W07-12 Recap + Final Exam Practice Review

13B

CS 3244 Machine Learning

NUS National University of Singapore | Computing

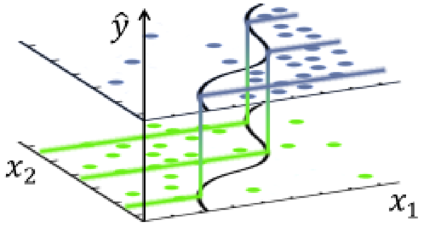
W07-12 Recap

NUS National University of Singapore
Department of Computer Science
School of Computing

Classification

$y \in \{0,1\}$ binary

$y \in \{y_A, y_B, \dots\}$ multi-class



$y = M(\boldsymbol{x}), \quad \boldsymbol{x} = \vec{x} = (x_1, x_2)^\top$

[W07b] Student Learning Outcomes

What did we learn for Evaluation?

1. Classification vs. Regression
2. Classification Metrics
   1. Accuracy
   2. Confusion Matrix, TP, TN, FP, FN
   3. Precision, Recall, F_1
   4. ROC, AUC
   5. Micro- and Macro-Averaging
   6. PR-AUC (Average Precision)
3. Regression Metrics
   1. 1D regression: MSE, MAE
   2. Vector regression: Euclidean distance, Angular distance / Cosine Similarity

Exercise E07.1

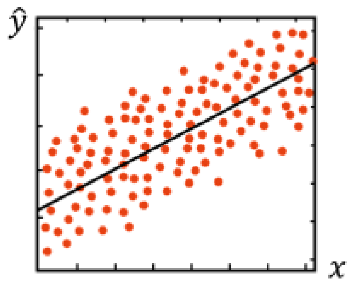
Match appropriate **evaluation metric** to **challenge**

Challenge	**Evaluation Metric**
Imbalanced actual classes	Accuracy (Emote :one:) Precision (:two:) Recall (:three:)
Multiclass classification	F_1 Score (:four:) ROC AUC (:five:) PRC AUC (:six:)
Cost-dependent classes	Micro-Average (:seven:) Macro-Average (:eight:)

Emote (react) in Slack #general channel one or more options (MRQ) for each challenge

Regression

$y \in \mathbb{R}$ any real number



$y = M(\boldsymbol{x}), \quad \boldsymbol{x} = x_1$

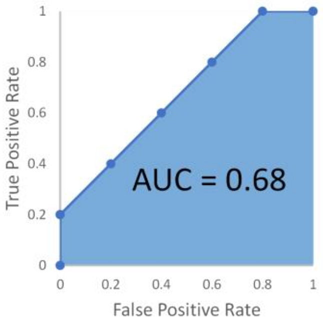
Classification Metrics

Predicted Label		Actual Label: Alert	Actual Label: Not	
Alert		2 True Positive	1 False Positive	3 ∑ Pred. Pos.
Not		3 False Negative	4 True Negative	7 ∑ Pred. Neg.
		5 ∑ Actual Pos.	5 ∑ Actual Neg.	

Precision
P = TP / (TP+FP)

Recall
R = TP / (TP+FN)

F_1 Score
$$F_1 = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$



AUC is a **concise metric** instead of a full figure.
Concise metrics enable *clearer comparisons*.
AUC > 0.5 means the model is better than chance.
AUC ≈ 1 means model is very accurate.

Regression Metrics

Average difference metrics

Mean Absolute Error (MAE)

$$MAE = \frac{1}{m}\sum_{j=1}^{m} |\hat{y}_j - y_j|$$

Mean Squared Error (MSE)

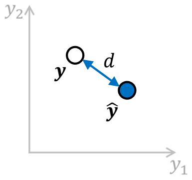
$$MSE = \frac{1}{m}\sum_{j=1}^{m} (\hat{y}_j - y_j)^2$$

Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{1}{m}\sum_{j=1}^{m} (\hat{y}_j - y_j)^2}$$

MSE and RMSE penalize larger differences more than MAE

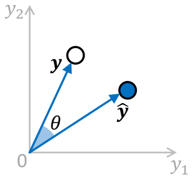
Vector Distances and Similarity



Euclidean Distance

$$d = \sqrt{(\hat{\boldsymbol{y}} - \boldsymbol{y})^\top (\hat{\boldsymbol{y}} - \boldsymbol{y})}$$

Dot Product



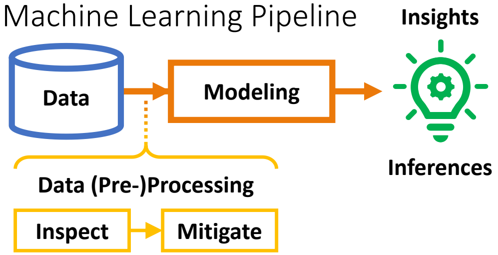
Cosine Similarity

$$s = \cos(\theta) = \frac{\hat{\boldsymbol{y}} \cdot \boldsymbol{y}}{\|\hat{\boldsymbol{y}}\|\|\boldsymbol{y}\|}$$

Angular Distance

$$\theta = \cos^{-1}(s)$$

Recap W08a – Data Preprocessing



Student Learning Outcomes

What did we learn this week?

Data Issues
1. Linear Separability
2. Curse of Dimensionality
3. Imbalanced Data

Issue Template
1. **What** is the issue?
2. **Why** is it a problem?
3. **When** would it happen?
4. **How to check** for it?
5. **How to mitigate** it?

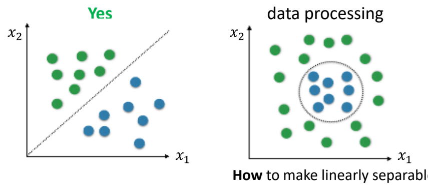
Checks
1. Linear SVM, PCA, LDA
2. Visualize Histograms

Mitigations
1. Dimensionality Reduction (PCA, LDA, Deep Auto-Encoders)
2. Feature Selection (Recursive Feature Elimination, Correlation, Mutual Information)
3. Resampling (Under/Oversampling, SMOTE)

NUS CS3244: Machine Learning 52

Issue: Linear Separability

Linearly Separable?



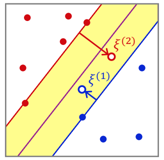
How to make linearly separable?

$x = \binom{x_1}{x_2}$ $\quad x' = \binom{(x_1 - \bar{x}_1)^2}{(x_2 - \bar{x}_2)^2} = (x - \bar{x})^\top (x - \bar{x})$

Feature Engineering!

Testing Linear Separability with Linear Soft-Margin SVM

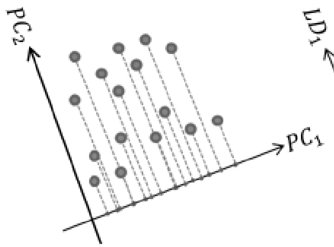
- Each $\xi^{(j)}$ is the **distance** that the misclassified point j is from its correct margin
- Total violation: $\sum_{j=1}^{m} \xi^{(j)}$
- Calculating the **total violation** indicates how **linearly separable** the data is in terms of its features
- Higher violation => Less linearly separable



Issue: Curse of Dimensionality

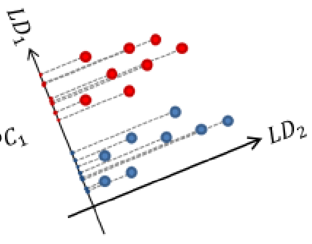
Dimensionality Reduction

PCA
Maximize Data Variance



Good for **dimensionality reduction** for supervised regression and unsupervised learning

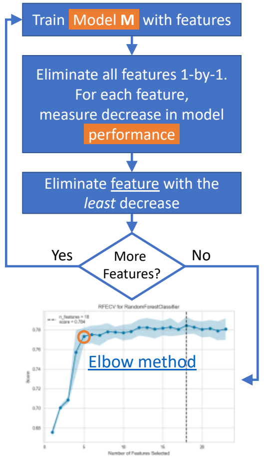
LDA
Maximize Class Separation



Good for **dimensionality reduction** for supervised classification

Feature Selection

Recursive Feature Elimination



Information gain

A chosen feature x_i divides the example set S into subsets $S_1, S_2, \ldots, S_c$ according to the C_i distinct values for x_i.

The entropy then reduces to the entropy of the subsets $S_1, S_2, \ldots, S_c$:

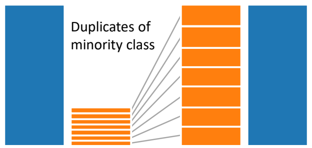
$$\text{remainder}(S, x_i) = \sum_{j=1}^{C_i} \frac{|S_j|}{|S|} H(S_j)$$

Information Gain (IG; "reduction in entropy") from knowing the value of x_i. Choose the attribute with the largest IG:

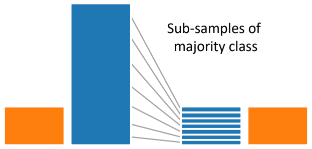
$$\text{IG}(S, x_i) = H(S) - \text{remainder}(S, x_i)$$

Issue: Imbalanced Data

Oversampling

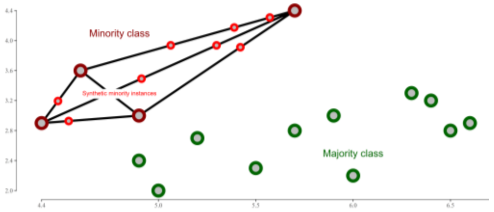


Undersampling

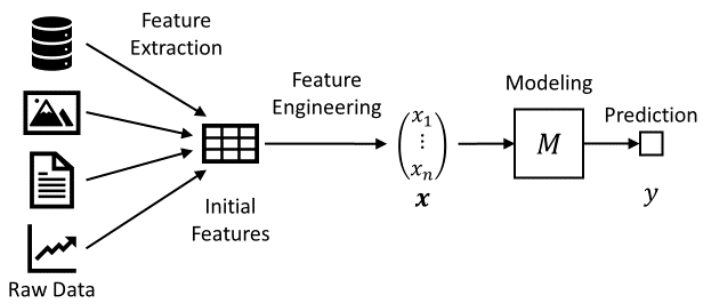


SMOTE

Steps
1. Consider minority and majority instances in vector space.
2. For each minority-class instance **pair**, **interpolate** their feature values.
3. Randomly synthesize instances and label with minority class
4. More instances added to minority class

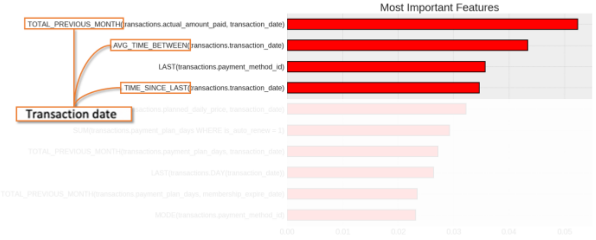


Feature Extraction/Engineering → Modeling



Tabular

Tabular Feature Engineering:
Counting, Aggregation, Difference, Min, Max



Source: https://github.com/Featuretools/predict-customer-churn

Temporal

Sliding Time Window

- Prediction Task: **Price Prediction**
- Features
  - Moving Average
  - Moving Standard Deviation
  - Moving Range (Min, Max)
  - Moving Trend (Slope of linear fit)

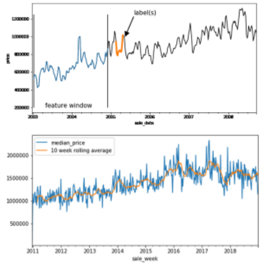


Image credit: https://cloud.google.com/blog/products/ai-machine-learning/how-to-quickly-solve-machine-learning-forecasting-problems-using-pandas-and-bigquery

[W08b] Student Learning Outcomes

What did we learn?

1. Describe **issues** when extracting features for various data types
2. Describe **techniques** of feature extraction/engineering for different data types

Tabular	Temporal	Image	Text
• Domain-specific custom equations • Features from counting, aggregation, difference, min, max	• Features from previous values, aggregate statistics, linear regression • Wave analysis features	• RGB image as 3D tensor • Color features from RGB histogram • Shape features from edge detection • Edge detection via Convolution	• Tokenization • Stemming, Lemmatization • Stop words • Bag-of-Words encoding

Image

Feature: Edge Detection Kernels

$$\frac{\partial c}{\partial p_x} \approx \frac{c_{(x+1,y)} - c_{(x-1,y)}}{p_{(x+1,y)} - p_{(x-1,y)}}$$

$$I_{p_x} = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad \boldsymbol{x} = \begin{pmatrix} c_{(-1,-1)} & c_{(0,-1)} & c_{(1,-1)} \\ c_{(-1,0)} & c_{(0,0)} & c_{(1,0)} \\ c_{(-1,1)} & c_{(0,1)} & c_{(1,1)} \end{pmatrix}$$

$*$ is **Convolution** operator
Means: element-wise multiply, then sum

$$I_{p_x} * \boldsymbol{x} = \left(-c_{(-1,-1)} + 0 + c_{(1,-1)}\right) + \left(-c_{(-1,0)} + 0 + c_{(1,0)}\right) + \left(-c_{(-1,1)} + 0 + c_{(1,1)}\right)$$

I_{p_x} is a Convolution Matrix (**Kernel**)

$$\boldsymbol{x} = \begin{pmatrix} 9 & 9 & 3 & 3 & 4 \\ 9 & 3 & 3 & 4 & 5 \\ 9 & 3 & 3 & 5 & 5 \\ 9 & 3 & 3 & 4 & 5 \\ 9 & 9 & 3 & 3 & 4 \end{pmatrix} \quad I_{p_x} * \boldsymbol{x} = \begin{pmatrix} -6-6-6 & -6+1+2 & 1+2+2 \\ -6-6-6 & 1+2+1 & 2+2+2 \\ -6-6-6 & 2+1-6 & 2+2+1 \end{pmatrix}$$

Text

Bag-of-Words (BOW) Encoding

1. Preprocess string s to array of words $\boldsymbol{w}$
2. Array of words → One-hot vector (fixed length)
3. $\mathrm{BOW}(\boldsymbol{w}) \rightarrow \boldsymbol{x}$
4. Problem: high dimensions if many words

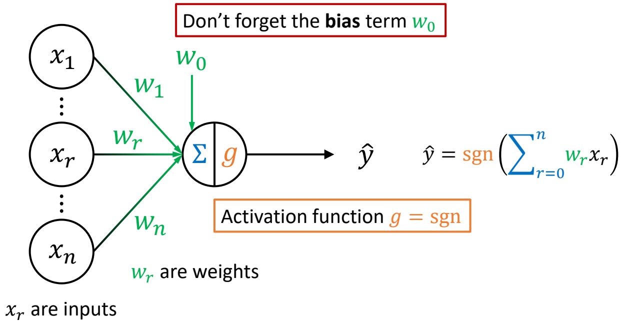
$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_{13} \end{pmatrix} \quad x^{(1)} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad x^{(2)} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

#	Original Text s	Pre-Processed Words $\boldsymbol{w}$	chicken	wings	amazing	honestly	but	way	too	long	wait	not	worth	salty	expensive
1	"Chicken wings were amazing honestly"	['chicken', 'wings', 'amazing', 'honestly']	1	1	1	1	0	0	0	0	0	0	0	0	0
2	"Amazing wings, but waaaay to long to wait."	['amazing', 'wings', 'but', 'way', 'too', 'long', 'wait']	0	1	1	0	1	1	1	1	1	0	0	0	0
3	"Not worth it! Too salty chicken and expensive!"	['not', 'worth', 'too', 'salty', 'chicken', 'expensive']	1	0	0	0	0	0	1	0	0	1	1	1	1

The **word "too"** could predict negative sentiment

Perceptron

Perceptron



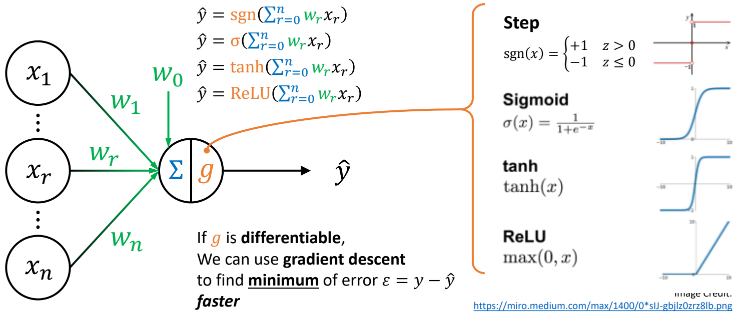
Perceptron Learning Algorithm

1. Initialize weights $\boldsymbol{w}$
   - Could be all zero, or random small values
2. For each instance i with features $\boldsymbol{x}^{(i)}$
   - Classify $\hat{y}^{(i)} = \text{sgn}\left(\boldsymbol{w}^\top \boldsymbol{x}^{(i)}\right)$
3. Select one misclassified instance
   - Update weights: $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta(y - \hat{y})\boldsymbol{x}$
4. Iterate steps 2 to 3 until
   - Convergence (classification error < threshold), or
   - Maximum number of iterations

$$\begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_r \\ \vdots \\ w_n \end{pmatrix} \leftarrow \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_r \\ \vdots \\ w_n \end{pmatrix} + \eta(y - \hat{y}) \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_r \\ \vdots \\ x_n \end{pmatrix}$$

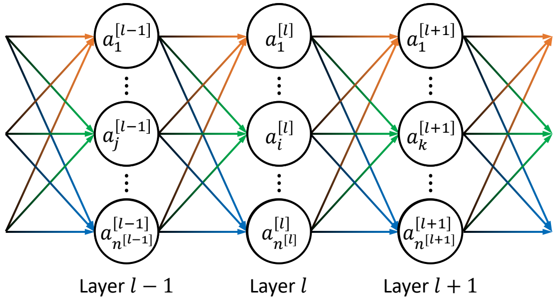
$$w_r \leftarrow w_r + \eta(y - \hat{y})x_r$$

Differentiable Activation Functions



Neural Networks

Multi-Layer Perceptron (Neural Network)



Chain Rule

Consider composite function

$$g(x) = g(f(x))$$

$$g = g(f), f = f(x)$$

$$g'(x) = \frac{dg}{dx} = \frac{dg}{df}\frac{df}{dx}$$

Intuition

Rate of change of g relative to x is the product of

- rates of change of g relative to f and
- rates of change of f relative to x

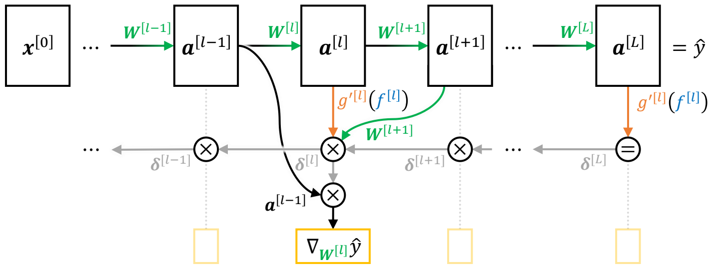
"If

- *a car travels 2x fast as a bicycle and*
- *the bicycle is 4x as fast as a walking man,*

then the car travels 2 × 4 = 8 times as fast as the man."

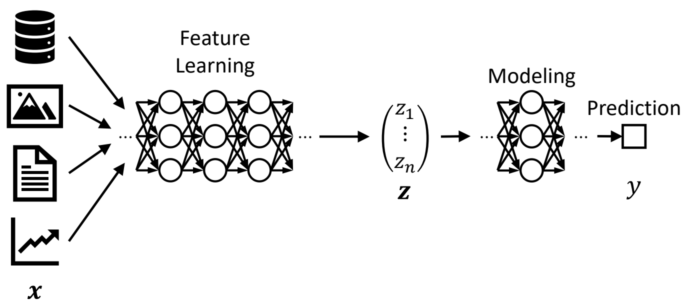
– George F. Simmons, Calculus with Analytic Geometry (1985)

Backward Propagation



$$\hat{y}'\left(\boldsymbol{W}^{[l]}\right) = \boldsymbol{a}^{[l-1]}\left(\delta^{[l]}\right)^\top \qquad \delta^{[l]} = \left[g'^{[l]}\left(f^{[l]}\right)\right] \circ \left(\boldsymbol{W}^{[l+1]}\delta^{[l+1]}\right)$$

From Manual Feature Engineering To Architecture Engineering

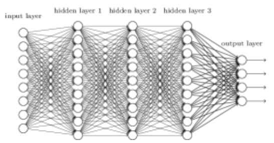


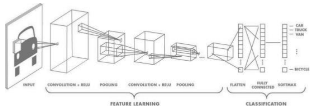
NUS CS3244: Machine Learning 19

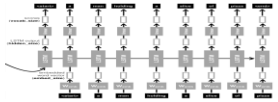
[W09a] Student Learning Outcomes

What did we learn?

- Feature Engineering → Architecture Engineering
- **CNN**: exploits spatial information using **convolutions**
- **RNN**: exploits history information using **recurrence**







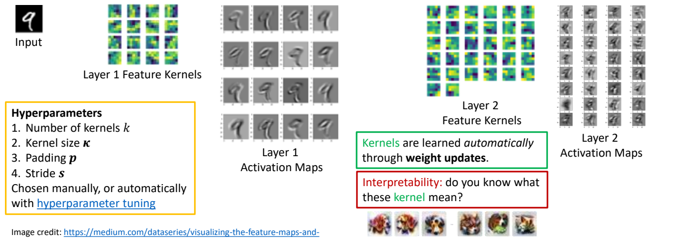
NUS CS3244: Machine Learning 20

CNN

Convolutional Layer: Feature Kernels & Feature Maps

$k^{[l-1]} = c^{[l]}$
filters from previous layer $k^{[l-1]}$ is equal to #channels into current layer $c^{[l]}$

$\boldsymbol{X}^{[0]} \to g^{[1]}(\boldsymbol{W}^{[1]} * \boldsymbol{X}^{[0]}) = \boldsymbol{A}^{[1]} \xrightarrow{\text{Pooling}} g^{[2]}(\boldsymbol{W}^{[2]} * \boldsymbol{X}^{[1]}) = \boldsymbol{A}^{[2]}$



Hyperparameters
1. Number of kernels k
2. Kernel size κ
3. Padding p
4. Stride s

Chosen manually, or automatically with hyperparameter tuning

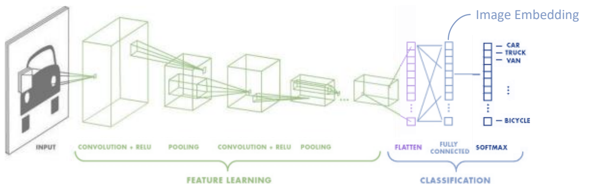
Kernels are learned *automatically* through **weight updates**.

Interpretability: do you know what these kernel mean?

Image credit: https://medium.com/dataseries/visualizing-the-feature-maps-and-filters-by-convolutional-neural-networks-e1462340518e

NUS CS3244: Machine Learning 39

Convolutional Neural Network



Key concepts

❶ Learn Spatial Feature
- Series of multiple convolution + pooling layers
- Progressively learn more diverse and higher-level features

❷ Flattening
- Convert to fixed-length 1D vector

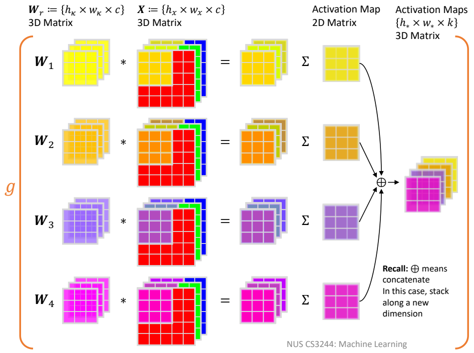
❸ Learn Nonlinear Features
- With fully connected layers (regular neurons)
- Learns nonlinear relations with multiple layers

❹ Classification
- Softmax ≔ Multiclass Logistic Regression
- Feature input = image embedding vector (typically large vector)

Image credit: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

NUS CS3244: Machine Learning 41

Multi-Channel Convolutions ($c = 3$ channels, $k = 4$ filters)



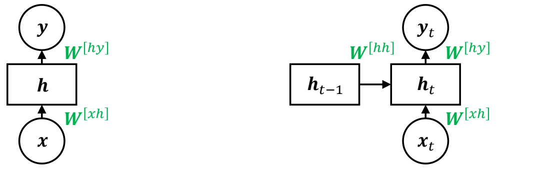
Key Takeaways
- Each Kernel convolves on **all channels** of image
- Each Activation Map is summed across channels
- 1 Activation Map per Kernel
- Kernel indexed in 3rd dimension of Activation Maps

NUS CS3244: Machine Learning 15

RNN

RNN Weights

Question: Do these weights change for different time t?



Feedforward Neural Network

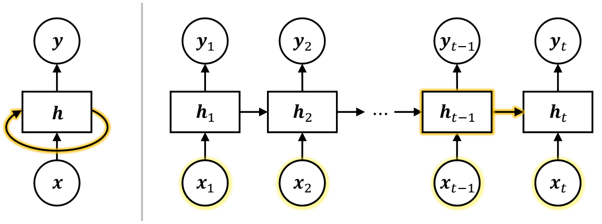
$$\boldsymbol{y} = g^{[y]}\left((\boldsymbol{W}^{[hy]})^\top \boldsymbol{h}\right)$$
$$\boldsymbol{h} = g^{[h]}\left((\boldsymbol{W}^{[xh]})^\top \boldsymbol{x}\right)$$

Recurrent Neural Network

$$\boldsymbol{y}_t = g^{[y]}\left((\boldsymbol{W}^{[hy]})^\top \boldsymbol{h}_t\right)$$
$$\boldsymbol{h}_t = g^{[h]}\left((\boldsymbol{W}^{[xh]})^\top \boldsymbol{x}_t + (\boldsymbol{W}^{[hh]})^\top \boldsymbol{h}_{t-1}\right)$$
$$\boldsymbol{h}_t = g^{[h]}\left((\boldsymbol{W}^{[xh]} \oplus \boldsymbol{W}^{[hh]})^\top (\boldsymbol{x}_t \oplus \boldsymbol{h}_{t-1})\right)$$

NUS CS3244: Machine Learning 30

Neurons with Recurrence



$$\hat{\boldsymbol{y}} = g^{[y]}\left(g^{[h]}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})\right)$$

Recurrent Neural Network (RNN)

$$\hat{\boldsymbol{y}} = g^{[y]}(\boldsymbol{h}_t)$$
$$\boldsymbol{h}_t = g^{[h]}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$$

NUS CS3244: Machine Learning 29

Training Issues

Deep Learning Training Issues → Mitigations

- Overfitting → Dropouts
- Saturating Gradient → ReLU Activation Function
- Vanishing Gradients → ResNet "Shortcuts", LSTM "Forget" Gates

Recap W11a – Explainable AI (XAI)

Interpretable “Glassbox” Models

Exercise E11a.1 Solution

How would you interpret?
Linear Regression

$$\hat{y} = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = \sum_{r=0}^{n} w_rx_r$$
$$= \boldsymbol{w} \cdot \boldsymbol{x} = \boldsymbol{w}^\top \boldsymbol{x}$$

$\hat{y} \propto w_rx_r, \forall r$

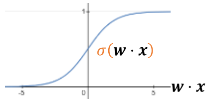
If $w_1 = kw_2$,
then x_1 is k times more important than x_2

Weighted Sum Interpretation	Gradient Interpretation
Bigger w_r means • **Larger** weight • More **importance** for to x_r • Direction? Supportive (positive) or opposing (negative) **influence**	Bigger w_r means • **Steeper** slope for x_r axis • Changes in x_r lead to bigger in $\hat{y}$ changes • More **importance** for x_r • Direction indicates increasing or decreasing **influence**

Exercise E11a.2 Solution

How would you interpret?
Logistic Regression

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}}$$
$$f = \boldsymbol{w} \cdot \boldsymbol{x} = \sum_{r=0}^{n} w_rx_r$$



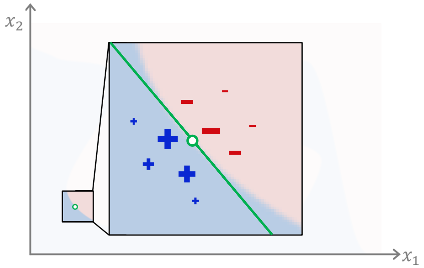
$\text{logit}(P(\hat{y})) \propto w_rx_r, \forall r$

If $w_1 = kw_2$, then log odds ratio of x_1 is k times bigger than of x_2

Weighted Sum Interpretation	Gradient Interpretation
Bigger w_r means • **Larger** importance • Direction indicates **influence**	Bigger w_r means? • **Steeper** slope for x_r near decision boundary • **Decision boundary** more *perpendicular* to x_r • Weight sign indicates direction of **pos/neg** prediction

LIME
Local, Model-Agnostic Explanations

LIME
Local Interpretable Model-agnostic Explanations



Prediction Model
$f(\boldsymbol{x})$
- Non-linear model of $\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \end{pmatrix}$
- Shown as curvy decision boundary

Explanation: **LIME**
1. Starting with **instance** $\boldsymbol{x}$ to explain
2. Focus on **Local** region
3. Training set as **neighbors** $\boldsymbol{x}^{(\eta)} \in X^{(\eta)}$
4. Train **surrogate model**, e.g., linear:

$$g(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} = \sum_{r=0}^{n} w_rx_r$$

Image Credit: https://santiagof.medium.com/model-interpretability-making-your-model-confess-lime-89db7f70a72b

LIME
Find “best” explainer g that minimizes $\xi(\boldsymbol{x})$

Python API: https://github.com/marcotcr/lime

$$\xi(\boldsymbol{x}) = \underset{g \in G}{\text{argmin}}\big(\underbrace{L(f, g, \pi_x)}_{\text{“Faithful”}} + \underbrace{\Omega(g)}_{\text{“Simple”}}\big)$$

f is the **predictor** function (model)
g is the **explainer** function (model)
$\pi_x(\boldsymbol{x}^{(\eta)})$ is the neighbor **proximity** function
- E.g., exponential decay $\exp\left(-\left(d(\boldsymbol{x}, \boldsymbol{x}^{(\eta)})\right)^2\right)$

$L(f, g, \pi_x)$ is the ***locally-weighted* error loss** function between predictor f and explainer g

$$L(f, g, \pi_x) = \sum_{\boldsymbol{x}^{(\eta)} \in X^{(\eta)}} \pi_x(\boldsymbol{x}^{(\eta)}) \left(f(\boldsymbol{x}^{(\eta)}) - g(\boldsymbol{x}^{(\eta)})\right)^2$$

$\Omega(g)$ is the **sparsity regularization**
- Want simpler explanation
  - ⇒ *fewer weights*
  - ⇒ Lasso (L1 norm)
- Penalizes if total weights is too large
- λ is hyperparameter on how much to penalize

$$\Omega(g) = \lambda \|w_r\|_1 = \lambda \sum_{r=1}^{n} w_r$$

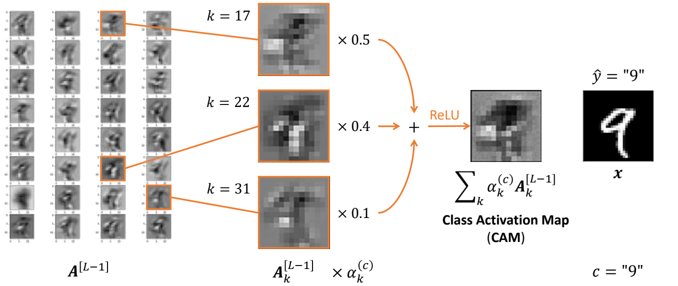
Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " Why should i trust you?" Explaining the predictions of any classifier. KDD'16.

Grad-CAM
Gradient-Weighted
Class Activation Maps

Grad-CAM Steps

1. Compute Activation Maps $\boldsymbol{A}^{[L]}$ of last conv layer L
   1. via Forward Propagation
2. Choose class label c to explain about (e.g., predict “9”, “car”)
3. Filter prediction $\hat{y}$ to be about class c
   1. Given: $\hat{\boldsymbol{y}} = \begin{pmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \hat{y}^{(c)} \\ \hat{y}^{(n)} \end{pmatrix}$, $\boldsymbol{e}^{(c)} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$, then $\hat{\boldsymbol{y}}^{(c)} = \hat{\boldsymbol{y}} \circ \boldsymbol{e}^{(c)} = \begin{pmatrix} 0 \\ 0 \\ y^c \\ 0 \end{pmatrix}$
   2. To generate explanation only for that class c
4. Compute importance weight $\alpha_k^{(c)}$ for each Activation Map $\boldsymbol{A}_k^{[L]}$
   1. Backprop from $\hat{\boldsymbol{y}}^{(c)}$ to get gradients (relative to activations) at last conv layer
5. Compute weighted sum with ReLU to get **Class Activation Map**

Grad-CAM example: Why did the CNN predict “9”?



k-Means Clustering

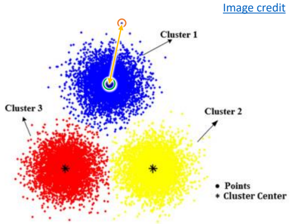
k-Means Objective

Minimize **Within-Cluster Sum-of-Squares (WCSS)** (i.e. variance)

$$L = \arg\min_{S} \sum_{c=1}^{k} \sum_{x \in S_c} \|x - \mu_c\|^2$$

Image credit

- $S = \{S_1, S_2, \dots, S_c, \dots, S_k\}$ is the set of all clusters
  - k is the total number of clusters
- S_c is the cth cluster of points
  - Note that c refers to cluster, not class
  - $x \in S_c$ refers to a point in cluster S_c
  - $\mu_c = \frac{1}{|S_c|}\sum_{x \in S_c} x$ is the centroid point in cluster S_c
  - $\|x - \mu_c\|^2$ refers to the squared Euclidean distance from x to μ_c



k-Means clustering algorithm

for $c = 1$ to k: — 1) Initialize cluster centroids
    $\mu_c \leftarrow \text{Random}()$

while not Converged(): — Iterative refinement
    for $j = 1$ to m: — 2) Assign datapoints to clusters
        $y^{(j)} \leftarrow c = \arg\min_c \|x^{(j)} - \mu_c\|^2$
        $x \leftarrow S_c$
    for $c = 1$ to k: — 3) Update cluster centroids
        $\mu_c \leftarrow \frac{1}{|S_c|}\sum_{x \in S_c} x$
    $t \mathrel{+}= 1$

return y — Cluster labels of all datapoints

How to choose k (number of clusters)?

- Use domain knowledge
- Note the k with *diminishing* return
  - When k is too high, marginal decrease in **within-cluster sum-of-squares** (WCSS)
  - How to see?
    - "Elbow" method

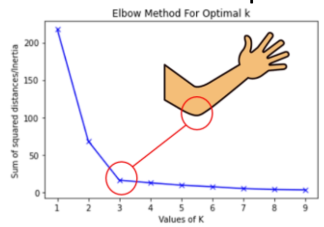


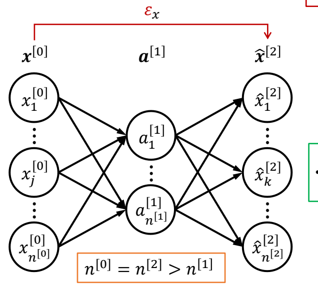
Image credit: https://www.analyticsvidhya.com/blog/2021/05/k-mean-getting-the-optimal-number-of-clusters/

Auto-Encoders

Auto-Encoder (with **Bottleneck**)

Reconstruction Loss Function
Squared Euclidean Distance
$\varepsilon_x(\hat{x}, x) = (\hat{x} - x)^\top(\hat{x} - x)$

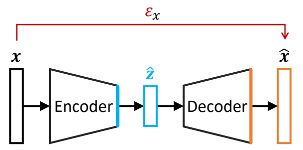
$M(x) = \hat{x}$



Benefits
- Compressed representation of x in the middle layer

Auto-Encoder Training

Reconstruction Loss Function
Squared Euclidean Distance
$\varepsilon_x(\hat{x}, x) = (\hat{x} - x)^\top(\hat{x} - x)$



$E(x) = \hat{z}$ $\quad D(\hat{z}) = \hat{x}$

$D(E(x)) = \hat{x}$

Take-Away:
Backprop gradient descent for weight update
$W \leftarrow W - \eta \frac{\partial \varepsilon_x}{\partial W}$
Same as all neural networks, but through 2 models

Gradient $\quad \frac{\partial \varepsilon_x}{\partial W} = \frac{\partial \hat{x}}{\partial W}\frac{\partial \varepsilon_x}{\partial \hat{x}} \qquad \hat{y} \coloneqq \hat{x}$

$$\frac{\partial \hat{x}}{\partial W} = \frac{\partial D}{\partial W} = \frac{\partial E}{\partial W}\frac{\partial D}{\partial E} = \frac{\partial E}{\partial W}\frac{\partial D}{\partial \hat{z}}$$

$$\frac{\partial D}{\partial W^{[l_D]}} = \frac{\partial f^{[l_D]}}{\partial W^{[l_D]}} \cdots \frac{\partial f^{[L_D]}}{\partial g^{[L_D-1]}}\frac{\partial g^{[L_D]}}{\partial f^{[L_D]}}$$

$l_E < l_D$

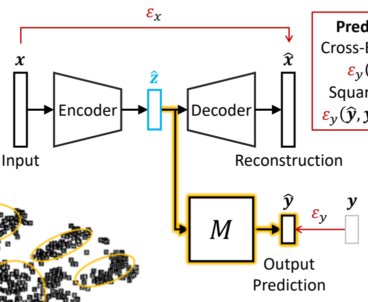
$$\frac{\partial E}{\partial W^{[l_E]}} = \frac{\partial f^{[l_E]}}{\partial W^{[l_E]}} \cdots \frac{\partial f^{[L_E]}}{\partial g^{[L_E-1]}}\frac{\partial g^{[L_E]}}{\partial f^{[L_E]}}$$

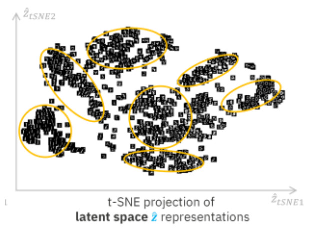
$$\frac{\partial D}{\partial W^{[l_E]}} = \frac{\partial E}{\partial W^{[l_E]}}\frac{\partial D}{\partial \hat{z}}$$

Auto-Encoder for **Feature Representation Learning**

Reconstruction Loss Function
Squared Euclidean Distance
$\varepsilon_x(\hat{x}, x) = (\hat{x} - x)^\top(\hat{x} - x)$

Prediction Loss Function
Cross-Entropy (Classification)
$\varepsilon_y(\hat{y}, y) = -y^\top \log \hat{y}$
Squared Error (Regression)
$\varepsilon_y(\hat{y}, y) = (\hat{y} - y)^\top(\hat{y} - y)$





Final Assessment Practice Review

Past Year + New Questions

NUS National University of Singapore
Department of Computer Science
School of Computing

Bonus (Optional) AMA

Reading Week, 17 Nov, Wed @ 4-6pm

Department of Computer Science
School of Computing

Goodbye and Good Luck

Final Exam

24 Nov (Wed) 16:00-18:00.

Venue: Zoom Proctor Rooms and also Seminar Room 1 (COM1 #02-06; SR1)

Physical, online notes, (online) calculator.
Not open internet / Web.

Stay in touch with our CS3244 alumni and general interest group on Facebook (search for "cs3244")

Thanks for joining us for the journey!

Photo Credits: People's Online Daily (English Version)