat{\theta_m}))^2 + Var(\widehat{\theta_m})$
  - (Pf)
    - $(bias(\widehat{\theta_m}))^2 = (E\widehat{\theta_m} - \theta)^2 = (E\widehat{\theta_m})^2 - 2\theta E\widehat{\theta_m} + \theta^2$
    - $Var(\widehat{\theta_m}) = E(\widehat{\theta_m}^2) - (E\widehat{\theta_m})^2$

- The tradeoff suggests that to minimize MSE we should consider both bias and variance

Bias and Variance



Maximum Likelihood Estimation

- Consider a set of m examples $X = \{x^{(1)}, \ldots, x^{(m)}\}$ drawn from the true but unknown data generating distribution $p_{data}(x)$
- Let $p_{model}(x; \theta)$ be a parametric family of our model distribution
- The maximum likelihood estimator for θ is defined as
  - $\theta_{ML} = argmax_\theta \, p_{model}(X; \theta)$

    $= argmax_\theta \, \prod_{i=1}^{m} p_{model}(x^{(i)}; \theta)$

    $= argmax_\theta \, \sum_{i=1}^{m} \log p_{model}(x^{(i)}; \theta)$

    $= argmax_\theta \, E_{x \sim \hat{p}_{data}} \log p_{model}(x; \theta)$

    where $\hat{p}_{data}$ is the empirical distribution defined by the training data

  - E.g., estimating mean of a Gaussian

Maximum Likelihood Estimation

- The maximum likelihood estimator (MLE) for θ is defined as
  - $\theta_{ML} = argmax_\theta \, E_{x \sim \hat{p}_{data}} \log p_{model}(x; \theta)$
- MLE is equivalent to minimizing the dissimilarity between the empirical distribution $\hat{p}_{data}$ and the model distribution in terms of KL divergence
  - $D_{KL}(\hat{p}_{data} || p_{model}) = E_{x \sim \hat{p}_{data}}[\log \hat{p}_{data}(x) - \log p_{model}(x)]$
  - Minimizing the above KL divergence by training the model (finding the best parameters) is equivalent to minimizing $-E_{x \sim \hat{p}_{data}}[\log p_{model}(x)]$
- Note that minimizing $D_{KL}(P||Q)$ with regard to Q is equivalent to mining the cross entropy H(P,Q) with regard to Q
  - Thus, minimizing cross entropy is equal to finding MLE
- Maximum likelihood is an attempt to make the model distribution match the empirical distribution $\hat{p}_{data}$
  - Ideally, we would like to match p_{data}, but we do not really know it

Conditional Log-Likelihood

- The maximum likelihood estimator (MLE) can be readily generalized to the case where our goal is to estimate a conditional probability $\mathrm{P}(\mathrm{y}|\mathrm{x};\theta)$ in order to predict y given x
- This is in fact the most common situation because it forms the basis for most supervised learning
- The conditional maximum likelihood estimator is given by
  - $\theta_{ML} = argmax_\theta \, P(Y \mid X; \theta)$
  - If the examples are assumed to be i.i.d., then
  - $\theta_{ML} = argmax_\theta \sum_{i=1}^{m} \log P\left(y^{(i)}|x^{(i)};\theta\right)$

Bayesian Statistics

- We have discussed frequentist statistics: we want to estimate a fixed but unknown θ
- Bayesian statistics: the parameter θ is not fixed; it is a random variable
- Before observing the data, we represent our knowledge of θ using the prior probability distribution p(θ)
- After observing the data, our belief on θ changes
  - $P(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$
- Prediction using Bayesian estimation is different from that of maximum likelihood estimation
  - $P\left(x^{(m+1)}\right) = \int p\left(x^{(m+1)}\middle|\theta\right) p\left(\theta\middle|x^{(1)..(m)}\right) d\theta$
  - That is, we use the full posterior distribution of θ

Maximum A Posteriori (MAP) Estimation

- Although prediction in Bayesian estimation uses full posterior distribution of θ, it is often desirable to have a single point estimate
  - Why? Many operations involving Bayesian posterior is intractable
  - A point estimate offers a tractable approximation

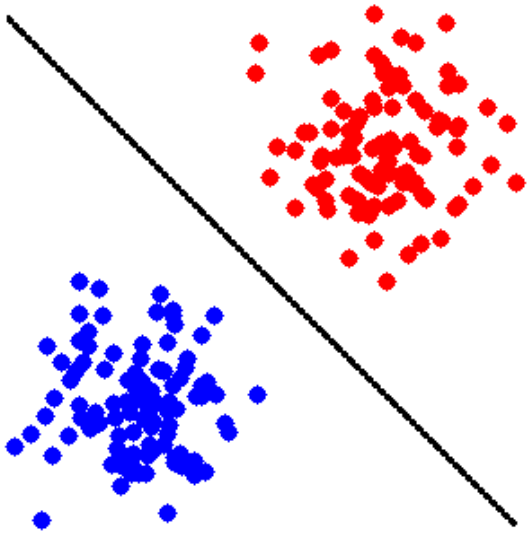
- MAP (Maximum a Posteriori): chooses the point of maximal posterior probability
  - $\theta_{MAP} = argmax_\theta \, p(\theta|x) = argmax_\theta \log p(x|\theta) + \log p(\theta)$

Supervised Learning Algorithms

- Logistic Regression
- Support Vector Machine
- K Nearest Neighbor
- Decision Tree

Logistic Regression

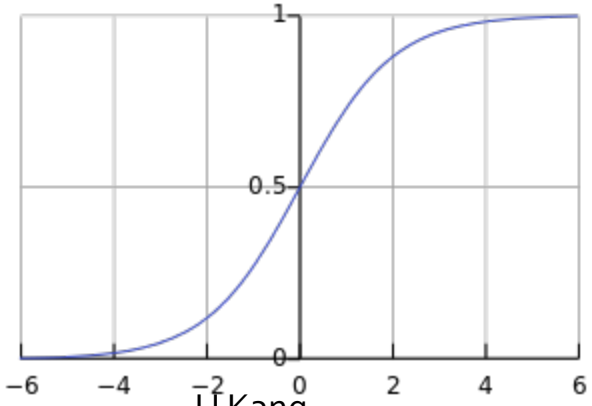
- Probabilistic supervised learning: estimate $p(y|\boldsymbol{x})$
  - We can do this by finding the best parameter vector $\boldsymbol{\theta}$ for a parametric family of distributions $p(y|\boldsymbol{x};\boldsymbol{\theta})$
- Linear classifier



- I.e., x belongs to class +1 if $w^T x > 0$, and to class -1 if $w^T x < 0$

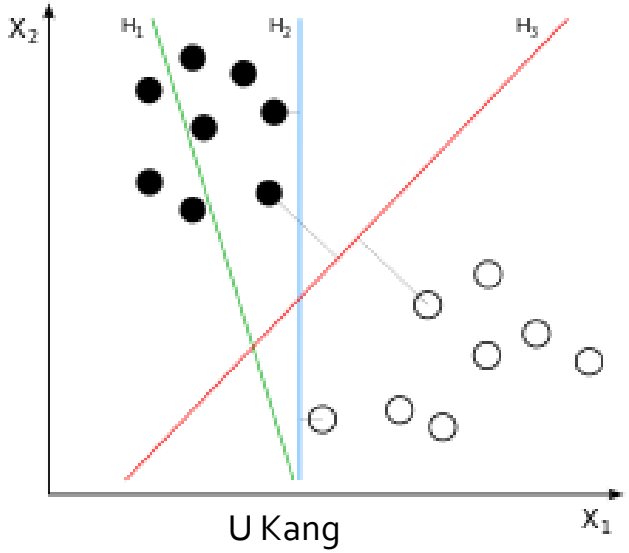
Logistic Regression (LR)

- Logistic regression is a linear classifier
  - y has two labels (1 and -1)
- Intuitively, we want p(y=1|x) be large if $w^T x$ is large
  - Also, we want p(y=1|x) be small if $w^T x$ is small
  - Logistic regression uses sigmoid function $\boldsymbol{\sigma}$ to squash $w^T x$ into a probability in [0,1]
- The probability of y=1 in logistic regression is given by
  - $p(y = 1|\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{\sigma}(\boldsymbol{\theta^T x}) = \frac{\mathbf{1}}{\mathbf{1}+\boldsymbol{exp}(-\boldsymbol{\theta^T x})}$



Support Vector Machine (SVM)

- SVM is similar to logistic regression in that the class is determined by a linear function $w^T x$
- However, unlike LR, SVM does not provide probabilities, but only outputs a class identity
- SVM predicts that y=+1 if $w^T x > 0$. Likewise it predicts y=-1 if $w^T x < 0$
- SVM is called maximum margin classifier



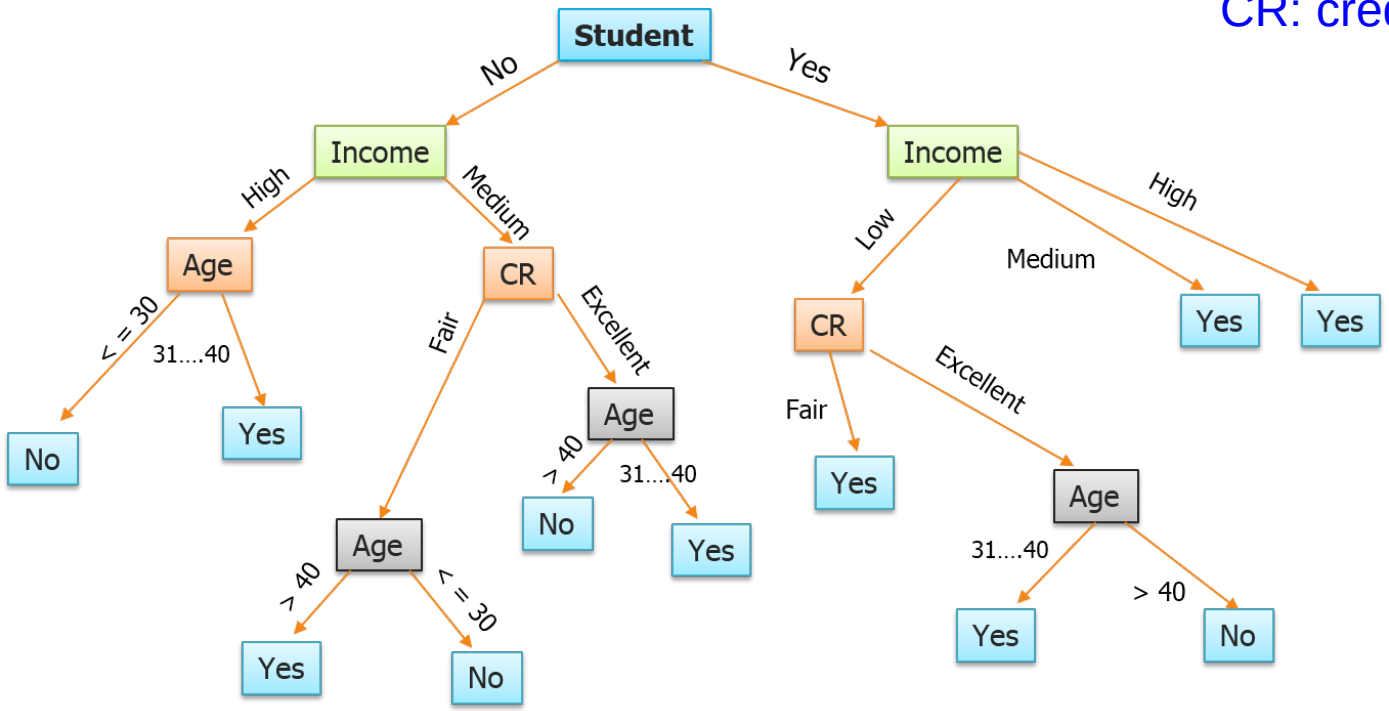
K Nearest Neighbor (K-NN)

- Can be used for classification or regression
- Given a new test input x, we want to produce an output y
- We find the k nearest neighbors of x in the training data X
- We then return the average of the corresponding y values
  - For classification, we compute the average of one-hot code vectors with c_y=1 and c_i=0 for all other values of i
- K-NN is a nonparametric learning algorithm: it can achieve very high capacity
  - Given a large training set, K-NN gives high accuracy
- Disadvantage of K-NN
  - It has a high computational cost
  - It may generalize very badly given a small, finite training set
  - It cannot learn that one feature is more discriminative than another

Decision Trees

- Breaks the input space into regions
- E.g., buy_computer ?

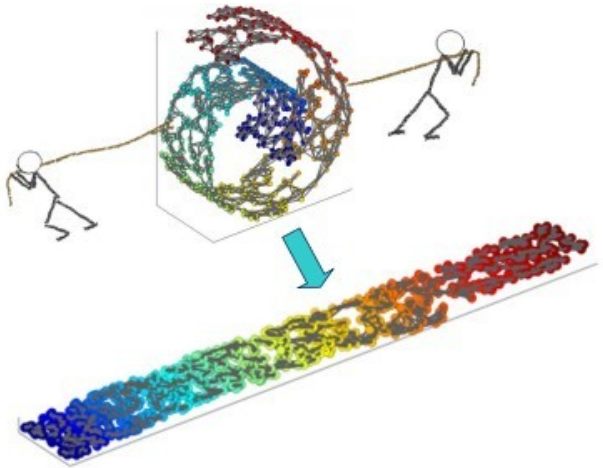
CR: credit rating



- Limitation: it cannot correctly find the non-axis-aligned decision boundary

Unsupervised Learning Algorithms

- Unsupervised algorithms experience only “features”, but not a supervised signal
- Examples
  - Density estimation
  - Learn to denoise data from some distribution
  - Finding a manifold that the data lies near
  - Clustering the data into groups of related examples



Manifold learning

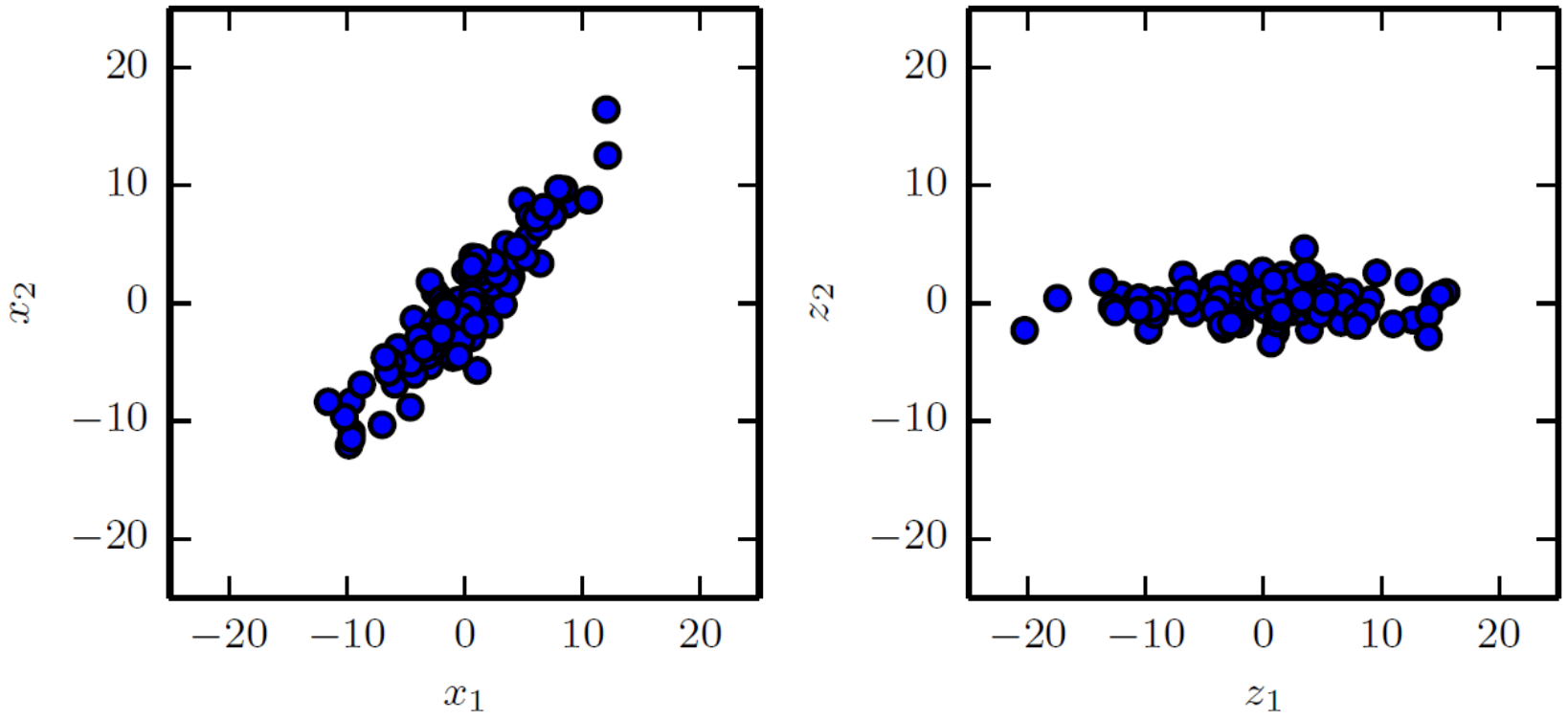
Unsupervised Learning Algorithms

- Classic unsupervised learning task: find the 'best' representation of the data
  - 'best' means that we look for a simpler representation while preserving as much information of the original data

- 3 common ways of defining a simple representation
  - Low-dimensional representation
  - Sparse representation
  - Independent representation
    - Disentangle the sources of variation underlying the data distribution such that the dimensions of the representation are statistically independent

Principal Component Analysis

- PCA disentangles the factors of variation underlying the data



k-means Clustering

- k-means algorithm divides the training set into k different clusters of examples that are near each other
- k-means algorithm
  - Initialize k different centroids $\{\mu^{(1)}, \ldots, \mu^{(k)}\}$
  - Until convergence
    - Assign each training example to cluster i where i is the index of the nearest centroid $\mu^{(i)}$
    - Each centroid $\mu^{(i)}$ is updated to the mean of all training examples assigned to cluster i

Stochastic Gradient Descent (SGD)

- A recurring problem in ML: large training sets are necessary for good generalization, but large training sets are more computationally expensive

- Cost function using cross-entropy:
  - $J(\theta) = E_{x,y\sim\hat{p}_{data}} L(x, y, \theta) = \frac{1}{m}\sum_{i=1}^{m} L(x^{(i)}, y^{(i)}, \theta)$

    where L is the per-example loss $L(x, y, \theta) = -\log p(y|x; \theta)$
  - Gradient descent requires computing $\nabla_\theta J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta L(x^{(i)}, y^{(i)}, \theta)$
  - The computational cost of the gradient descent is O(m) which can take long for large m

- Insight of SGD: gradient is an expectation which can be approximately estimated using a small set of samples

Stochastic Gradient Descent (SGD)

- SGD
  - We sample a minibatch of examples $B = \{x^{(1)}, \ldots, x^{(m')}\}$ drawn uniformly from the training set
  - The minibatch size m' is typically a small number (1 to few hundred)
  - m' is usually fixed as the training set size m grows
  - The estimate of the gradient is $g = \frac{1}{m'} \sum_{i=1}^{m'} \nabla_\theta L(x^{(i)}, y^{(i)}, \theta)$
  - Then the gradient descent is given by $\theta \leftarrow \theta - \epsilon g$

- SGD works well in practice
  - I.e., it often finds a very low value of the cost function quickly enough to be useful

Building a Machine Learning Algorithm

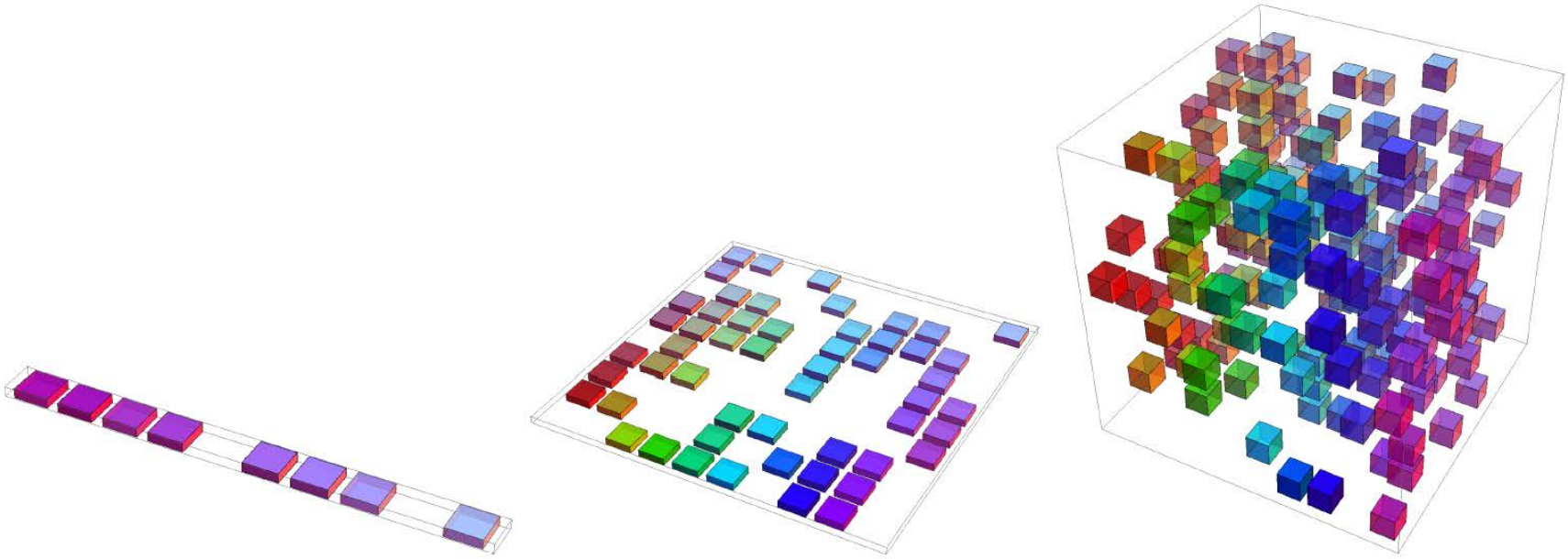
- Simple recipe: combine a specification of
  - a dataset
  - a cost function
    - The most common one is the negative log-likelihood (or cross entropy)
  - an optimization procedure, and
  - a model: p(y|x)

Challenges Motivating Deep Learning

- The simple ML algorithms discussed work well on many problems; however they failed in solving the central problems in AI, such as speech recognition or object recognition
- Deep learning is motivated in part by the failure
- Challenges of generalizing to new examples becomes exponentially more difficult when working with high-dimensional data
  - Curse of dimensionality
  - Local constancy and smoothness regularization
  - Manifold learning

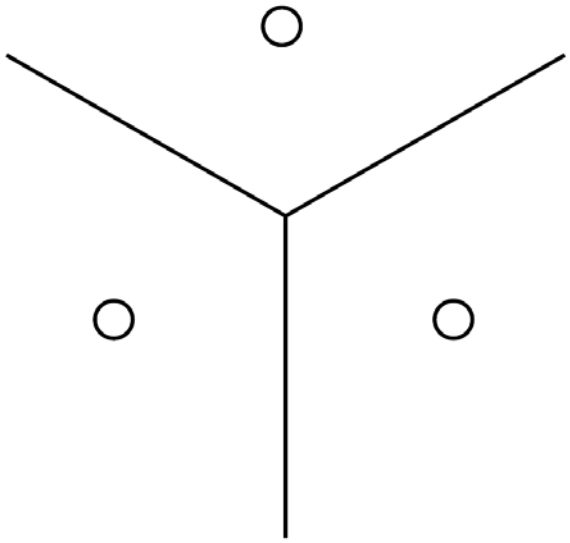
Curse of Dimensionality

- The number of possible distinct configurations of a set of variables increases exponentially as the number of variables increases



Local Constancy and Smoothness Regularization

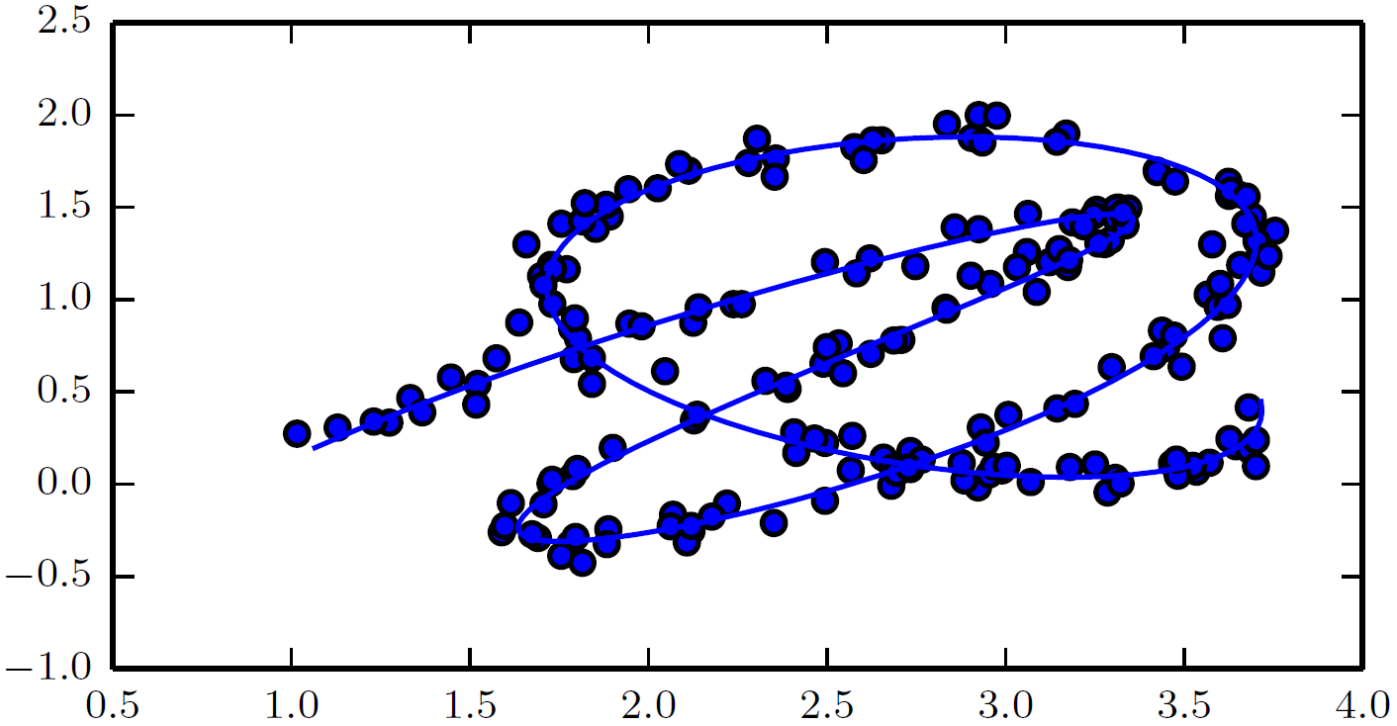
- In order to generalize well, ML algorithms need to be guided by prior beliefs about what kind of function they should learn
- Widely used priors: smoothness prior or local constancy prior
  - The function we learn should not change very much within a small region
  - To distinguish O(k) regions in input space, we would require O(k) examples
  - For very high dimensional data, the required number of examples would be too large



Nearest neighbor algorithm breaks up the input space into regions

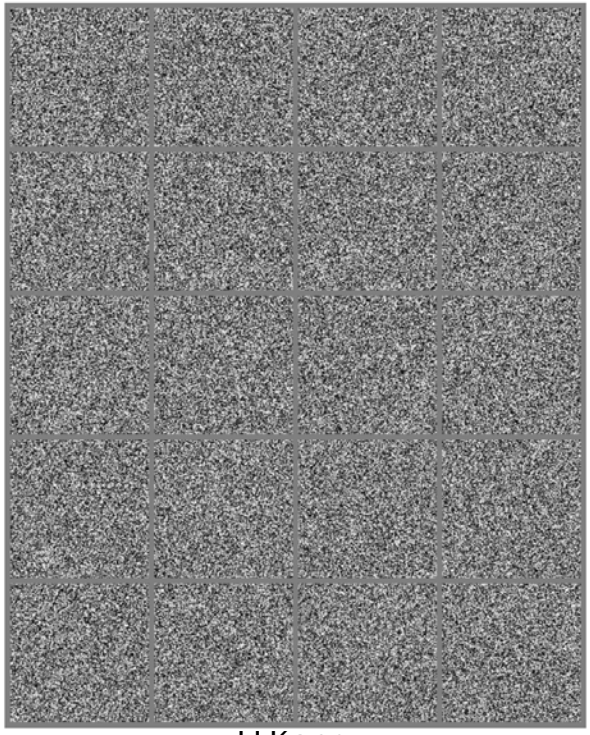
Manifold Learning

- Manifold is a connected region: a set of points associated with a neighborhood around each region
  - The surface of the earth is a 2-D manifold in 3-D space



Manifold Learning

- Real world objects are not random; they lie in low-dimensional manifolds
  - If you generate a document by picking letters uniformly at random, you would get a meaningful English text with almost 0 probability
  - Sampling pixels uniformly at random gives rise to noisy images

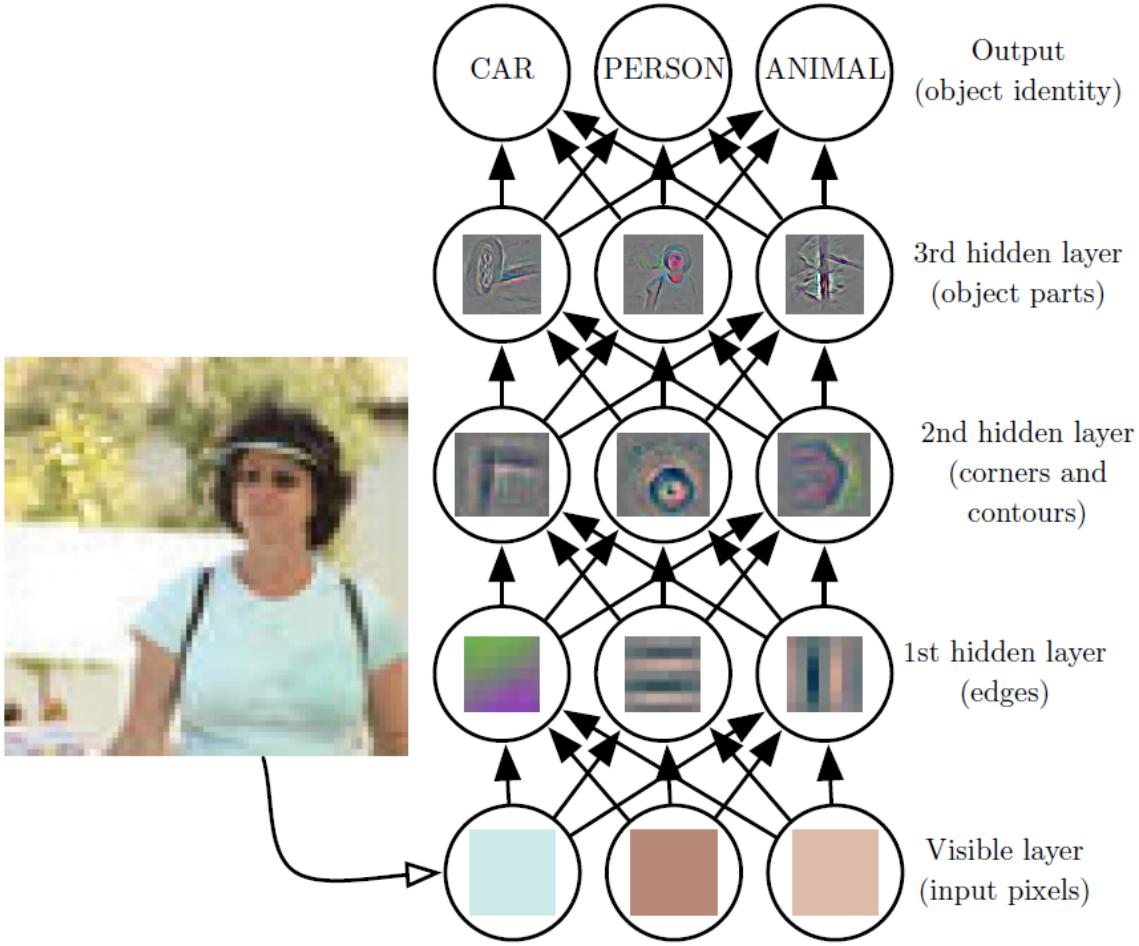


How Deep Learning Helps

- Deep learning finds an effective representation for very high dimensional objects by the following two main ideas
  - Distributed representation
    - Uses composition of factors
      - A vision system can recognize cars, trucks, and birds, and these objects can each be red, green, or blue
      - One way of representing these inputs is to have a separate neuron that activates for each of the nine possible combinations
      - Distributed representation: three neurons for objects, three neurons for colors => total six neurons
    - Allows a small number of features to represent input
  - Deep layers

How Deep Learning Helps

- Deep layers



What you need to know

- Maximum likelihood estimation or cross entropy is widely used as cost function
- SGD is a popular and efficient optimization method for many machine learning algorithms including deep learning
- Deep learning solves the problem of previous machine learning algorithms in handling high dimensional data, by finding effective representations

Questions?