Deep Learning Book §5.10 – 5.11

Apr.26 Takeshi Teshima (M1)

Overview

These sections are mostly conceptual.

5.10 Building a Machine Learning Algorithm

- A Learning machine = data + loss + model + optimizer

5.11 Challenges Motivating Deep Learning

- History of fighting for "generalization."
- The Curse of Dimensionality: a pitfall on the way to generalization.
- Principle that deep models employ to workaround "the curse."

§ 5.10 Building a Machine Learning Algorithm

5.10 Building a Machine Learning Algorithm

Summary

This section is just to help you organize in your mind many different machine learning ideas.

Recipe of learning machines

Most models / algorithms etc. can be understood through this "recipe".

A learning machine =

1. Specification of a dataset +
2. Model +
3. Cost function +
4. Optimization procedure

(We will quickly go through 4 examples.)

[ex. 1/4] Recipe of linear regression

1. Spec. of a dataset
   - (x_i, y_i) supervised learning
   - x: predictor variable (d dimensional), y: predicted variable
2. Model
   - $p_{model}(y|x) = \mathcal{N}(y; x^T w + b, 1), \quad w \in \mathbb{R}^d, b \in \mathbb{R}$
3. Cost function

$$J(w,b) = -E_{(x,y)\sim\hat{p}_{data}} \left[\log p_{model}(y|x)\right]$$

$$\left(= \frac{1}{n} \sum_{i=1}^{n} (x_i^T w + b - y_i)^2 \quad (+\text{const.}) \right)$$

4. Optimization procedure
   - Closed-form solution
   - $\hat{v} = (X^T X)^{-1} X^T y$, where
     $X = (1, x_1, \ldots, x_n)^T, \hat{v} = (\hat{b}, \hat{w}_1, \ldots, \hat{w}_d)^T.$

Likewise...

[ex. 2/4] Recipe of linear regression w/ regularization

1. Spec. of a dataset
   - (x, y)
2. Model
   - $p_{model}(y|x) = \mathcal{N}(y; x^T w + b, 1)$
3. Cost function

$$J(w, b) = -E_{(x,y)\sim\hat{p}_{data}} \left[\log p_{model}(y|x)\right] + \lambda \|w\|_2^2$$

$$\left(= \frac{1}{n} \sum_{i=1}^{n} (x_i^T w + b - y_i)^2 + \lambda \|w\|_2^2 + \text{const.} \right)$$

4. Optimization procedure
   - Closed-form solution
   - $\hat{w} = (X^T X - \lambda I)^{-1} X^T y$, where $X = (1, x_1, \ldots, x_n)^T$

Likewise...

[ex. 3/4] Recipe of nonlinear regression

1. Spec. of a dataset
    - (x, y)
2. Model
    - $p_{model}(y|x)$
    - e.g. Sigmoid: $p_{model}(y|x) = \frac{1}{1+\exp -w^T x}$
3. Cost function
    - $J(w, b) = -E_{(x,y)\sim\hat{p}_{data}} [\log p_{model}(y|x)]$
4. Optimization procedure
    - (Often) no closed form
    - e.g. Gradient descent

Likewise...

[ex. 4/4] Recipe of PCA (first principal component vector)

1. Spec. of a dataset
   - x (with $\bar{x} = 0$)
   - unsupervised
2. Model
   - $\|w\| = 1$
   - $r(x; w) = w^T x w$ (Reconstruction function)
3. Cost function

$$
\begin{aligned}
J(w) &= E_{x \sim p_{data}} \|x - r(x; w)\|^2 \\
&= \frac{1}{n} \sum_{i=1}^{n} \|x_i - w^T x_i w\|^2
\end{aligned}
$$

4. Optimization procedure
   - Corresponds to finding the first eigenvector of $\sum_{i=1}^{n} (x_i x_i^T)$.

Some comments on the recipe

- In case a cost function $J(w)$ takes time to calculate

  $\rightsquigarrow$ try approximating $\nabla J(w)$

  $\rightsquigarrow$ (approximate) minimization of $J(w)$ by iterative methods.

- In case an algorithm seems unique (doesn't fit the recipe)

  $\rightsquigarrow$ usually understood as using a special-case optimizer.

Examples of models with a hand-made optimizer

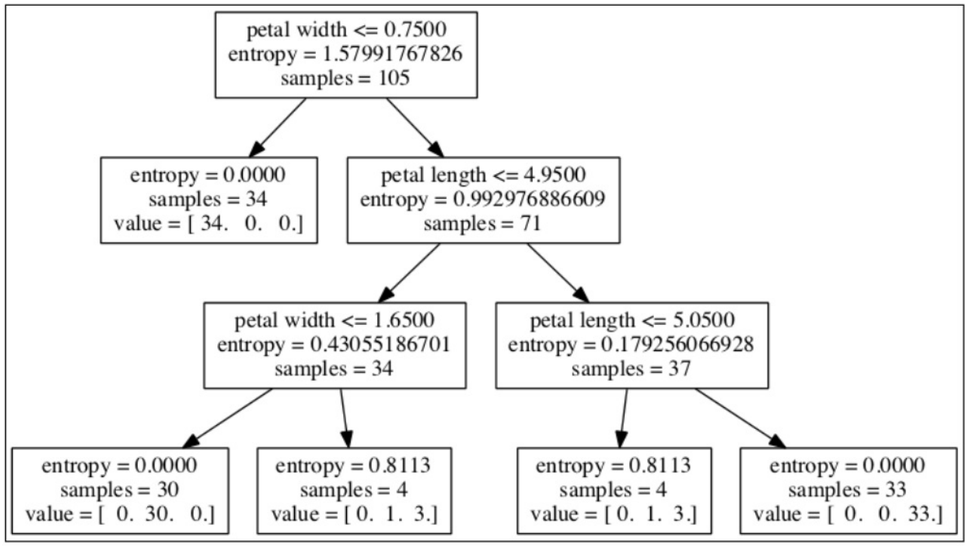
- Decision tree
  - Decision of which feature to use is required: discrete parameter

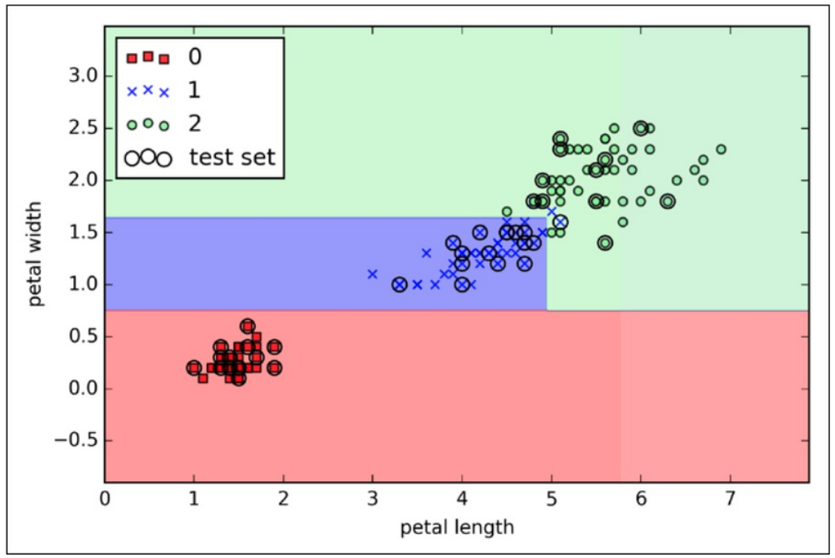
    ⇝ No gradient based methods
- k-means
  - Discrete parameter (Cluster partition)

    ⇝ No gradient based methods

Decision tree (e.g. entropy as the impurity measure)

1. Data
   - $(x_i, y_i)(i = 1, \ldots, n)$ supervised
   - $x \in \mathbb{R}^d$: features, $y \in \{1, \ldots, k\}$: label
2. Model
   - Tree = (Nodes, Edges). Each node has a rule of the form:
     $(\text{An element of } x) \leq (\text{a threshold}) \quad (\text{or} \geq)$
   - $I(X_{\text{node}}) = -\sum_{j=1}^{c} \hat{p}(\text{class} = j | x \in X_{\text{node}}) \log_2 \hat{p}(\text{class} = j | x \in X_{\text{node}})$
   - $\hat{p}(\text{class} = j | x \in X_{\text{node}}) = \frac{n_{\text{node},j}}{n_{\text{node}}}$
3. Cost Function
   - $J(\text{Tree}) = -\sum_{\text{leaf}} \sum_{j=1}^{k} n_{\text{leaf}} \times \hat{p}(j | X_{\text{leaf}}) \log_2 \hat{p}(j | X_{\text{leaf}})$
4. Optimizer
   - Maximize the information gain at each split
     $I(X_{\text{parent}}) - \sum_{\text{child}} \frac{N_{\text{child}}}{N_{\text{parent}}} I(X_{\text{child}})$.
   - Continue until all leaves are pure i.e. all the training samples in each leaf are in the same class.





Both taken from: S. Raschka, Python machine learning: unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics. Birmingham Mumbai: Packt Publishing, 2016.

k-means

1. Data
   - $x_i \in \mathbb{R}^d (i = 1, \ldots, n)$ unsupervised
2. Model
   - $k \in \mathbb{N}$ given, $S_1, \ldots, S_k \subset (\text{All samples})$: MECE
   - $Center(S_j) = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i$
3. Cost Function
   - $J(S_1, \ldots, S_k) = \sum_{j=1}^{k} \sum_{x_i \in S_j} \|x_i - Center(S_j)\|^2$
4. Optimizer [Demo]
   - Iterative optimization:
     1. Randomly choose a $V_j \in \mathbb{R}^d$ for each j and assign x_i to S_j where $j = \arg\min_{j'} \|x_i - V_{j'}\|^2$.
     2. Calculate $Center(S_j)$ for each j
     3. Assign each x_i to S_j if $j = \arg\min_{j'} \|x_i - Center(S_{j'})\|^2$.

Summary of §5.10

Recipe:

A learning machine =

1. Specification of a dataset +
2. Model +
3. Cost function +
4. Optimization procedure

When you encounter a brand new algorithm, consider this recipe to organize your mind.

§ 5.11 Challenges Motivating Deep Learning

Machine learning is a fight for "generalization."

Generalization: learning a function f that gives "a fairly good answer," y, for an unknown example x.

Conventional workaround for generalization

Conceptually, many algorithms use one of the following two assumptions to achieve generalization.

Local constancy and local smoothness

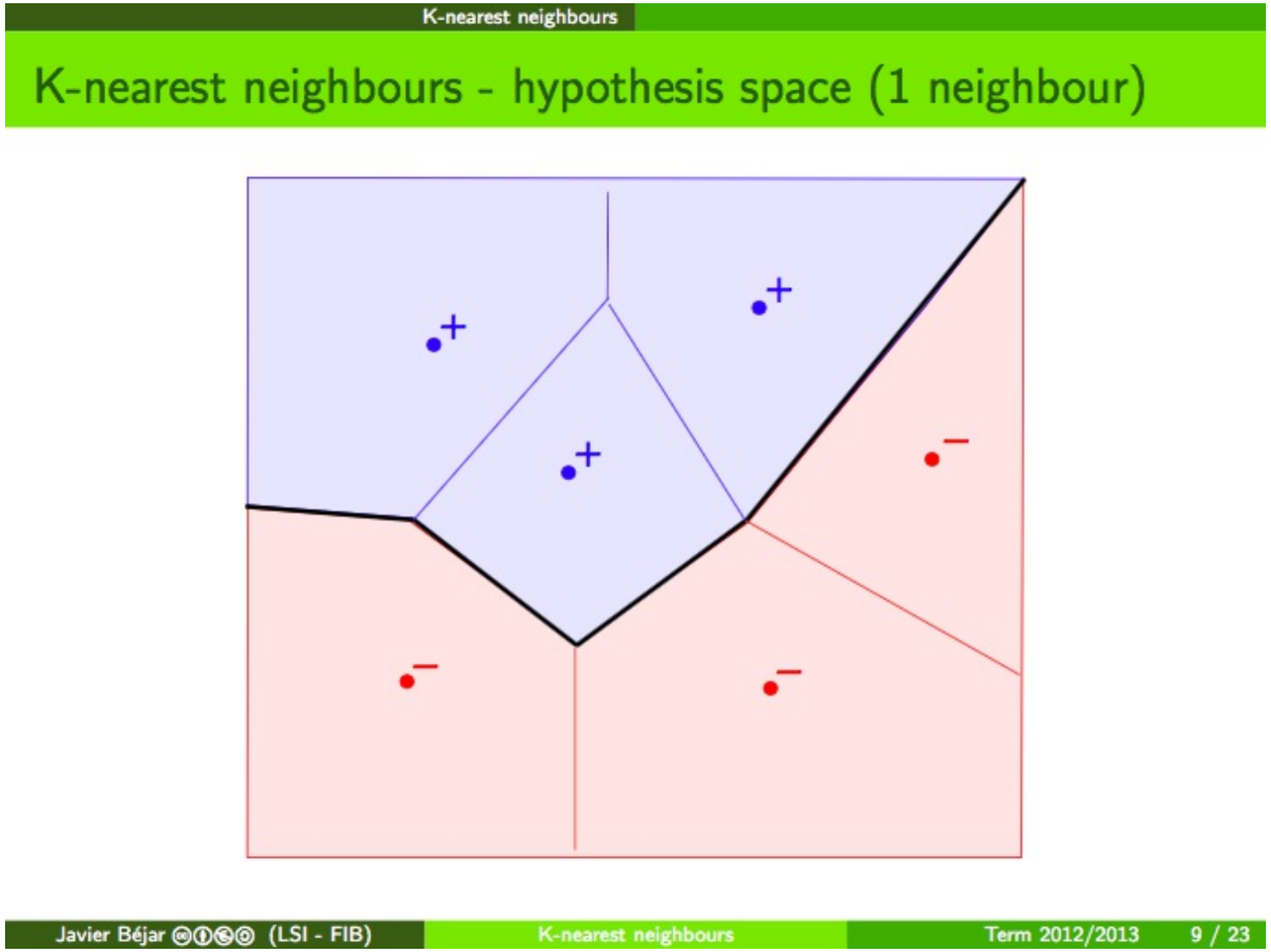
- Optimal f would satisfy: $f(x) = f(x + \epsilon)$
- Optimal f would satisfy: $f(x) \approx f(x + \epsilon)$

However, these assumptions don't scale to high dimensional settings.

We will quickly see 3 examples of these assumptions.

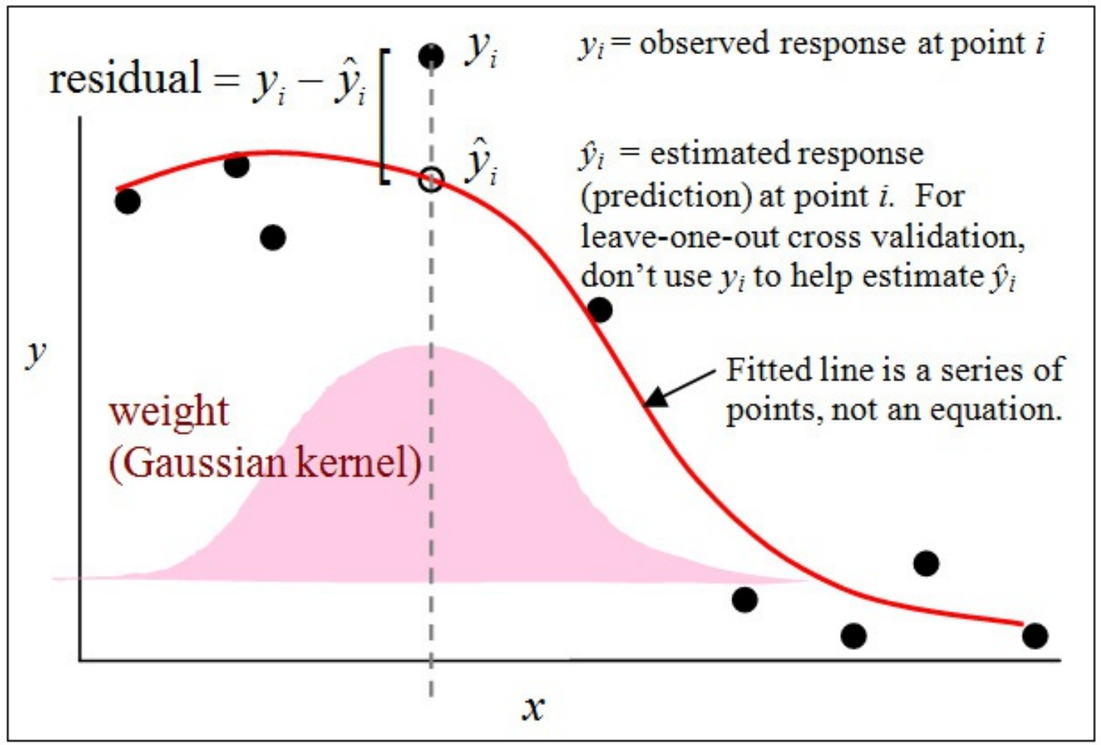
Local constancy example: k-nearest neighbors

- Learned function f predicts label according to the nearest k sample points.
- Assumes local constancy of f.



Local smoothness example: Kernel machines (local kernels)

- Employs a local kernel $k(u, v)$: large when $u = v$, decreases as u and v are farther apart
- Kernel regression: more weights on samples in the neighborhood. (Local template matching): $f(x) = \sum_{i=1}^{n} \theta_i \phi(x, X_i)$



By Brmccune - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=17877129

Caveat: local constancy and smoothness are not enough (as assumption) in high dimension.

Because there will be many regions without training samples.

This is what is called "the curse of dimensionality."

5.11.1 The Curse of Dimensionality

How local constancy and local smoothness fail:

Dimensionality high $\Rightarrow$ Sample points occupy less proportion of regions

Figure 5.9

- The number of regions $\approx O(v^d)$
  - v = (the number of possible values for each dimension)
  - d = (the dimension of x)

Locality will not work in high dimension

Insufficient samples $\Rightarrow$ locality will not work... (Low generalization)

Now, how to overcome "the curse ?"

How to overcome the curse of dimensionality?

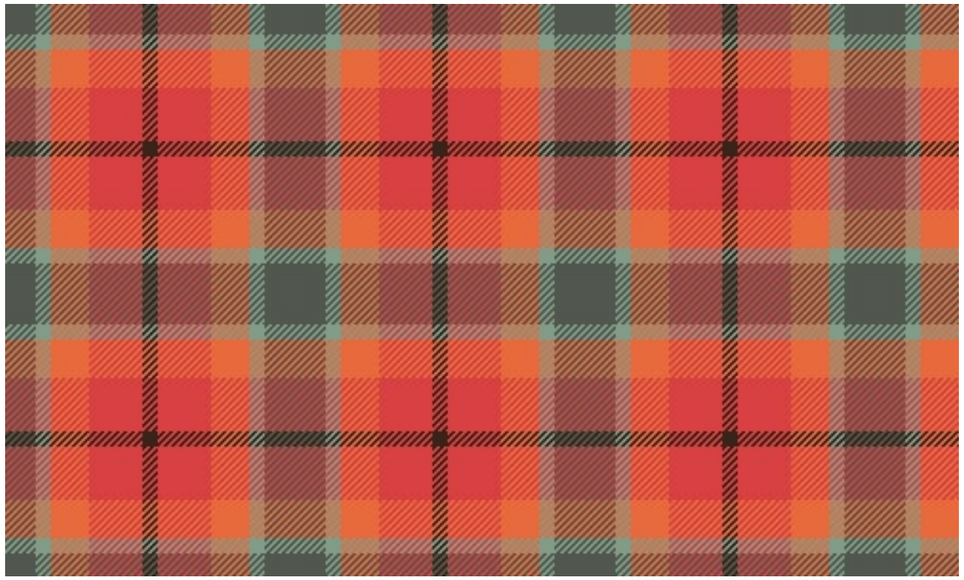
Insight:

- Conventional learning machines treat each training sample as if it only informs the learner about how to generalize in its neighborhood.

How to overcome the curse of dimensionality?

Idea for a workaround

- Find simple rules
- e.g. Checkerboard: you can find periodicity



- x : position $f(x)$: color

Samples in many (27×18) cells can be predicted with only $5 \sim 6$ rules $\rightsquigarrow$
Learn such a rule from samples

New principle for generalization, which avoids the curse

Assumption:

- The data was generated by the composition of factors or features, potentially at multiple levels in a hierarchy.

Is this assumption sufficient?

- Apparently a mild assumption, but works very well.
- In fact, the generalizability is exponential to n (the sample size). More in § 6.4.1, § 15.4, §15.5.

(For reference) How other approaches to machine learning avoid the curse of dimensionality

Make task-specific assumptions.

For example, we could easily solve the checkerboard task by providing the assumption that the target function is periodic.

New principle: Learn simple rules from training samples

Now, how do we express or formulate "simple rules?"

Or, does the world really have "simple rules?"

That's where the "manifold hypothesis" comes in.

5.11.3 Manifold Learning

How to learn / represent / formulate “simple rules”?

Hypothesis: In the real world, meaningful data don't distribute over the whole space.

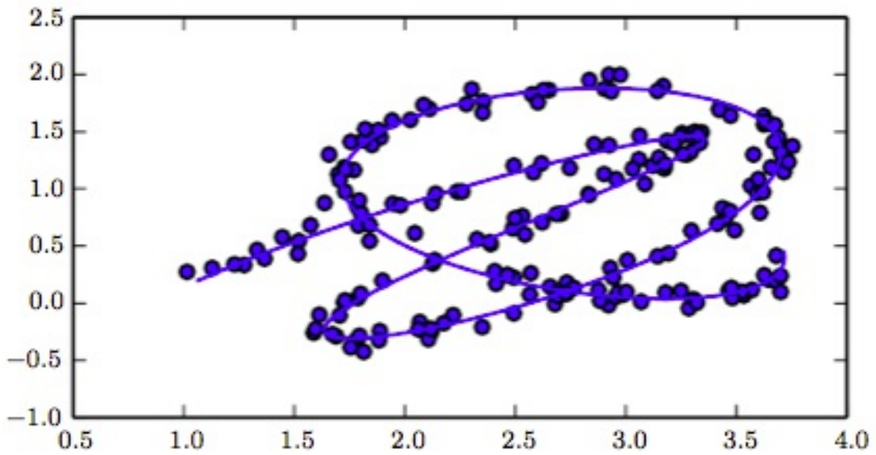


Figure 5.11

Therefore: "Find simple rules in the data" $\approx$ "Find the essential coordinate."

Manifold Hypothesis

"Data don't distribute uniformly within the whole $\mathbb{R}^n$. They lie on a lower-dimensional manifold in $\mathbb{R}^n$."

+ "interesting change occur only along the manifold / only when moving from one manifold to another."

Note: Here, the term "manifold" is casually used:

- A manifold $\approx$ "a connected set of points that has only a small dimension, embedded in a higher-dimensional space."
- Different dimensions around each point are allowed. e.g. crossing (allow the existence of singular points in the data manifold)

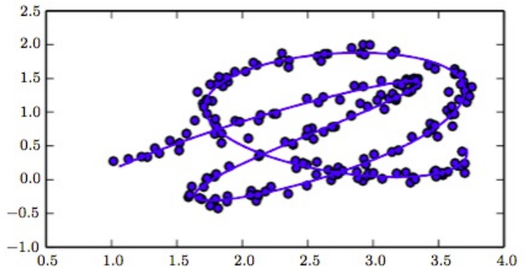


Figure 5.11

Cf. Existing methods for finding the coordinate: Manifold Learning

Manifold learning was introduced in the case of

- continuous-valued data + unsupervised

However, extendable to:

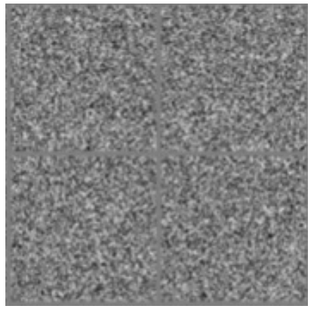
- discrete data
- supervised learning

Justification for the manifold hypothesis

Can we trust this principle?

1. Real world data seems low-dimensional.

- The probability distribution over images, text strings, and sounds that occur in real life is highly concentrated.
- Uniform noise essentially never resembles structured inputs from these domains:



- The distribution of natural language sequences occupies a very small volume in the total space of sequences of letters

Justification for the manifold hypothesis

Can we trust this principle?

2. Real world data seem to have a local coordinate.

The data are likely to form a small number of manifold (i.e. every point p has a Euclidean-like neighborhood)

- we can imagine neighborhoods and transformations.
- we can imagine gradual modifications (dim / brighten the lights, move / rotate objects, alter the colors, etc.).

Remark

It remains likely that there are multiple manifolds involved in most applications.

- For example, the manifold of images of human faces may not be connected to the manifold of images of cat faces.

Justification for the manifold hypothesis

Can we trust this principle?

3. (According to the book) More rigorous experiments clearly support the hypothesis for a large class of datasets of interest in AI.

- Cayton, 2005
- Narayanan and Mitter, 2010
- Schölkopf et al., 1998
- Roweis and Saul, 2000
- Tenenbaum et al., 2000
- Brand, 2003
- Belkin and Niyogi, 2003
- Donoho and Grimes, 2003
- Weinberger and Saul, 2004

How to employ the assumption?

Find the essential coordinate (representation).

Challenging, but this should improve many machine learning algorithms.

Ex. Variational Auto Encoder (VAE) successfully finds the coordinate (Fig. 20.6).

A VAE generated the following; the internal expression of this VAE has a "meaning" in two different dimensions (rotation & facial expression).

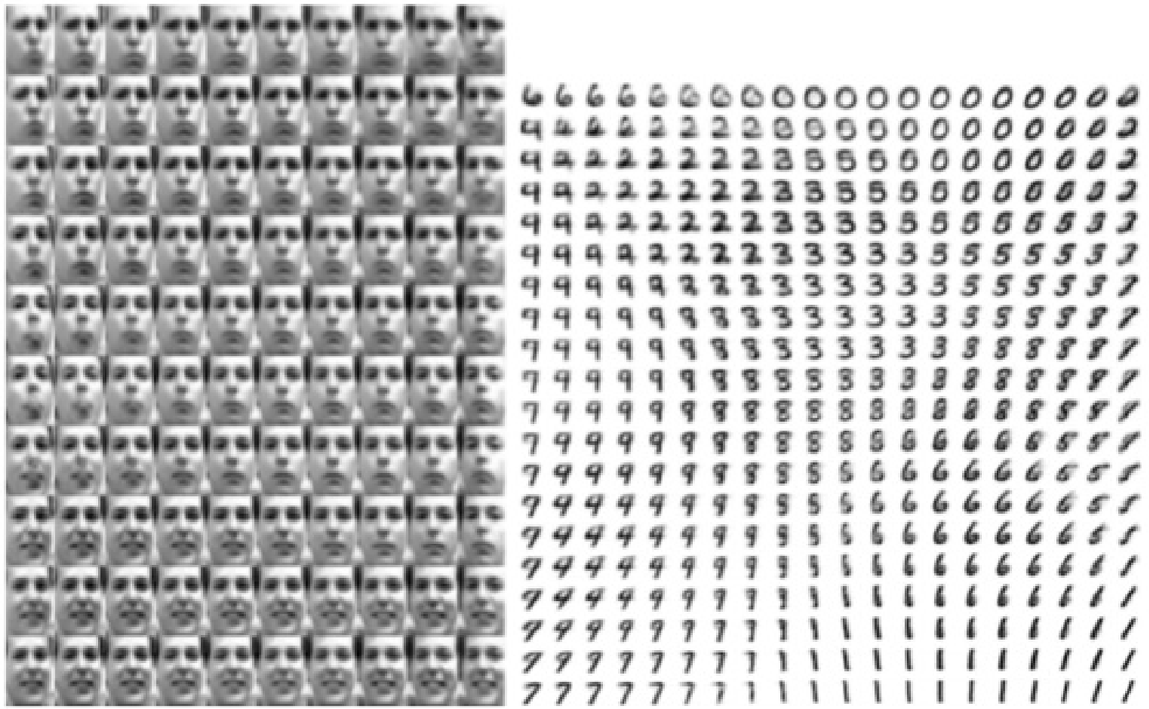


Figure 20.6

Summary of §5.11

- Locality is not a useful assumption in high dimensional setting because training samples will be sparse = the curse of dimensionality.
- Deep learning avoids the curse of dimensionality by incorporating representation learning.
    - we believe it's possible due to manifold hypothesis: the world has simple (i.e. low-dimensional) rules.