Deep Learning: Why and What

Case Study: Image Classification

Hao Zhang

haomoodzhang@gmail.com

LAMDA Group
National Key Laboratory for Novel Software Technology
Department of Computer Science and Technology
Nanjing University

May 6, 2016

Table of Contents I

Table of Contents II

Table of Contents III

Table of Contents

Hao Zhang
haomoodzhang@gmai

Image Classification

Image Classification

- Given a fixed categories, such as {cat, dog, plane, trunk, ...},

Image Classification

- Given a fixed categories, such as {cat, dog, plane, trunk, ...},
- For each image, assign one of those categories as its label.

Semantic Gap

Image recognition task

Semantic Gap

Image recognition task

- Easy for human,

Semantic Gap

Image recognition task

- Easy for human,
- Hard for computer,

Deep Learning: Why and What

Hao Zhang
haomoodzhang@gmai

Semantic Gap

Image recognition task

- Easy for human,
- Hard for computer,
- Computer can only see an array of numbers.

Other Challenges

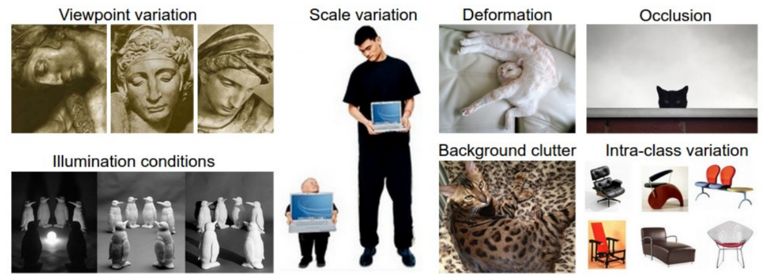


Figure: Other challenges for image recognition. This figure is reproduced from [3].

Data-Driven Approach

How to do image classification?

Data-Driven Approach

How to do image classification?

- Can not be solved by rule-based methods,

Hao Zhang
haomoodzhang@gmai

Data-Driven Approach

How to do image classification?

- Can not be solved by rule-based methods,
- So we mimic the human learning process,

Data-Driven Approach

How to do image classification?

- Can not be solved by rule-based methods,
- So we mimic the human learning process,
- Let machine learn from data.

Hao Zhang
haomoodzhang@gmai

Table of Contents

Training Data

$$D = \{(\vec{x}^{(i)}, y^{(i)})\}_{i=1}^{m} \,, \tag{1}$$

$$\vec{x}^{(i)} \in \mathbb{R}^n, \forall i \,, \tag{2}$$

$$y^{(i)} \in \{0, 1, 2, 3, \ldots, K-1\}, \forall i \,. \tag{3}$$

Hao Zhang
haomoodzhang@gmai

Training Data

$$D = \{(\vec{x}^{(i)}, y^{(i)})\}_{i=1}^{m}, \tag{1}$$

$$\vec{x}^{(i)} \in \mathbb{R}^n, \forall i, \tag{2}$$

$$y^{(i)} \in \{0, 1, 2, 3, \ldots, K-1\}, \forall i. \tag{3}$$

- m: training data size,

Training Data

$$D = \{(\vec{x}^{(i)}, y^{(i)})\}_{i=1}^{m}, \tag{1}$$

$$\vec{x}^{(i)} \in \mathbb{R}^n, \forall i, \tag{2}$$

$$y^{(i)} \in \{0, 1, 2, 3, \ldots, K-1\}, \forall i. \tag{3}$$

- m: training data size,
- K: total classes,

Training Data

$$D = \{(\vec{x}^{(i)}, y^{(i)})\}_{i=1}^{m}, \tag{1}$$

$$\vec{x}^{(i)} \in \mathbb{R}^n, \forall i, \tag{2}$$

$$y^{(i)} \in \{0, 1, 2, 3, \dots, K-1\}, \forall i. \tag{3}$$

- m: training data size,
- K: total classes,
- $\vec{x}^{(i)}$: i-th input image, expand it into vector,

Training Data

$$D = \{(\vec{x}^{(i)}, y^{(i)})\}_{i=1}^{m}\,, \tag{1}$$

$$\vec{x}^{(i)} \in \mathbb{R}^n, \forall i\,, \tag{2}$$

$$y^{(i)} \in \{0, 1, 2, 3, \ldots, K-1\}, \forall i\,. \tag{3}$$

- m: training data size,
- K: total classes,
- $\vec{x}^{(i)}$: i-th input image, expand it into vector,
- n: dimension of input data. E.g., if input image size $1024 \times 768 \times 3$, then $n = 1024 \times 768 \times 32 = 359296$,

Training Data

$$D = \{(\vec{x}^{(i)}, y^{(i)})\}_{i=1}^{m}, \tag{1}$$

$$\vec{x}^{(i)} \in \mathbb{R}^n, \forall i, \tag{2}$$

$$y^{(i)} \in \{0, 1, 2, 3, \ldots, K-1\}, \forall i. \tag{3}$$

- m: training data size,
- K: total classes,
- $\vec{x}^{(i)}$: i-th input image, expand it into vector,
- n: dimension of input data. E.g., if input image size $1024 \times 768 \times 3$, then $n = 1024 \times 768 \times 32 = 359296$,
- $y^{(i)}$: label corresponds to $\vec{x}^{(i)}$.

Hypothesis Function

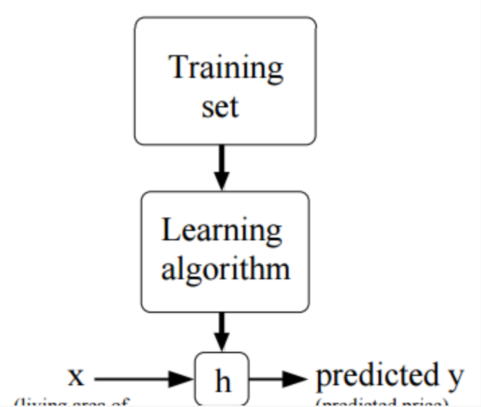


Figure: $h(\vec{x})$ gives confidence or score of $\vec{x}$ belonging to different classes. This figure is reproduced from [8].

Hypothesis of Linear Classifier

h: Linear mapping

Hypothesis of Linear Classifier

h: Linear mapping

$$h(\vec{x}; W, \vec{b}) = W\vec{x} + \vec{b}, \tag{4}$$

$$W \in \mathbb{R}^{K \times n}, \vec{b} \in \mathbb{R}^{K}. \tag{5}$$

Hypothesis of Linear Classifier

h: Linear mapping

$$h(\vec{x}; W, \vec{b}) = W\vec{x} + \vec{b}, \tag{4}$$

$$W \in \mathbb{R}^{K \times n}, \vec{b} \in \mathbb{R}^{K}. \tag{5}$$

- $(W, \vec{b})$: parameters of h,

Hypothesis of Linear Classifier

h: Linear mapping

$$h(\vec{x}; W, \vec{b}) = W\vec{x} + \vec{b}, \tag{4}$$

$$W \in \mathbb{R}^{K \times n}, \vec{b} \in \mathbb{R}^{K}. \tag{5}$$

- $(W, \vec{b})$: parameters of h,
- W: weights,

Hypothesis of Linear Classifier

Hao Zhang
haomoodzhang@gmai

h: Linear mapping

$$h(\vec{x}; W, \vec{b}) = W\vec{x} + \vec{b}, \tag{4}$$

$$W \in \mathbb{R}^{K \times n}, \vec{b} \in \mathbb{R}^{K}. \tag{5}$$

- $(W, \vec{b})$: parameters of h,
- W: weights,
- $\vec{b}$: bias vector.

Parameters Learning

Goal

Parameters Learning

Hao Zhang
haomoodzhang@gmai

Goal

- Set $(W, \vec{b})$ to let the score computed match the ground truth,

Parameters Learning

Hao Zhang
haomoodzhang@gmai

Goal

- Set $(W, \vec{b})$ to let the score computed match the ground truth,
- Once learning done, we can throw away the training set, and make prediction based on $(W, \vec{b})$.

Understanding of Hypothesis Function

$$
\begin{aligned}
h(\vec{x}; W, \vec{b}) &= W\vec{x} + \vec{b} && (6)\\
&= \begin{bmatrix} \vec{w}_0^T \\ \vec{w}_1^T \\ \vdots \\ \vec{w}_{K-1}^T \end{bmatrix} \vec{x} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{K-1} \end{bmatrix} && (7)\\
&= \begin{bmatrix} \vec{w}_0^T \vec{x} + b_0 \\ \vec{w}_1^T \vec{x} + b_1 \\ \vdots \\ \vec{w}_{K-1}^T \vec{x} + b_{K-1} \end{bmatrix}. && (8)
\end{aligned}
$$

Understanding of Hypothesis Function

$$h(\vec{x}; W, \vec{b}) = W\vec{x} + \vec{b} \tag{6}$$

$$= \begin{bmatrix} \vec{w}_0^T \\ \vec{w}_1^T \\ \vdots \\ \vec{w}_{K-1}^T \end{bmatrix} \vec{x} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{K-1} \end{bmatrix} \tag{7}$$

$$= \begin{bmatrix} \vec{w}_0^T \vec{x} + b_0 \\ \vec{w}_1^T \vec{x} + b_1 \\ \vdots \\ \vec{w}_{K-1}^T \vec{x} + b_{K-1} \end{bmatrix} . \tag{8}$$

- Simultaneously has K classifiers,

Understanding of Hypothesis Function

$$h(\vec{x}; W, \vec{b}) = W\vec{x} + \vec{b} \tag{6}$$

$$= \begin{bmatrix} \vec{w}_0^T \\ \vec{w}_1^T \\ \vdots \\ \vec{w}_{K-1}^T \end{bmatrix} \vec{x} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{K-1} \end{bmatrix} \tag{7}$$

$$= \begin{bmatrix} \vec{w}_0^T \vec{x} + b_0 \\ \vec{w}_1^T \vec{x} + b_1 \\ \vdots \\ \vec{w}_{K-1}^T \vec{x} + b_{K-1} \end{bmatrix} . \tag{8}$$

- Simultaneously has K classifiers,
- For each $0 \leq d \leq K-1$, $\vec{w}_d^T \vec{x} + b_d$ computes the confidence or score $\vec{x}$ belonging to the d-th class.

Deep Learning: Why and What

Hao Zhang
haomoodzhang@gmai

Hypothesis of Softmax Classifier

$$\vec{s} = h(\vec{x}) = W\vec{x} + \vec{b}\,. \tag{9}$$

Hypothesis of Softmax Classifier

$$\vec{s} = h(\vec{x}) = W\vec{x} + \vec{b}\,. \tag{9}$$

Probability interpretation

$$\Pr(y = k|\vec{x}) = \frac{\exp(s_k)}{\sum_{j=0}^{K-1}\exp(s_j)} \in [0, 1], \forall k\,. \tag{10}$$

Loss Function

- Given h, for each $\vec{x}^{(i)}$, we can compute scores $\vec{s}^{(i)}$ belonging to each class,

Loss Function

- Given h, for each $\vec{x}^{(i)}$, we can compute scores $\vec{s}^{(i)}$ belonging to each class,
- We need a measure $\vec{s}^{(i)}$ does or does not match ground truth $y^{(i)}$,

Loss Function

- Given h, for each $\vec{x}^{(i)}$, we can compute scores $\vec{s}^{(i)}$ belonging to each class,
- We need a measure $\vec{s}^{(i)}$ does or does not match ground truth $y^{(i)}$,
- This measure tells us goodness of parameters $(W, \vec{b})$,

Loss Function

- Given h, for each $\vec{x}^{(i)}$, we can compute scores $\vec{s}^{(i)}$ belonging to each class,
- We need a measure $\vec{s}^{(i)}$ does or does not match ground truth $y^{(i)}$,
- This measure tells us goodness of parameters $(W, \vec{b})$,
- Loss function measures how much $\vec{s}^{(i)}$ does not match $y^{(i)}$.

Cross Entropy Loss

Loss function of softmax classifier can be computed by maximum likelihood estimate

Deep Learning: Why and What

Hao Zhang
haomoodzhang@gmai

Cross Entropy Loss

Loss function of softmax classifier can be computed by maximum likelihood estimate

$$
\begin{aligned}
W^\star, \vec{b}^\star &= \arg\max_{W,\vec{b}} \prod_{i=1}^{m} p(y^{(i)} | \vec{x}^{(i)}; W, \vec{b}) && (11) \\
&= \arg\min_{W,\vec{b}} \frac{1}{m} \sum_{i=1}^{m} \left(-s_{y^{(i)}}^{(i)} + \log\left(\sum_{j=0}^{K-1} \exp(s_j^{(i)})\right) \right) && (12) \\
&\stackrel{\text{def}}{=} \arg\min_{W,\vec{b}} \frac{1}{m} \sum_{i=1}^{m} \mathrm{err}(W, \vec{b}; \vec{x}^{(i)}, y^{(i)}) && (13) \\
&\stackrel{\text{def}}{=} \arg\min_{W,\vec{b}} J(W, \vec{b}) \,. && (14)
\end{aligned}
$$

Cross Entropy Loss

Loss function of softmax classifier can be computed by maximum likelihood estimate

$$W^\star, \vec{b}^\star = \arg\max_{W,\vec{b}} \prod_{i=1}^{m} p(y^{(i)} | \vec{x}^{(i)}; W, \vec{b}) \tag{11}$$

$$= \arg\min_{W,\vec{b}} \frac{1}{m} \sum_{i=1}^{m} \left(-s_{y^{(i)}}^{(i)} + \log(\sum_{j=0}^{K-1} \exp(s_j^{(i)})) \right) \tag{12}$$

$$\stackrel{\text{def}}{=} \arg\min_{W,\vec{b}} \frac{1}{m} \sum_{i=1}^{m} \mathrm{err}(W, \vec{b}; \vec{x}^{(i)}, y^{(i)}) \tag{13}$$

$$\stackrel{\text{def}}{=} \arg\min_{W,\vec{b}} J(W, \vec{b})\,. \tag{14}$$

For simplicity, write $\mathrm{err}(W, \vec{b}; \vec{x}^{(i)}, y^{(i)})$ as $\mathrm{err}^{(i)}$.

Regularization

Good $(W, \vec{b})$ min.

$$J(W, \vec{b}) = \frac{1}{m} \sum_{i=1}^{m} - \log \frac{\exp(s_{y^{(i)}}^{(i)})}{\sum_{j=0}^{K-1} \exp(s_j^{(i)})} . \tag{15}$$

Regularization

Good $(W, \vec{b})$ min.

$$J(W, \vec{b}) = \frac{1}{m} \sum_{i=1}^{m} -\log \frac{\exp(s^{(i)}_{y^{(i)}})}{\sum_{j=0}^{K-1} \exp(s^{(i)}_{j})}. \tag{15}$$

But that $(W, \vec{b})$ is not unique

Regularization

Good $(W, \vec{b})$ min.

$$J(W, \vec{b}) = \frac{1}{m} \sum_{i=1}^{m} -\log \frac{\exp(s_{y^{(i)}}^{(i)})}{\sum_{j=0}^{K-1} \exp(s_j^{(i)})} . \tag{15}$$

But that $(W, \vec{b})$ is not unique

- Suppose $(W^\star, \vec{b}^\star)$ is the opt. sol.,

Regularization

Good $(W, \vec{b})$ min.

$$J(W, \vec{b}) = \frac{1}{m} \sum_{i=1}^{m} -\log \frac{\exp(s_{y^{(i)}}^{(i)})}{\sum_{j=0}^{K-1} \exp(s_j^{(i)})} . \tag{15}$$

But that $(W, \vec{b})$ is not unique

- Suppose $(W^\star, \vec{b}^\star)$ is the opt. sol.,
- Then the result of adding const. $c, c > 0$ to every element of $(W^\star, \vec{b}^\star)$ is also opt.,

Regularization

Good $(W, \vec{b})$ min.

$$J(W, \vec{b}) = \frac{1}{m} \sum_{i=1}^{m} - \log \frac{\exp(s_{y^{(i)}}^{(i)})}{\sum_{j=0}^{K-1} \exp(s_j^{(i)})} . \tag{15}$$

But that $(W, \vec{b})$ is not unique

- Suppose $(W^\star, \vec{b}^\star)$ is the opt. sol.,
- Then the result of adding const. $c, c > 0$ to every element of $(W^\star, \vec{b}^\star)$ is also opt.,
- We handle this problem by make preference of $(W, \vec{b})$, this is achieved by adding regularization term $\Omega(W)$,

Regularization (cont'd)

- Commonly used regularization term is ℓ_2 norm

$$\Omega(W) = \|W\|_F^2 = \sum_{d=0}^{K-1} \sum_{j=0}^{n-1} W_{dj}^2 \,. \tag{16}$$

Regularization (cont'd)

- Commonly used regularization term is ℓ_2 norm

$$\Omega(W) = \|W\|_F^2 = \sum_{d=0}^{K-1} \sum_{j=0}^{n-1} W_{dj}^2 \,. \tag{16}$$

- ℓ_2 prefers small and diffused weights, which is good for generalization.

Regularization (cont'd)

- Commonly used regularization term is ℓ_2 norm

$$\Omega(W) = \|W\|_F^2 = \sum_{d=0}^{K-1} \sum_{j=0}^{n-1} W_{dj}^2 \,. \tag{16}$$

- ℓ_2 prefers small and diffused weights, which is good for generalization.

Regularization (cont'd)

- Commonly used regularization term is ℓ_2 norm

$$\Omega(W) = \|W\|_F^2 = \sum_{d=0}^{K-1} \sum_{j=0}^{n-1} W_{dj}^2 . \qquad (16)$$

- ℓ_2 prefers small and diffused weights, which is good for generalization.

Then the total loss function becomes

$$J(W, \vec{b}) = \frac{1}{m} \sum_{i=1}^{m} -\log \frac{\exp(s_{y^{(i)}}^{(i)})}{\sum_{j=0}^{K-1} \exp(s_j^{(i)})} + \frac{\lambda}{2} \|W\|_F^2 \qquad (17)$$

$$= \frac{1}{m} \sum_{i=1}^{m} \mathrm{err}^{(i)} + \frac{\lambda}{2} \|W\|_F^2 . \qquad (18)$$

Optimization

Goal:

$$W^\star, \vec{b}^\star = \arg\min_{W, \vec{b}} J(W, \vec{b})\,. \tag{19}$$

Hao Zhang
haomoodzhang@gmai

Optimization

Goal:

$$W^\star, \vec{b}^\star = \arg\min_{W, \vec{b}} J(W, \vec{b}) \,. \tag{19}$$

This is an unconstrained optimization problem.

Visualing Loss Function

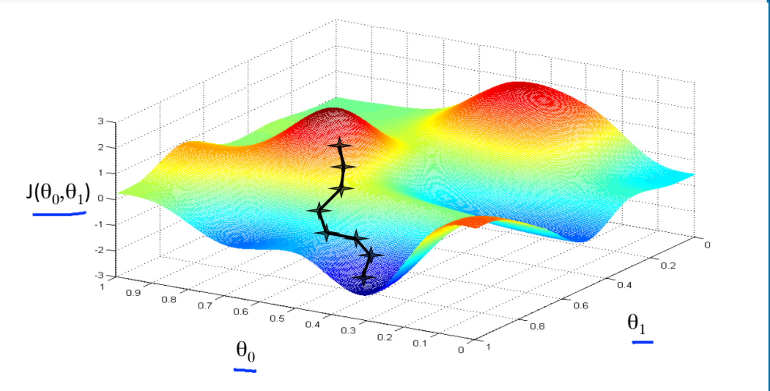


Figure: Visualizing J wrt $(W, \vec{b})$ is hard, but we can choose 2 directions (θ_0, θ_1) in parameter space, and draw J along that 2 directions. This figure is reproduced from [9].

Gradient Descent

How to find opt. $(W, \vec{b})$

Gradient Descent

How to find opt. $(W, \vec{b})$

- Find them directly is hard,

Gradient Descent

How to find opt. $(W, \vec{b})$

- Find them directly is hard,
- Our strategy is start with random $(W, \vec{b})$, and refine it iteratively

Hao Zhang
haomoodzhang@gmai

Gradient Descent

How to find opt. $(W, \vec{b})$

- Find them directly is hard,
- Our strategy is start with random $(W, \vec{b})$, and refine it iteratively

Gradient Descent

How to find opt. $(W, \vec{b})$

- Find them directly is hard,
- Our strategy is start with random $(W, \vec{b})$, and refine it iteratively

$$W \leftarrow W - \alpha \nabla_W J(W, \vec{b}) = W - \alpha \frac{1}{m} \sum_{i=1}^{m} \nabla_W \mathrm{err}^{(i)} - \alpha \lambda W \,, \tag{20}$$

$$\vec{b} \leftarrow \vec{b} - \alpha \nabla_{\vec{b}} J(W, \vec{b}) = \vec{b} - \alpha \frac{1}{m} \sum_{i=1}^{m} \nabla_{\vec{b}} \mathrm{err}^{(i)} \,. \tag{21}$$

Softmax Update Rule

$$\vec{s}^{(i)} = W\vec{x}^{(i)} + \vec{b}, \forall i \,, \tag{22}$$

$$\mathrm{err}^{(i)} = -\log \frac{\exp(s^{(i)}_{y^{(i)}})}{\sum_{j=0}^{K-1} \exp(s^{(i)}_j)} \,. \tag{23}$$

Softmax Update Rule

$$\vec{s}^{(i)} = W\vec{x}^{(i)} + \vec{b}, \forall i\,, \tag{22}$$

$$\mathrm{err}^{(i)} = -\log \frac{\exp(s^{(i)}_{y^{(i)}})}{\sum_{j=0}^{K-1} \exp(s^{(i)}_j)}\,. \tag{23}$$

By caculus,

$$\nabla_{\vec{s}^{(i)}} \mathrm{err}^{(i)} = \frac{\exp(\vec{s}^{(i)})}{\sum_{j=0}^{K-1} \exp(s^{(i)}_j)} - \vec{e}_{y^{(i)}}\,. \tag{24}$$

Softmax Update Rule

$$\vec{s}^{(i)} = W\vec{x}^{(i)} + \vec{b}, \forall i\,, \tag{22}$$

$$\mathrm{err}^{(i)} = -\log \frac{\exp(s_{y^{(i)}}^{(i)})}{\sum_{j=0}^{K-1} \exp(s_j^{(i)})}\,. \tag{23}$$

By caculus,

$$\nabla_{\vec{s}^{(i)}} \mathrm{err}^{(i)} = \frac{\exp(\vec{s}^{(i)})}{\sum_{j=0}^{K-1} \exp(s_j^{(i)})} - \vec{e}_{y^{(i)}}\,. \tag{24}$$

- $\vec{e}_{y^{(i)}}$ is a K-dimension vector,

Softmax Update Rule

$$\vec{s}^{(i)} = W\vec{x}^{(i)} + \vec{b}, \forall i \,, \tag{22}$$

$$\mathrm{err}^{(i)} = -\log \frac{\exp(s^{(i)}_{y^{(i)}})}{\sum_{j=0}^{K-1} \exp(s^{(i)}_j)} \,. \tag{23}$$

By caculus,

$$\nabla_{\vec{s}^{(i)}} \mathrm{err}^{(i)} = \frac{\exp(\vec{s}^{(i)})}{\sum_{j=0}^{K-1} \exp(s^{(i)}_j)} - \vec{e}_{y^{(i)}} \,. \tag{24}$$

- $\vec{e}_{y^{(i)}}$ is a K-dimension vector,
- it has 1 in $y^{(i)}$'s position, and 0 otherwise.

Softmax Update Rule (cont'd)

By chain rule

$$\nabla_W \mathrm{err}^{(i)} = (\nabla_{\vec{s}^{(i)}} \mathrm{err}^{(i)}) \vec{x}^{(i)T}, \tag{25}$$

$$\nabla_{\vec{b}} \mathrm{err}^{(i)} = \nabla_{\vec{s}^{(i)}} \mathrm{err}^{(i)}. \tag{26}$$

Prediction and Evaluation

Prediction and Evaluation

- Let the classifier model to predict on an unseen test set,

Prediction and Evaluation

- Let the classifier model to predict on an unseen test set,
- A good model will have many predictions match the true label.

Hao Zhang
haomoodzhang@gmai

Prediction

For unseen $\vec{x}$, the classifier make a prediction by computing

$$\hat{y} = \arg\max_k h(\vec{x})_k = \arg\max_k s_k \,. \tag{27}$$

Hao Zhang
haomoodzhang@gmai

Prediction

For unseen $\vec{x}$, the classifier make a prediction by computing

$$\hat{y} = \arg\max_k h(\vec{x})_k = \arg\max_k s_k \,. \tag{27}$$

I.e., assign $\vec{x}$ with label whose score is the highest.

Evaluation

Given test set

$$D_{test} = \{(\vec{x}^{(i)}, y^{(i)})\}_{i=1}^{m_{test}}, \tag{28}$$

Evaluation

Given test set

$$D_{test} = \{(\vec{x}^{(i)}, y^{(i)})\}_{i=1}^{m_{test}}, \tag{28}$$

For each $\vec{x}^{(i)}$, compute prediction $\hat{y}^{(i)}$.

Evaluation

Hao Zhang
haomoodzhang@gmai

Given test set

$$D_{test} = \{(\vec{x}^{(i)}, y^{(i)})\}_{i=1}^{m_{test}}, \tag{28}$$

For each $\vec{x}^{(i)}$, compute prediction $\hat{y}^{(i)}$.
Accuracy of the model is

$$Acc = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} 1\{\hat{y}^{(i)} = y^{(i)}\}. \tag{29}$$

Table of Contents

Linear Models

Linear Models

- Softmax classifier represents a series of linear models,

Linear Models

- Softmax classifier represents a series of linear models,
- Training is convex optimization,

Hao Zhang
haomoodzhang@gmai

Linear Models

- Softmax classifier represents a series of linear models,
- Training is convex optimization,
- Guaranteed to converge to global optimal,

Linear Models

- Softmax classifier represents a series of linear models,
- Training is convex optimization,
- Guaranteed to converge to global optimal,
- But it can only use multiple hyperplanes to seperate input space into some simple regions,

Linear Models

- Softmax classifier represents a series of linear models,
- Training is convex optimization,
- Guaranteed to converge to global optimal,
- But it can only use multiple hyperplanes to seperate input space into some simple regions,
- For many problems, the raw pixels are not linearly separable.

Feature Engineering

Figure: Find proper representation $\phi(\vec{x})$ to take place $\vec{x}$, such that $\phi(\vec{x})$'s are linearly separable. This figure is reproduced from [6].

SIFT Feature

Figure: SIFT can handle occlusion. This figure is reproduced from [7].

Hao Zhang
haomoodzhang@gmai

HOG feature

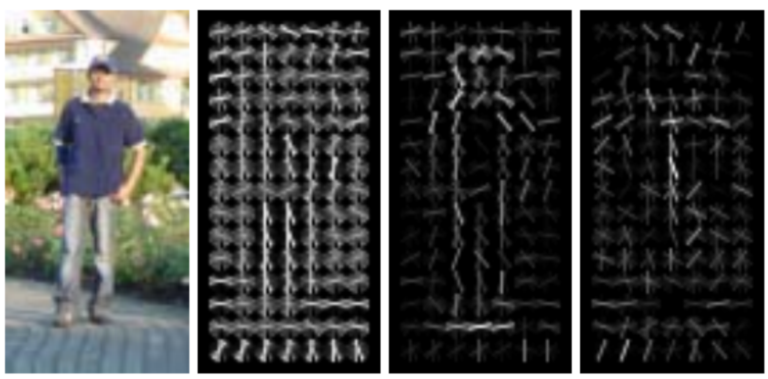


Figure: HOG feature. This figure is reproduced from [1].

Kernel Methods

Hao Zhang
haomoodzhang@gmai

Kernel Methods

- Use fixed ϕ to map $\vec{x}$ to a higher dimension space,

Hao Zhang
haomoodzhang@gmai

Kernel Methods

- Use fixed ϕ to map $\vec{x}$ to a higher dimension space,
- The optimization is still convex,

Kernel Methods

- Use fixed ϕ to map $\vec{x}$ to a higher dimension space,
- The optimization is still convex,
- The choice of ϕ is kernel engineering,

Hao Zhang
haomoodzhang@gmai

Kernel Methods

- Use fixed ϕ to map $\vec{x}$ to a higher dimension space,
- The optimization is still convex,
- The choice of ϕ is kernel engineering,
- Commonly used kernel is Gauss kernel: $k(\vec{x}, \vec{x}') = \exp(-\gamma \|\vec{x} - \vec{x}'\|^2)$.

Representation Learning

Representation Learning

Machine learning can learn the mapping from representation to output.

Representation Learning

Machine learning can learn the mapping from representation to output.
Can we also let machine learn appropriate representation?

Representation Learning

Machine learning can learn the mapping from representation to output.
Can we also let machine learn appropriate representation?
Autoencoder is one example of representation learning.

Deep Learning

Hao Zhang
haomoodzhang@gmai

Deep Learning

Learn appropriate mapping directly from raw pixel is still very hard.

Deep Learning

Learn appropriate mapping directly from raw pixel is still very hard.

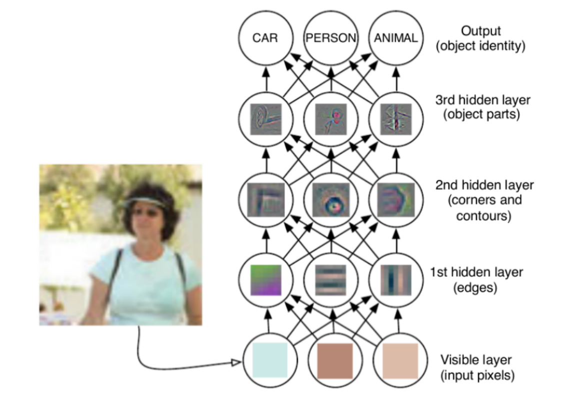
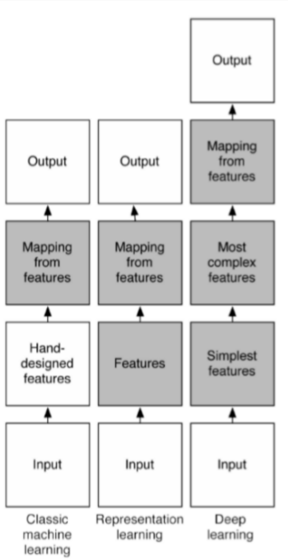


Figure: Deep learning divides it into a series of easy learning problems. $\phi(\vec{x}) = \phi_3(\phi_2(\phi_1(\vec{x})))$. This figure is reproduced from [4].

Comparation



Data Representation

Linear hypothesis function

$$h(\vec{x}) = W\vec{x} + \vec{b}. \tag{30}$$

Data Representation

Linear hypothesis function

$$h(\vec{x}) = W\vec{x} + \vec{b}. \tag{30}$$

Changes to

$$h(\vec{x}) = W\phi(\vec{x}) + \vec{b}. \tag{31}$$

Data Representation

Linear hypothesis function

$$h(\vec{x}) = W\vec{x} + \vec{b}\,. \tag{30}$$

Changes to

$$h(\vec{x}) = W\phi(\vec{x}) + \vec{b}\,. \tag{31}$$

Nested representation learning

$$\phi(\vec{x}) = \phi_{L-1}(\phi_{L-2}(\ldots\phi_2(\phi_1(\vec{x}))))\,. \tag{32}$$

Composition of each $\phi_l(\vec{a})$

Linear part

$$\vec{z} = W^{(l)}\vec{a} + \vec{b}^{(l)}, \forall l = 1, 2, \ldots, L-1\,. \tag{33}$$

Composition of each $\phi_l(\vec{a})$

Linear part

$$\vec{z} = W^{(l)}\vec{a} + \vec{b}^{(l)}, \forall l = 1, 2, \ldots, L-1\,. \tag{33}$$

Non linear part (activation function)

$$\phi_l(\vec{a}) = \max(0, \vec{z}), \forall l = 1, 2, \ldots, L-1\,. \tag{34}$$

Activation Function

Rectified linear units, ReLU

$$\max(0, s) = 1\{s > 0\}s\,. \tag{35}$$

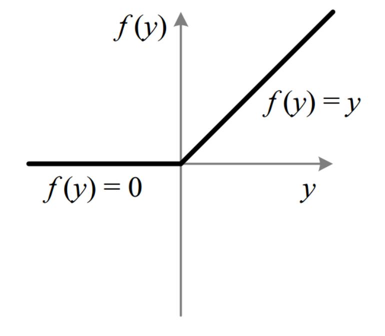


Figure: ReLU. This figure is reproduced from [5].

Activation Function (cont'd)

Sigmoid

$$\sigma(s) = \frac{1}{1 + \exp(-s)} \,. \tag{36}$$

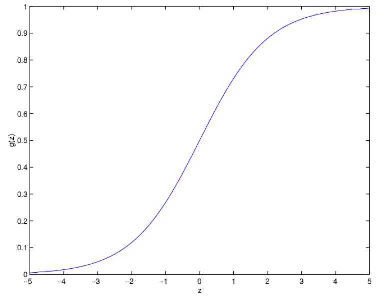


Figure: Sigmoid. This figure is reproduced from [8].

In All

Hao Zhang
haomoodzhang@gmai

If define

$$\vec{a}^{(0)} = \vec{x} \,. \tag{37}$$

In All

If define

$$\vec{a}^{(0)} = \vec{x}\,. \tag{37}$$

$$\vec{z}^{(l)} = W^{(l)}\vec{a}^{(l-1)} + \vec{b}^{(l)}, \forall l = 1, 2, \ldots, L-1\,, \tag{38}$$

$$\vec{a}^{(l)} = \max(0, \vec{z}^{(l)}), \forall l = 1, 2, \ldots, L-1\,, \tag{39}$$

$$h(\vec{x}) = W^{(L)}\vec{a}^{(L-1)} + \vec{b}^{(L)}\,. \tag{40}$$

In All

If define

$$\vec{a}^{(0)} = \vec{x} \,. \tag{37}$$

$$\vec{z}^{(l)} = W^{(l)} \vec{a}^{(l-1)} + \vec{b}^{(l)}, \forall l = 1, 2, \dots, L-1 \,, \tag{38}$$

$$\vec{a}^{(l)} = \max(0, \vec{z}^{(l)}), \forall l = 1, 2, \dots, L-1 \,, \tag{39}$$

$$h(\vec{x}) = W^{(L)} \vec{a}^{(L-1)} + \vec{b}^{(L)} \,. \tag{40}$$

- $\vec{a}^{(L-1)}$ is the representation $\phi(\vec{x})$.

Artificial Neural Networks

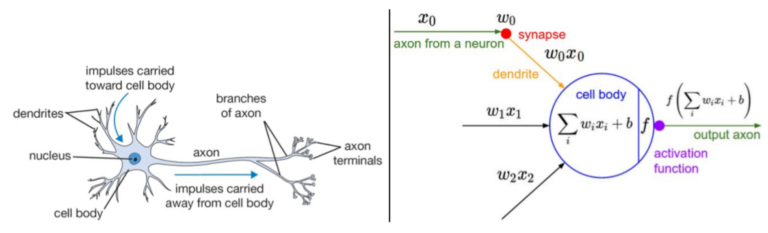


Figure: Biology model and math model. This figure is reproduced from [3].

Architecture

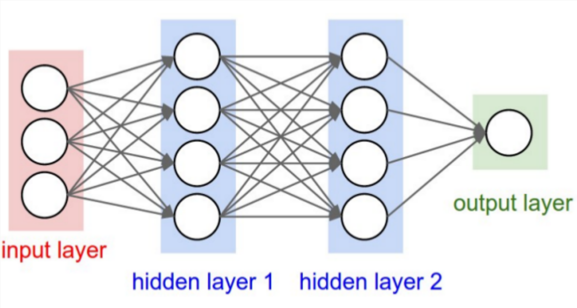


Figure: Multilayer feedforward nueral network. This figure is reproduced from [3].

Loss Function

$$J(W, \vec{b}) = \frac{1}{m} \sum_{i=1}^{m} -\log \frac{\exp(s_{y^{(i)}}^{(i)})}{\sum_{j=0}^{K-1} \exp(s_j^{(i)})} + \sum_{l=1}^{L} \frac{\lambda}{2} \| W^{(l)} \|_F^2 \tag{41}$$

$$= \frac{1}{m} \sum_{i=1}^{m} \mathrm{err}^{(i)} + \sum_{l=1}^{L} \frac{\lambda}{2} \| W^{(l)} \|_F^2 \,. \tag{42}$$

$$\vec{s}^{(i)} = h(\vec{x}^{(i)}) \,. \tag{43}$$

Gradient Descent

Update rule

$$W^{(l)} \leftarrow W^{(l)} - \alpha \nabla_{W^{(l)}} J = W^{(l)} - \alpha \frac{1}{m} \sum_{i=1}^{m} \nabla_{W^{(l)}} \mathrm{err}^{(i)} - \alpha \lambda W^{(l)}, \forall l, \quad (44)$$

$$\vec{b}^{(l)} \leftarrow \vec{b}^{(l)} - \alpha \nabla_{\vec{b}^{(l)}} J = \vec{b}^{(l)} - \alpha \frac{1}{m} \sum_{i=1}^{m} \nabla_{\vec{b}^{(l)}} \mathrm{err}^{(i)}, \forall l. \quad (45)$$

Gradient Descent (cont'd)

$$\nabla_{\vec{s}^{(i)}} \mathrm{err}^{(i)} = \frac{\exp(\vec{s}^{(i)})}{\sum_{j=0}^{K-1} \exp(s_j^{(i)})} - \vec{e}_{y^{(i)}} , \tag{46}$$

$$\vec{s}^{(i)} = h(\vec{x}^{(i)}) = W^{(L)} \vec{a}^{(L-1)} + \vec{b}^{(L)} . \tag{47}$$

Gradient Descent (cont'd)

$$\nabla_{\vec{s}^{(i)}} \mathrm{err}^{(i)} = \frac{\exp(\vec{s}^{(i)})}{\sum_{j=0}^{K-1} \exp(s_j^{(i)})} - \vec{e}_{y^{(i)}} \,, \tag{46}$$

$$\vec{s}^{(i)} = h(\vec{x}^{(i)}) = W^{(L)} \vec{a}^{(L-1)} + \vec{b}^{(L)} \,. \tag{47}$$

Easy one

$$\nabla_{W^{(L)}} \mathrm{err}^{(i)} = (\nabla_{\vec{s}^{(i)}} \mathrm{err}^{(i)}) \vec{x}^{(i)T} \,, \tag{48}$$

$$\nabla_{\vec{b}^{(L)}} \mathrm{err}^{(i)} = \nabla_{\vec{s}^{(i)}} \mathrm{err}^{(i)} \,. \tag{49}$$

Gradient Descent (cont'd)

$$\nabla_{\vec{s}^{(i)}} \mathrm{err}^{(i)} = \frac{\exp(\vec{s}^{(i)})}{\sum_{j=0}^{K-1} \exp(s_j^{(i)})} - \vec{e}_{y^{(i)}} \,, \tag{46}$$

$$\vec{s}^{(i)} = h(\vec{x}^{(i)}) = W^{(L)} \vec{a}^{(L-1)} + \vec{b}^{(L)} \,. \tag{47}$$

Easy one

$$\nabla_{W^{(L)}} \mathrm{err}^{(i)} = (\nabla_{\vec{s}^{(i)}} \mathrm{err}^{(i)}) \vec{x}^{(i)T} \,, \tag{48}$$

$$\nabla_{\vec{b}^{(L)}} \mathrm{err}^{(i)} = \nabla_{\vec{s}^{(i)}} \mathrm{err}^{(i)} \,. \tag{49}$$

Hard One: $\nabla_{W^{(l)}} \mathrm{err}^{(i)}$ and $\nabla_{\vec{b}^{(l)}} \mathrm{err}^{(i)}$, $l < L$?

Error Back-Propagation

At layer l

$$\vec{z}^{(l)} = W^{(l)}\vec{a}^{(l-1)} + \vec{b}^{(l)}, \tag{50}$$

$$\vec{a}^{(l)} = \max(0, \vec{z}^{(l)}). \tag{51}$$

Error Back-Propagation

At layer l

$$\vec{z}^{(l)} = W^{(l)}\vec{a}^{(l-1)} + \vec{b}^{(l)}\,, \tag{50}$$

$$\vec{a}^{(l)} = \max(0, \vec{z}^{(l)})\,. \tag{51}$$

Assume $\nabla_{\vec{a}^{(l)}}\mathrm{err}^{(i)}$ is known, then

$$\nabla_{\vec{z}^{(l)}}\mathrm{err}^{(i)} = \left(\frac{\partial \vec{a}^{(l)}}{\partial \vec{z}^{(l)}}\right)^T \nabla_{\vec{a}^{(l)}}\mathrm{err}^{(i)} \tag{52}$$

$$= 1\{\vec{z}^{(l)} > 0\} \odot \nabla_{\vec{a}^{(l)}}\mathrm{err}^{(i)}\,. \tag{53}$$

Deep Learning: Why and What

Hao Zhang
haomoodzhang@gmai

Error Back-Propagation

At layer l

$$\vec{z}^{(l)} = W^{(l)}\vec{a}^{(l-1)} + \vec{b}^{(l)}\,, \tag{50}$$

$$\vec{a}^{(l)} = \max(0, \vec{z}^{(l)})\,. \tag{51}$$

Assume $\nabla_{\vec{a}^{(l)}}\mathrm{err}^{(i)}$ is known, then

$$\begin{aligned}\nabla_{\vec{z}^{(l)}}\mathrm{err}^{(i)} &= \left(\frac{\partial \vec{a}^{(l)}}{\partial \vec{z}^{(l)}}\right)^T \nabla_{\vec{a}^{(l)}}\mathrm{err}^{(i)} &(52)\\ &= 1\{\vec{z}^{(l)} > 0\} \odot \nabla_{\vec{a}^{(l)}}\mathrm{err}^{(i)}\,. &(53)\end{aligned}$$

$$\begin{aligned}\nabla_{\vec{a}^{(l-1)}}\mathrm{err}^{(i)} &= \left(\frac{\partial \vec{z}^{(l)}}{\partial \vec{a}^{(l-1)}}\right)^T \nabla_{\vec{z}^{(l)}}\mathrm{err}^{(i)} &(54)\\ &= W^T \nabla_{\vec{z}^{(l)}}\mathrm{err}^{(i)}\,, &(55)\\ \nabla_{W^{(l)}}\mathrm{err}^{(i)} &= (\nabla_{\vec{z}^{(l)}}\mathrm{err}^{(i)})\vec{a}^{(l-1)T}\,, &(56)\\ \nabla_{\vec{b}^{(l)}}\mathrm{err}^{(i)} &= \nabla_{\vec{z}^{(l)}}\mathrm{err}^{(i)}\,. &(57)\end{aligned}$$

Prediction and Evaluation

Prediction and Evaluation

- The same as linear models,

Prediction and Evaluation

- The same as linear models,
- But $\vec{s}^{(i)} = h(\vec{x}^{(i)})$ has different form.

Hao Zhang
haomoodzhang@gmai

Table of Contents

Training Data

Require image input

$$D = \{(\mathbf{X}^{(i)}, y^{(i)})\}_{i=1}^{m} \,, \tag{58}$$

$$\mathbf{X}^{(i)} \in \mathbb{R}^{H \times W \times D}, \forall i \,, \tag{59}$$

$$y^{(i)} \in \{0, 1, 2, 3, \ldots, K-1\}, \forall i \,. \tag{60}$$

Hypothesis Function

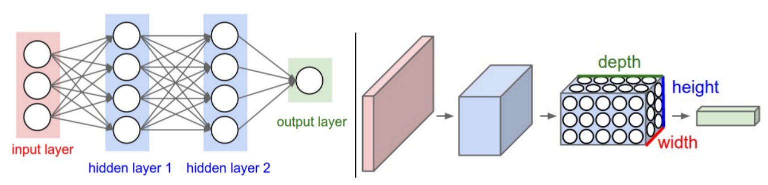


Figure: Normal neural networks and CNN. This figure is reproduced from [3].

Expand Fully Connected Affine to Tensor

$$\vec{a} = W\vec{x} + \vec{b} \tag{61}$$

$$= \begin{bmatrix} \vec{w}_0^T \\ \vec{w}_1^T \\ \vdots \\ \vec{w}_{n_l-1}^T \end{bmatrix} \vec{x} + \vec{b} \tag{62}$$

$$= \begin{bmatrix} \vec{w}_0^T \vec{x} + b_0 \\ \vec{w}_1^T \vec{x} + b_1 \\ \vdots \\ \vec{w}_{n_l-1}^T \vec{x} + b_{n_l-1} \end{bmatrix}. \tag{63}$$

Expand Fully Connected Affine to Tensor

$$\vec{a} = W\vec{x} + \vec{b} \tag{61}$$

$$= \begin{bmatrix} \vec{w}_0^T \\ \vec{w}_1^T \\ \vdots \\ \vec{w}_{n_l-1}^T \end{bmatrix} \vec{x} + \vec{b} \tag{62}$$

$$= \begin{bmatrix} \vec{w}_0^T\vec{x} + b_0 \\ \vec{w}_1^T\vec{x} + b_1 \\ \vdots \\ \vec{w}_{n_l-1}^T\vec{x} + b_{n_l-1} \end{bmatrix} . \tag{63}$$

$$a_{d_l} = \vec{w}_{d_l}^T\vec{x} + b_{d_l} = \vec{w}_{d_l} \odot \vec{x} + b_{d_l}, \forall d_l . \tag{64}$$

Expand Fully Connected Affine to Tensor (cont'd)

Hao Zhang
haomoodzhang@gmai

When $\mathbf{X} \in \mathbb{R}^{H_{l-1} \times W_{l-1} \times D_{l-1}}$,

Expand Fully Connected Affine to Tensor (cont'd)

When $\mathbf{X} \in \mathbb{R}^{H_{l-1} \times W_{l-1} \times D_{l-1}}$,
$\mathbf{W}_{d_l} \in \mathbb{R}^{H_{l-1} \times W_{l-1} \times D_{l-1}}$,

$$a_{d_l} = \mathbf{W}_{d_l} \odot \mathbf{X} + b_{d_l} = \sum_{i=0}^{H_{l-1}-1} \sum_{j=0}^{W_{l-1}-1} \sum_{d=0}^{D_{l-1}-1} \mathbf{X}(i,j,d) \mathbf{W}_{d_l}(i,j,d) + b_{d_l}, \forall d_l . \tag{65}$$

Two Simplifications of Convolutional Layer

- Sparse Connection
- Parameter Sharing

Sparse Connection

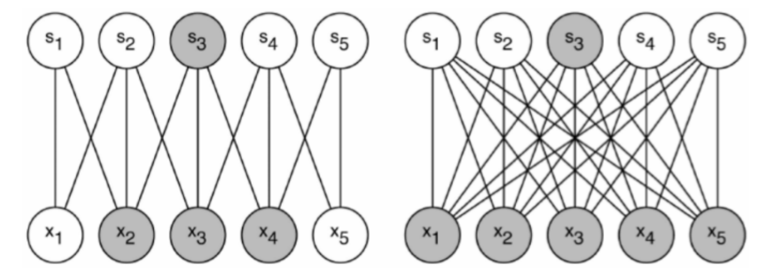


Figure: Sparse connection in 1 dimension. This figure is reproduced from [4].

Sparse Connection (cont'd)

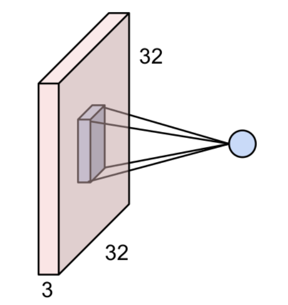


Figure: Sparse connection in 3 dimensions. This figure is reproduced from [3].

Sparse Connection (cont'd)

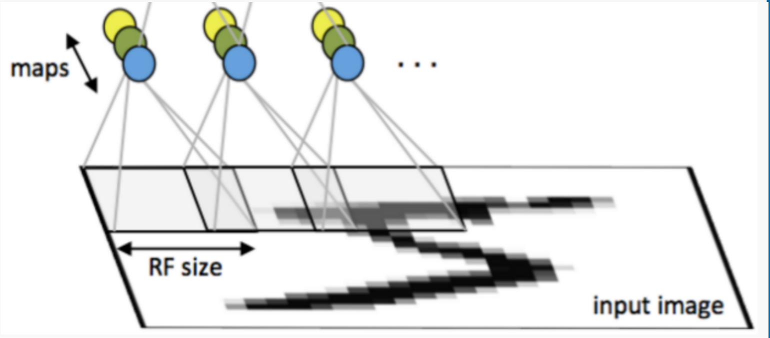


Figure: Activation map. This figure is reproduced from [2].

Sparse Connection (cont'd)

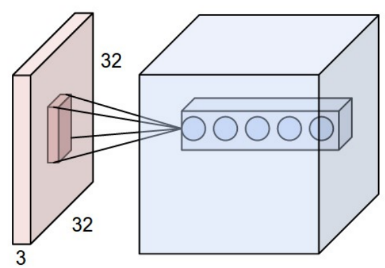


Figure: Input and output tensor. This figure is reproduced from [3].

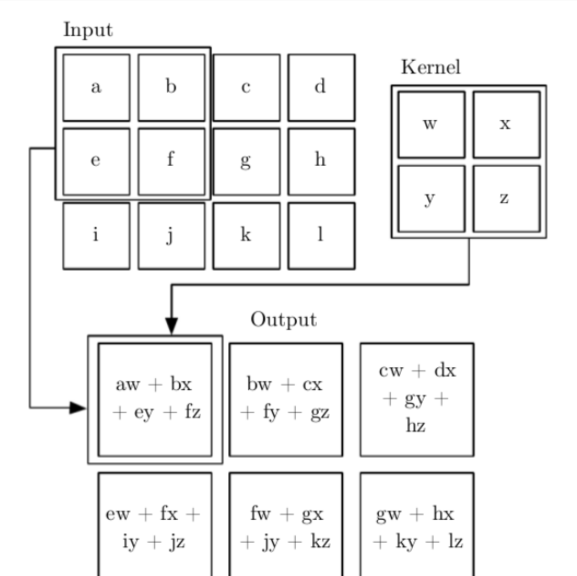
Parameter Sharing

If a motif can appear in one part of the image, it could appear anywhere,

Parameter Sharing

If a motif can appear in one part of the image, it could appear anywhere,
Hence units at different locations sharing the same weights

Convolution



Convolution (cont'd)

$$\mathbf{A}_{d_l} = \mathbf{X} * \mathbf{W}_{d_l} + b_{d_l}, \forall d_l \,. \tag{66}$$

$$\mathbf{A}(i_l, j_l, d_l) = \sum_{i=0}^{F_1-1} \sum_{j=0}^{F_2-1} \sum_{d=0}^{D_{l-1}-1} \mathbf{X}(i_l+i, j_l+j, d)\mathbf{W}(i, j, d, d_l)+b_{d_l}, \forall i_l, j_l, d_l \,. \tag{67}$$

Pooling Layer

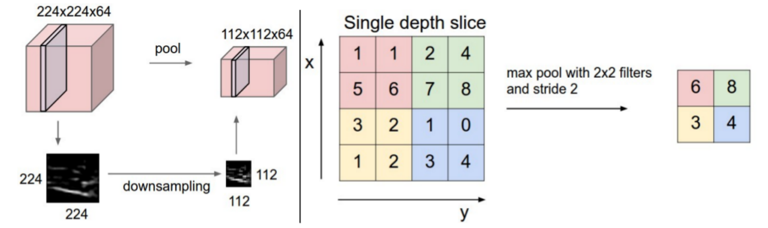


Figure: Max pooling. This figure is reproduced from [3].

Pooling Layer (cont'd)

$$\mathbf{A}(i_l, j_l, d_l) = \max_{0 \le i < F_1, 0 \le j < F_2} \mathbf{X}(i_l S + i, j_l S + j, d_l), \forall i_l, j_l, d_l \,. \tag{68}$$

ReLU Layer

$$\mathbf{A}(i_l, j_l, d_l) = \max(0, \mathbf{X}(i_l, j_l, d_l)), \forall i_l, j_l, d_l\,. \tag{69}$$

Fully Connected Layer

The same as normal feedforward neural networks.

Table of Contents

References I

Navneet Dalal and Bill Triggs.
Histograms of oriented gradients for human detection.
In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

Andrew Ng et al.
UFLDL tutorial.
Stanford.

Li Fei-Fei, Andrej Karpathy, and Justin Johnson.
CS231n: Convolutional neural networks for visual recognition.
Stanford, 2016.

References II

Ian Goodfellow, Yoshua Bengio, and Aaron Courville.
Deep learning.
Book in preparation for MIT Press, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.
Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.
In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton.
Deep learning.
Nature, 521(7553):436–444, 2015.

References III

David G Lowe.
Object recognition from local scale-invariant features.
In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

Andrew Ng.
CS229: Machine learning.
Stanford.

Andrew Ng.
Machine learning.
Coursera.

Thanks

Questions Please!

Hao Zhang
haomoodzhang@gmail.com