CS909/CS429
Revision

Dr. Fayyaz Minhas

Department of Computer Science
University of Warwick
https://warwick.ac.uk/fac/sci/dcs/teaching/material/cs909/

Structure of this lecture

- Online
- Will be Recorded

- At end
  – Questions on Moodle

Data Mining Objective

- Learning from Data
- Identifying Patterns in Data

- Generalization: Generating Correct Predictions for unseen data

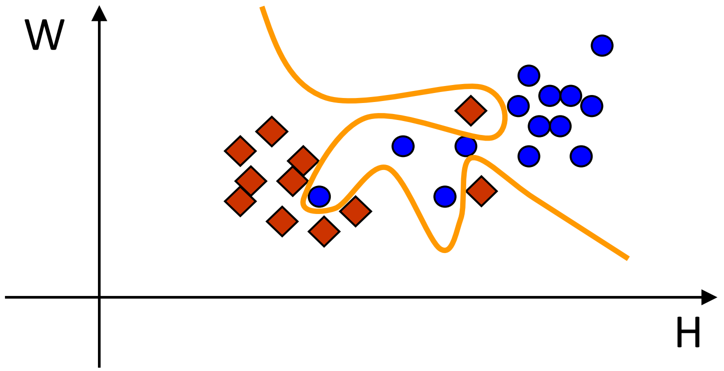
Updates

- Lectures now available on YouTube
  - https://www.youtube.com/playlist?list=PL9IcorxiyRbASB9DXjoWnBJO9RSKyzM2N
  - https://bit.ly/2S8hZZV
  - With improved captioning! ☺

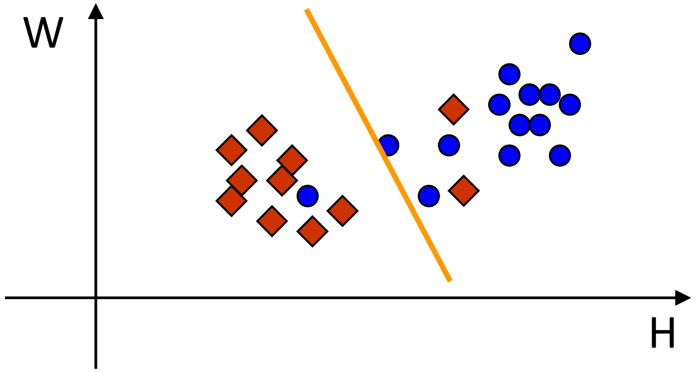
Generalization

- Generalization vs. Memorization
  - A particular issue in classification is the tradeoff between memorization vs. generalization
    - Remembering everything is not learning
    - The true test of learning is handling similar but unseen cases

Practice nearest neighbor distance calculations and classifications



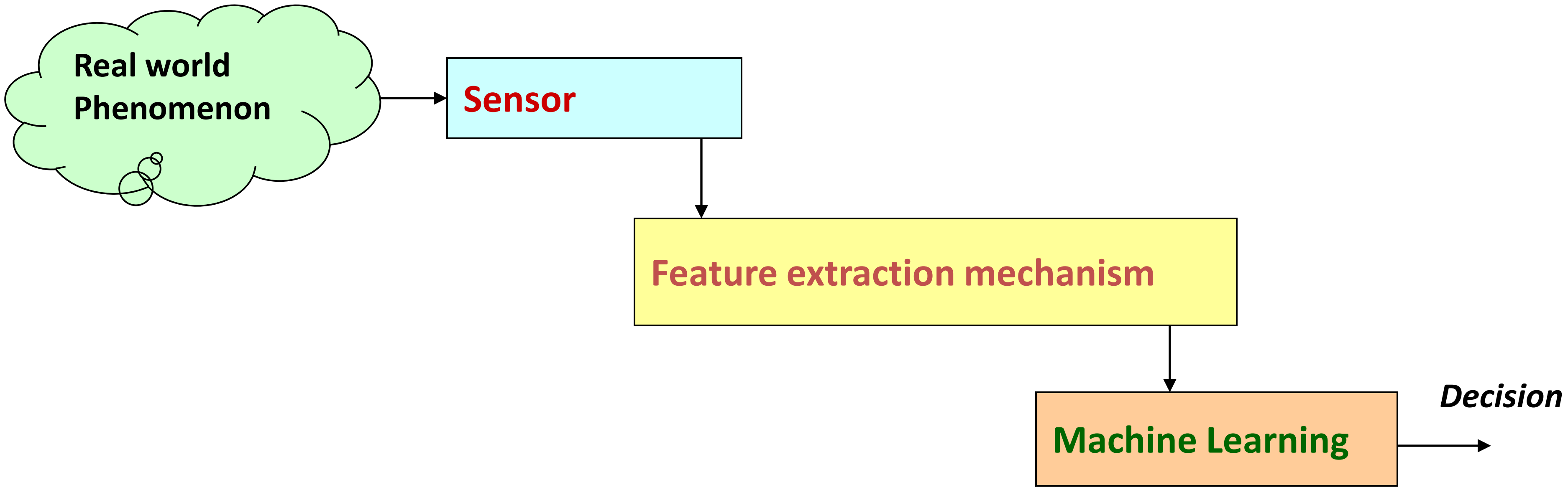
Has great memorization but may generalize poorly

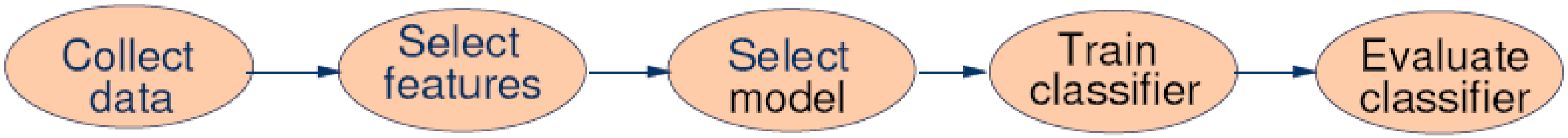


Has lesser memorization but may generalize better

Life Cycle

- Identify the objective
  - Identify the unit of classification (example)
    - Image block, protein sequence, ….





Types of ML problems

- Supervised Classification
  - Apple or orange
  - Inductive: Infer a rule for classification and use it to label unknown examples

Apple

Apple

Orange

Orange

- Multi-class Classification
  - Apple or orange or mango
  - For a binary classifier we can use
  - One vs. All
    - Apple vs. (Orange,Mango)
    - Orange vs. (Apple, Mango)
    - Mango vs, (Apple, Orange)
  - One against One
    - Apple vs. Orange
    - Apple vs. Mango
    - Orange vs. Mango

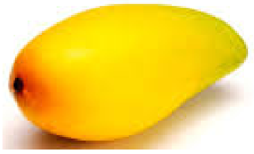
Apple

Apple

Orange

Orange

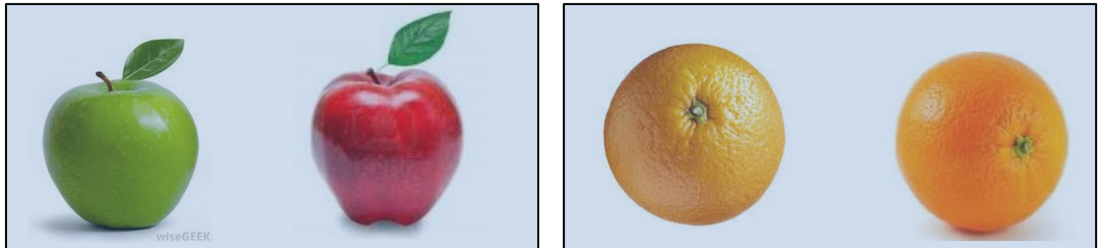
Mango



Mango

Types of ML problems

- **One Class Classification**
    - **Apple or not**
    - **Orange or not**
  - **One-Class SVM**



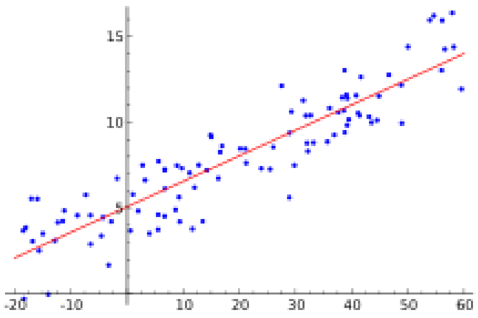
Apple Apple Orange Orange

- **Feature Selection**
  - **Select only the required features for classification**
  - **1-norm SVM**

Apple Apple Orange Orange

Types of ML problems

- Regression
  - Price of the apple vs. prices of the orange
  - Can be multi-variable in both input and output
  - Support Vector Regression
- Ranking
- Recommender Systems



- **Clustering**
  - **Unsupervised learning**
  - **Support Vector Clustering**
  - **Examples in one clusters should be similar (based on some criteria) to each other and different from other examples**
  - **Example: Apple sorting**

Composition of machine learning models

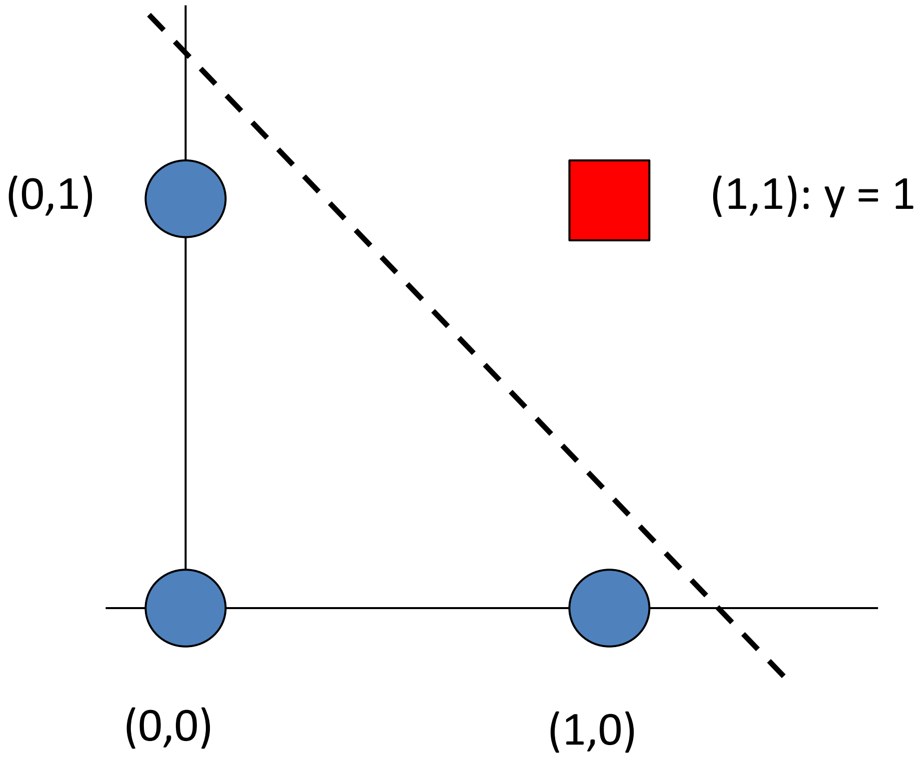
- Representation
  - How the model produces its output
  - Feature Representation
    - Denoted by a vector **x**
  - Linear
    - Perceptron $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}+b$
  - Non-linear
    - kNN: Assign class label to a novel example based on its nearest training example(s)

- Evaluation
  - Loss function

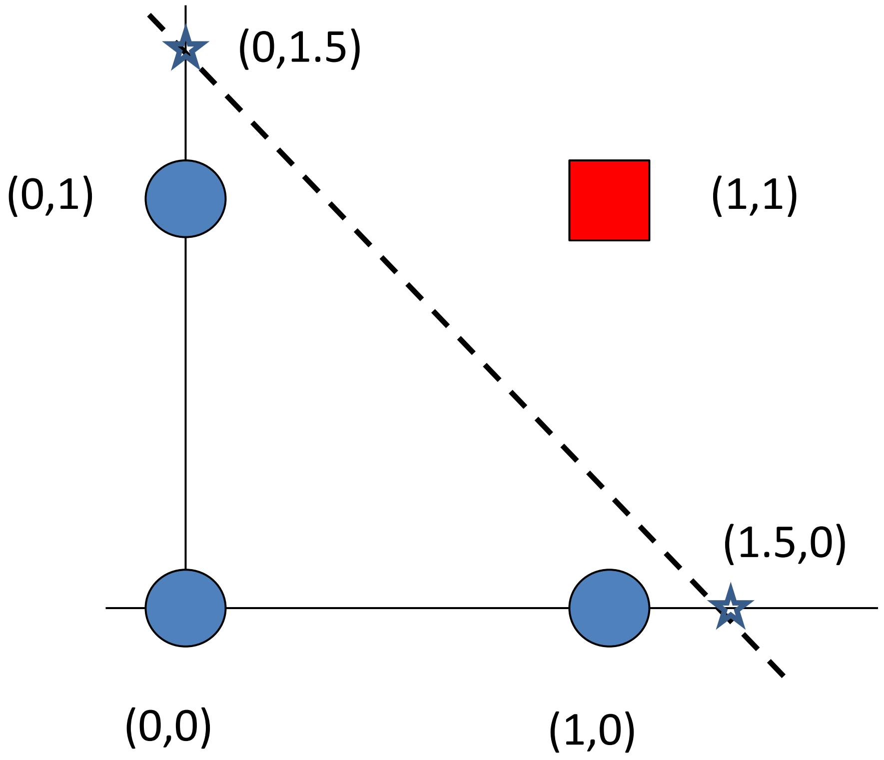
- Optimization
  - How to find parameters that minimize evaluation error

Linear Separability

$$f(\mathbf{x};\boldsymbol{\theta}) = w_1x^{(1)}+w_2x^{(2)}+b = 0$$



Example (Graphical Approach)



$f(\mathbf{x};\boldsymbol{\theta}) = w_1x^{(1)}+w_2x^{(2)}+b = 0$

$w_1(1.5)+w_2(0)+b = 0$

$b = -1.5w_1$

$w_1(0)+w_2(1.5)+b = 0$

$b = -1.5w_2$

If I set $w_1 = 1.0$

$b = -1.5$

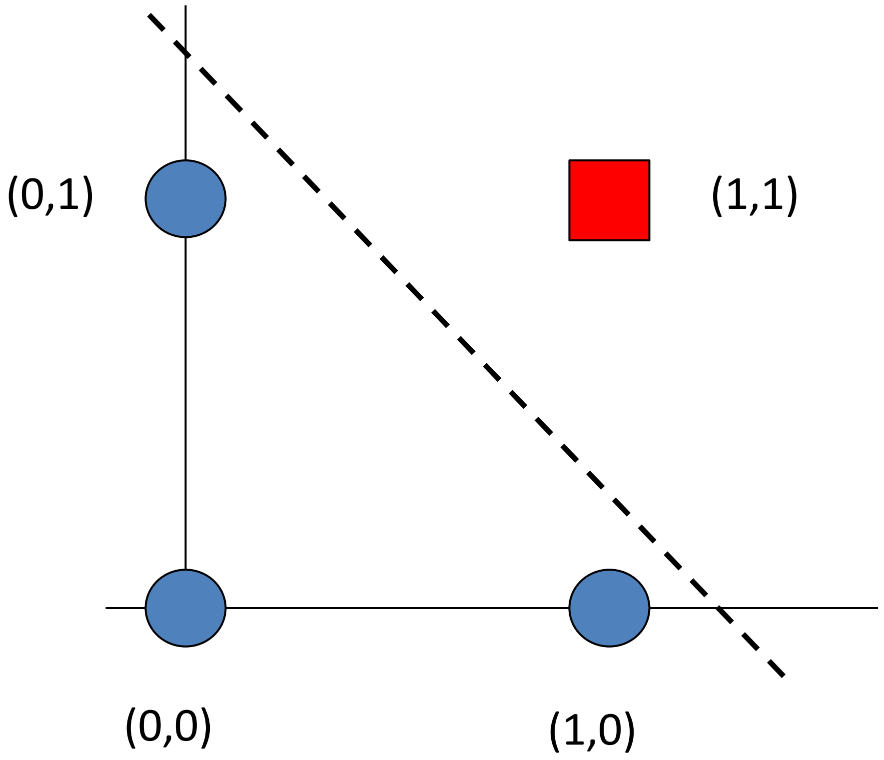
$w_2 = 1.0$

$f(\mathbf{x};\boldsymbol{\theta}) = x^{(1)}+x^{(2)} -1.5$

$(1,1) \rightarrow 0.5$

$(1,0) \rightarrow -0.5$, $(0,1) \rightarrow -0.5$, $(0,0) \rightarrow -1.5$

Example: Another Way (Algebraic Constraint Satisfaction)



$$f(\mathbf{x};\boldsymbol{\theta}) = w_1x^{(1)}+w_2x^{(2)}+b = 0$$

(1,1): $w_1(1.0)+w_2(1.0)+b > 0$

(1,0): $w_1(1.0)+w_2(0.0)+b < 0$

(0,1): $w_1(0.0)+w_2(1.0)+b < 0$

(0,0): $w_1(0.0)+w_2(0.0)+b < 0$

$w_1+w_2+b > 0$

$w_1+b < 0$

$w_2+b < 0$

$b < 0$

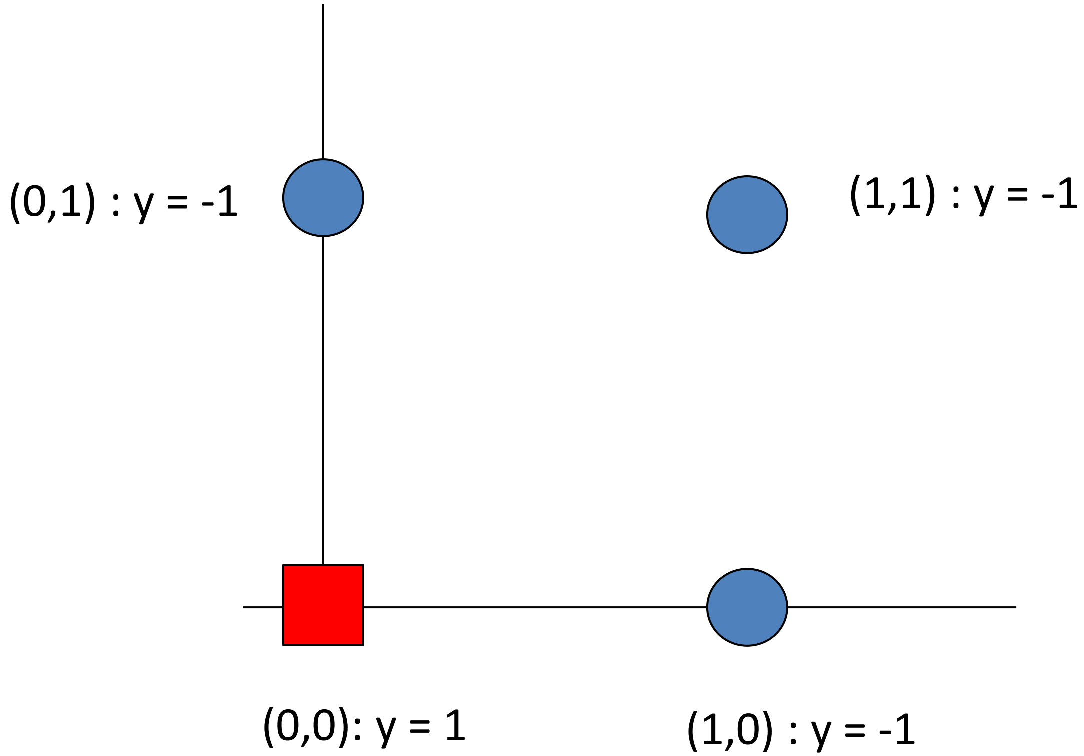
$b = -1.5$

$w_1= 1.0$

$w_2= 1.0$

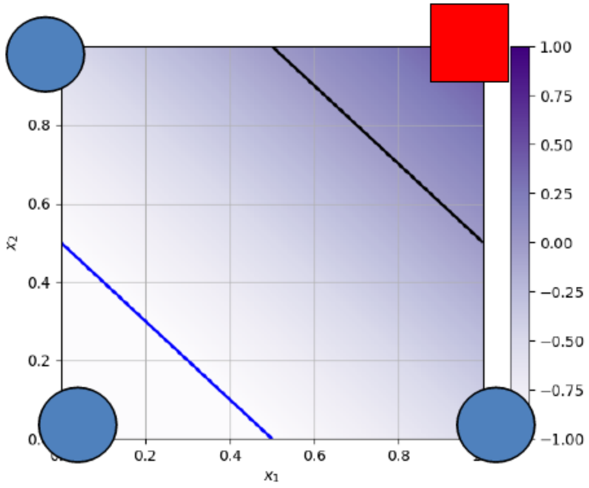
Exercise

- Is this problem linearly separable?



Error Minimization

- Representation
  - How does the model generate its output
  - $f(\mathbf{x};\mathbf{w}) = w_1x^{(1)}+w_2x^{(2)}+\ldots+w_2x^{(d)}+b = \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}$
- Evaluation
  - Define what constitutes as a prediction error
  - $L(\boldsymbol{X},\boldsymbol{Y};\boldsymbol{w}) = \sum_{i=1}^{N}(f(\boldsymbol{x}_i;\mathbf{w})-y_i)^2$
- Optimization
  - $\boldsymbol{w}^* = argmin_{\boldsymbol{w}}L(\boldsymbol{X},\boldsymbol{Y};\boldsymbol{w}) = \boldsymbol{X}^+\boldsymbol{y}$
- Code
  - `w = np.linalg.pinv(X)@y`
- Evaluate

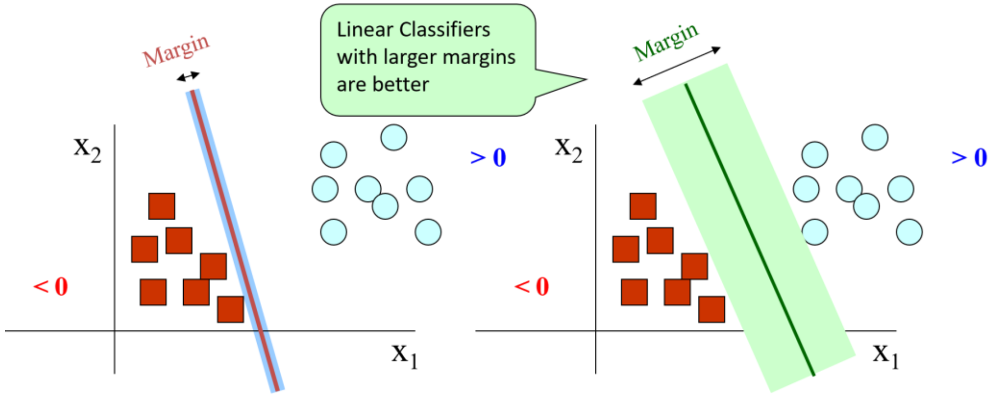


Examples of REO

- Try writing the
  - Representation
  - Evaluation (loss and regularization)
  - Optimization
- Of
  - OLS
  - Perceptron
  - SVM
  - MLP
  - Clustering problems
  - Ranking problems

Evaluation: Structural Risk Minimization

- Loss or error
  - Hinge Loss
  - Squared Loss
  - Cross-entropy loss
  - Can you plot these?
  - But it is not enough!!



- Regularization
  - A small change in the input should not have a large impact on the output
  - Related to Margin and “Freedom” or “Complexity” of the classifier
    - Related to the “VC Dimension” of the data
    - Do read about it!

Example of SRM: SVM

- Representation

  f(**x**;**w**) = $w_1x^{(1)}+w_2x^{(2)}+\ldots+w_2x^{(d)}+b = \boldsymbol{w^T x} + b$

- Evaluation & Optimization

$$\min_{\boldsymbol{w}} R(f) + CL(\boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{w})$$

structural risk

(Inverse of) Margin
AKA
Regularization

C > 0 is a weighting factor that controls the relative contribution of both

Empirical Error/Loss over Training Data

$$R(f) = \frac{1}{2}\boldsymbol{w^T w} = \frac{1}{2}\|\boldsymbol{w}\|^2$$

$$L(\boldsymbol{X}, \boldsymbol{Y}; \boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} \max\{0, 1 - y_i f(\boldsymbol{x}_i; \mathbf{w})\}$$

$$\min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w^T w} + \frac{C}{N}\sum_{i=1}^{N} \max\{0, 1 - y_i f(\boldsymbol{x}_i; \mathbf{w})\}$$

- Other loss functions
  - Cross-entropy, 0-1 loss, squared loss…

Regularization

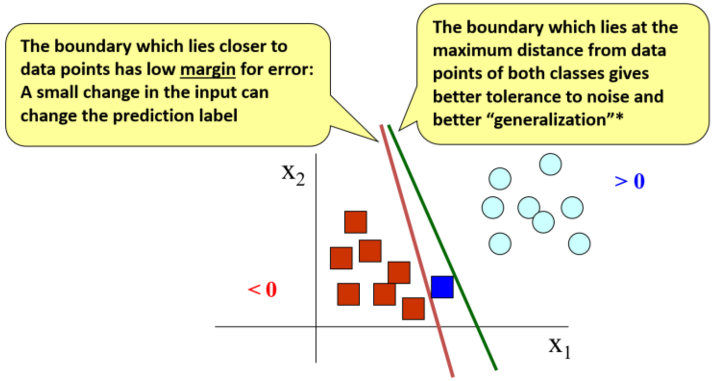
- Small changes in input should produce small changes in output
  - Achieved by minimization of the norm of the weight vector

$$R(\boldsymbol{w}) = \|\boldsymbol{w}\|_2^2 = w_1^2 + w_2^2 + \cdots + w_d^2$$

- In general

$$\|\boldsymbol{w}\|_p = (|w_1|^p + |w_2|^p + \cdots + |w_d|^p)^{1/p}$$
$$\|\boldsymbol{w}\|_1 = |w_1| + |w_2| + \cdots + |w_d|$$
$$\|\boldsymbol{w}\|_0 = number\ of\ non-zero\ vector\ elements$$

- Enables generalization esp. when the number of data points is quite small in comparison to the number of dimensions of each data point: A cure to the Curse of dimensionality
  - Given only training examples, optimizing empirical error over only a small number of training examples can lead to models that do not generalize to unseen examples effectively



Small weights limit "the butterfly effect"

- *Let's quantify how sensitive the model is to a perturbation of its input*

- $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$
- $\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{\delta x}) = \boldsymbol{w}^T(\boldsymbol{x}+\boldsymbol{\delta x}) + \boldsymbol{b} = \boldsymbol{w}^T\boldsymbol{x} + \boldsymbol{b} + \boldsymbol{w}^T\boldsymbol{\delta x} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{w}^T\boldsymbol{\delta x}$
- $\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{\delta x}) - \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{\delta x}$
- $\|\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{\delta x}) - \boldsymbol{f}(\boldsymbol{x})\| = \|\boldsymbol{w}^T\boldsymbol{\delta x}\| \le \|\boldsymbol{w}\|\|\boldsymbol{\delta x}\|$ **(using Cauchy-Schwarz inequality)**

- **Therefore,** $\frac{\|\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{\delta x}) - \boldsymbol{f}(\boldsymbol{x})\|}{\|\boldsymbol{\delta x}\|} \le \|\boldsymbol{w}\|$

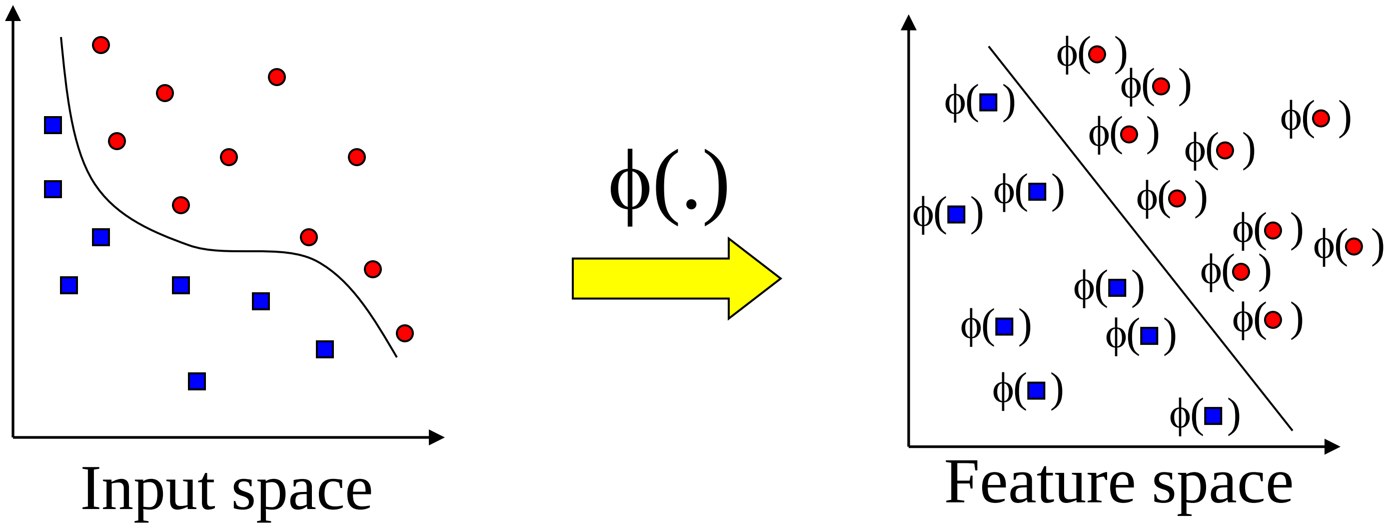
Change in model output per unit additive change in input is upper bounded by $\|\boldsymbol{w}\|$.

Consequently, minimizing the norm of the weight vector (or its square) would lead to a regularization effect as it would limit the effect of any change in the input on the output.

Vapnik showed that **minimizing "structural risk"** (combination of empirical error over training examples and the norm of the weight vector) **leads to minimization of the upper bound on generalization error over unseen examples effectively achieving a solution to the curse of dimensionality.**

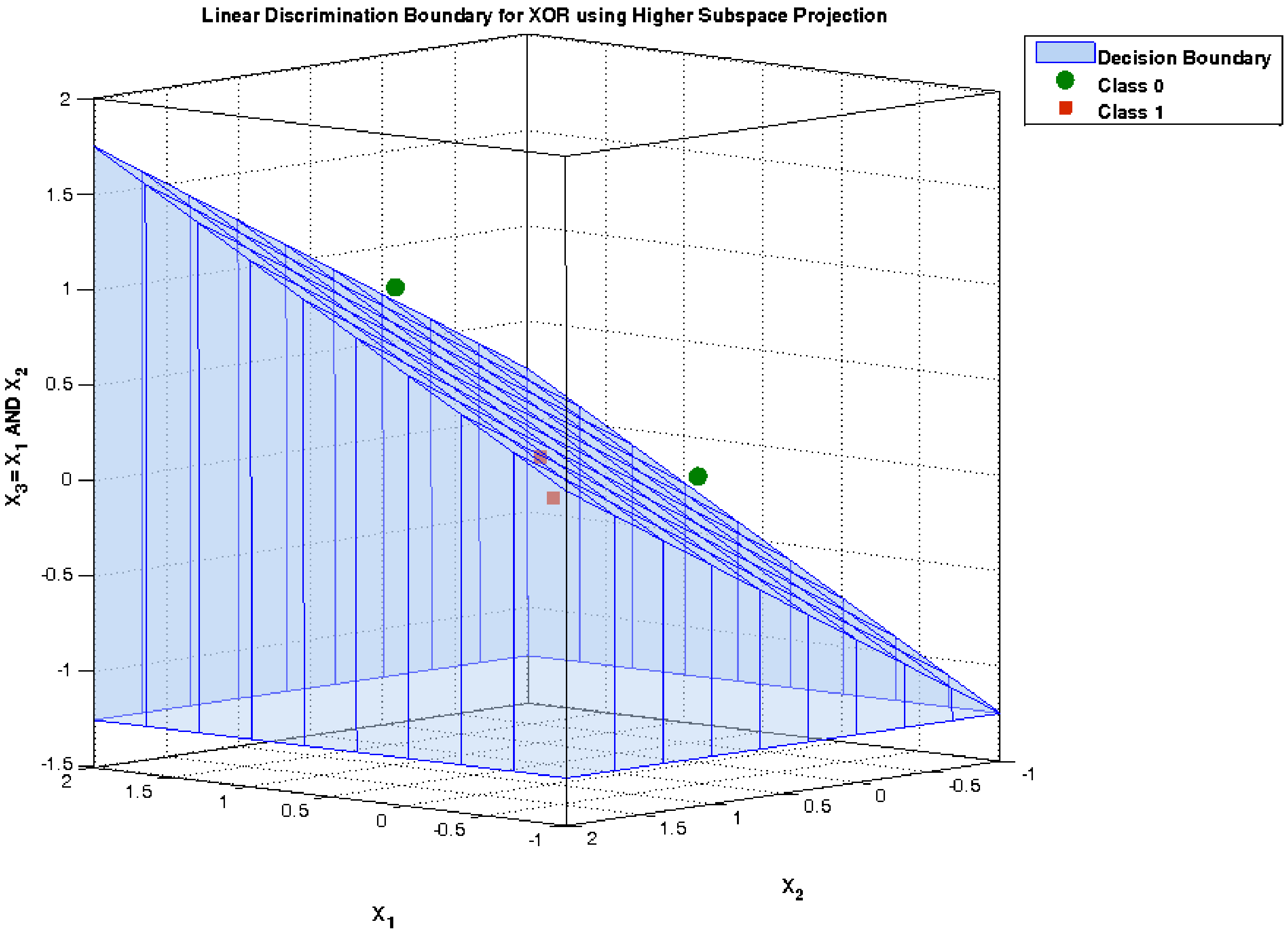
$$R(\boldsymbol{w}) \le R_{emp}(\boldsymbol{w}) + \Omega\left(\frac{1}{N}, \frac{1}{\|\boldsymbol{w}\|}, d\right)$$

Feature Transformations & Kernels



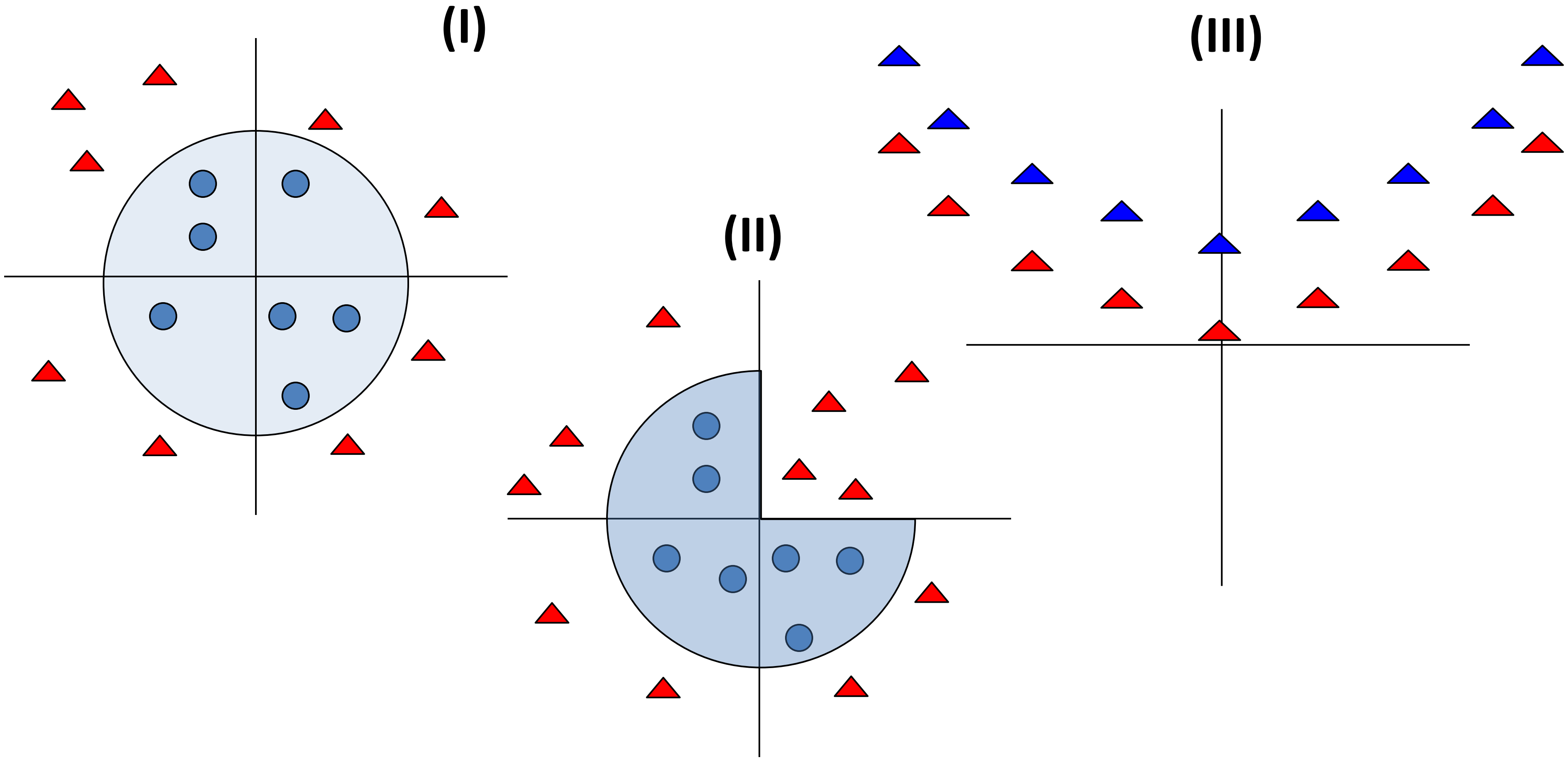
- Relevant features can lead to better accuracy
- Large number of features in a feature transformation can

XOR Linear Separability



Transformation Examples

- Can you find a transform that makes the following classification problems linear separable? Can you draw the data points in the new transformed feature space?



Effect of feature transformation

- A feature transformation changes the distance (or similarity) between points
- Can also be achieved through kernel functions by the "Kernel Trick"

$$\min_w \frac{1}{2}\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w} + \frac{C}{N}\sum_{i=1}^{N}\max\{0, 1-y_i f(\boldsymbol{x}_i; \mathbf{w})\} \xrightarrow{\boldsymbol{w} = \sum_{i=1}^{N}\alpha_i \boldsymbol{x}_i} \min_{\boldsymbol{\alpha},b} \sum_{i,j=1}^{N}\alpha_i\alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \frac{C}{N}\sum_{i=1}^{N}\max\left\{0, 1-y_i\left(b + \sum_{j=1}^{N}\alpha_j k(\boldsymbol{x}_{\boldsymbol{i}}, \boldsymbol{x}_j)\right)\right\}$$

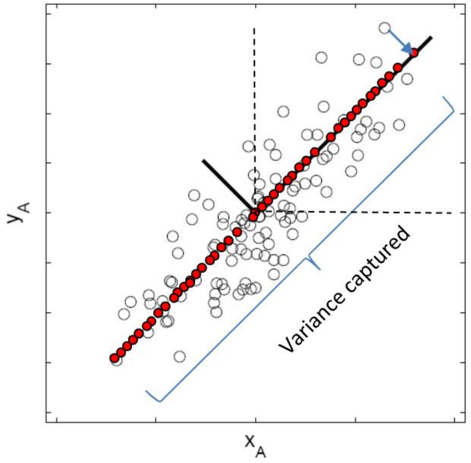
$$\min_{\boldsymbol{\alpha},b} \sum_{i,j=1}^{N}\alpha_i\alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \frac{C}{N}\sum_{i=1}^{N}\max\left\{0, 1-y_i\left(b + \sum_{j=1}^{N}\alpha_j k(\boldsymbol{x}_{\boldsymbol{i}}, \boldsymbol{x}_{\boldsymbol{j}})\right)\right\}$$

Comprehension Questions

- What are the different types of kernels?
  - What are linear and non-linear kernel functions?
  - What makes a kernel a valid kernel?
- What is the role of C?

Dimensionality Reduction

- PCA
  - Project data along the directions of large variance in the data
    - Proof: Directions of maximum variance are along the direction of the Eigen vectors of the covariance matrix of the data
  - Look at the proofs!
  - Can you identify directions of maximum variance in the data and write their unit vectors?



https://github.com/foxtrotmike/PCA-Tutorial/blob/master/Minhas-PCA.pdf

Other ML Problems

- Regression
  - Loss functions: squared error, absolute loss, huber loss, epsilon insensitive loss
  - Performance Metrics
    - MAE/MSE
    - R2
    - Correlation Coefficient
- Clustering
  - Hierarchical Clustering
  - kmeans
- One-Class Classification
- Ranking
- Recommender Systems

Representation: $\mathrm{f}(\mathbf{x};\mathbf{w},\mathrm{b}) = \boldsymbol{w}^T\boldsymbol{x} + \mathrm{b}$ or kernelized $\mathrm{f}(\mathbf{x};\boldsymbol{\alpha},\mathrm{b}) = \mathrm{b} + \sum_{j=1}^{N}\alpha_j k(\boldsymbol{x},\boldsymbol{x}_j)$ via the Representer Theorem with Structural Risk Minimization under the general form

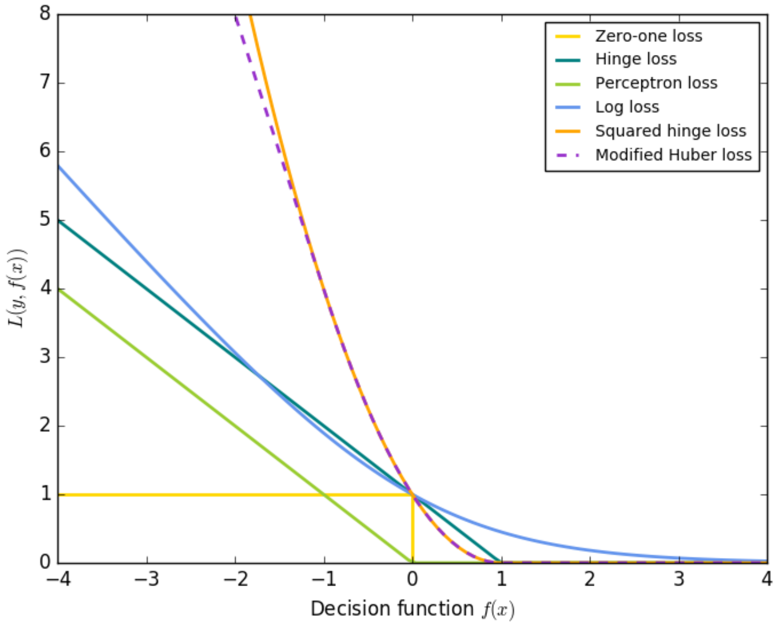
$$\min_{\boldsymbol{w}} \lambda R(\boldsymbol{w}) + E[error\ or\ loss\ over\ training\ examples]$$

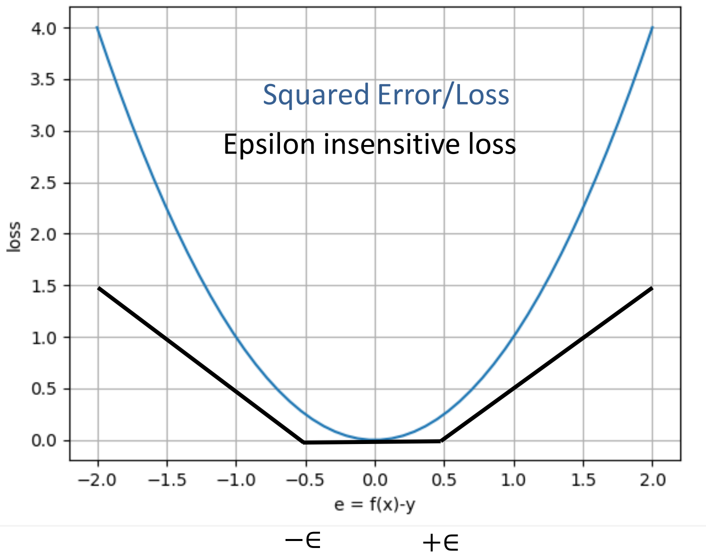
$R(\boldsymbol{w})$ is the regularization term and SRM provides a bound on generalization error. The goal is to minimize the expected error but under i.i.d. assumption $\mathrm{E}[\mathrm{loss}] = \frac{1}{N}\sum_{i=1}^{N} l(f(\boldsymbol{x}_i), y_i)$

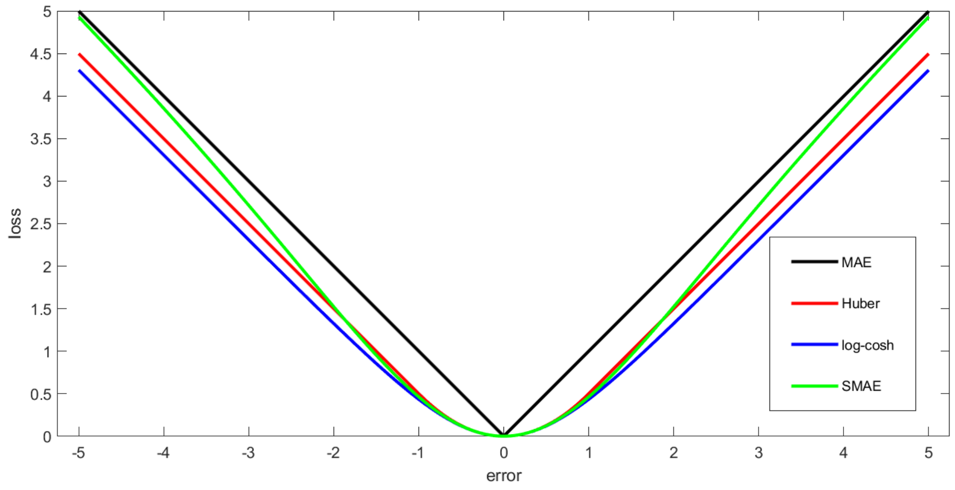
Name	Evaluation (Optimization Problem)	Explanation
Perceptron	$min_w \sum_{i=1}^{N} max(0, 1 - y_i f(\boldsymbol{x};\boldsymbol{w}))$	Uses hinge loss for classification
SVC (Linear)	$min_w \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{i=1}^{N} max(0, 1 - y_i f(\boldsymbol{x};\boldsymbol{w}))$	Regularized Perceptron
SVC (Kernelized)	$\min_{\alpha,b} \frac{\lambda}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j k(\boldsymbol{x}_i,\boldsymbol{x}_j) + \frac{1}{N}\sum_{i=1}^{N}\max\left\{0, 1 - y_i\left(b + \sum_{j=1}^{N}\alpha_j k(\boldsymbol{x}_i,\boldsymbol{x}_j)\right)\right\}$	Kernelized SVC
Logistic Regression	$\min_{\boldsymbol{w},b} \frac{1}{2}\lVert\boldsymbol{w}\rVert^2 + \frac{C}{\boldsymbol{N}}\sum_{\boldsymbol{i=1}}^{N}\log(\exp(-y_i f(x_i)) + 1)$	Uses the logistic loss for classification.
PCA	$\min_{\boldsymbol{w}} \boldsymbol{\lambda w^T w} + (\boldsymbol{V} - \boldsymbol{w^T C w})$	Find (orthogonal) direction(s) by minimizing the loss in variance after projection
OLS	$\min_w \sum_{i=1}^{N}(\boldsymbol{w^T x_i} - y_i)^2 = \lVert\boldsymbol{Xw} - \boldsymbol{y}\rVert^2$	Find best linear regression fit under squared loss
SVR (Linear)	$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w^T w} + \frac{C}{\boldsymbol{N}}\sum_{\boldsymbol{i=1}}^{N} max(0, \lvert f(\boldsymbol{x_i}) - y_i\rvert - \epsilon)$	Uses epsilon-insensitive loss for regression
SVR (Kernelized)	$\min_{\alpha,b} \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j\, k(\boldsymbol{x_i},\boldsymbol{x_j}) + \frac{C}{\boldsymbol{N}}\sum_{\boldsymbol{i=1}}^{N}\boldsymbol{max}\left(0, \left\lvert\sum_{j=1}^{N}k(\boldsymbol{x_i},\boldsymbol{x_j}) + b - y_i\right\rvert - \epsilon\right)$	Kernelized form of the above
Ridge Regression	$\min_{\boldsymbol{w},b} \alpha\lVert\boldsymbol{w}\rVert^2 + \lVert\boldsymbol{Xw} - \boldsymbol{y}\rVert^2$	OLS with regularization (squared norm)
Lasso	$\min_{\boldsymbol{w},b} \alpha\lVert\boldsymbol{w}\rVert_1 + \lVert\boldsymbol{Xw} - \boldsymbol{y}\rVert^2$	Use 1-norm regularization (minimize sum of absolute values rather than their squares)
Elastic Net	$\min_{\boldsymbol{w},b} \alpha\rho\lVert\boldsymbol{w}\rVert_1 + \frac{\alpha(1-\rho)}{2}\lVert\boldsymbol{w}\rVert^2 + \lVert\boldsymbol{Xw} - \boldsymbol{y}\rVert^2$	Uses both types of regularization
Huber Regressor	$\min_{\boldsymbol{w},b} \alpha\lVert\boldsymbol{w}\rVert^2 + \sum_{i=1}^{N} l_{huber}(f(x_i, y_i))$ with $l_{huber}(f(x_i, y_i)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & if\ \lvert y - f(x)\rvert < \delta \\ \delta(\lvert y - f(x)\rvert - \frac{1}{2}\delta) & else \end{cases}$	Used for robust regression as huber loss is less sensitive to outliers than squared loss

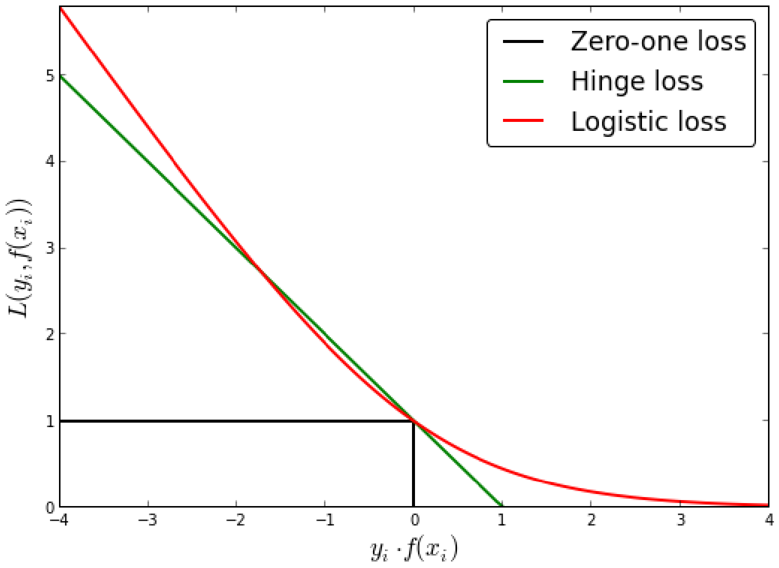
Loss Functions: $l(f(x_i, y_i))$

- Quantify Error
  - Misclassification
  - Misregression
  - Misreconstruction
  - Misclustering, Misranking, Misretrieval, ….
- The loss function determines the behaviour of the predictor
- More importantly, it determines the type of ML problem being solved
- Loss functions on the previous slide are all convex losses
  - Guaranteed single minima and convergence through gradient descent
  - Some even lead to closed form optimization which is great
  - However: LeCun, Yann. "Who is afraid of non-convex loss functions." *NIPS Workshop on Efficient Machine Learning*. 2007.
- A loss function doesn't even have to operate at a per-example level



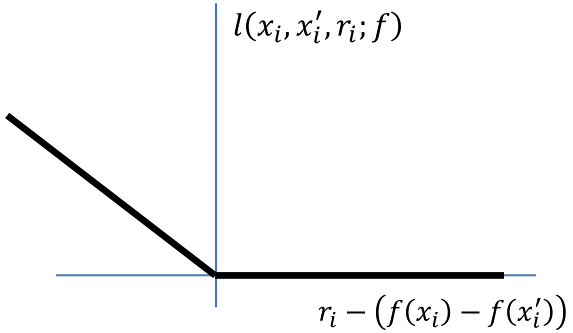






Generalized Instance Ranking Problem

- Misclassification vs. mis-ranking
  - Mis-classification: assign wrong classification label
  - Mis-ranking: one example should have been ranked higher than the other but is not
- Generalized ranking loss:



$$l(x_i, x_i', r_i; f) = \max\left(0, r_i - \left(f(x_i) - f(x_i')\right)\right)$$

$$\boldsymbol{min}_{\boldsymbol{w},\boldsymbol{b}}\ \frac{\mathbf{1}}{\mathbf{2}}\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w} + \frac{C}{N}\sum_{i=1}^{N} l(x_i, x_i', r_i; f)$$

ML Task	ML Task
Classification (Binary and Multi-class: OVR, OVA, etc)	Out of Domain Detection
Regression	**Novelty Detection/One-Class Classification**
Dimensionality Reduction / Decomposition	Retrieval / Vector Database Search
Clustering and Biclustering	Prediction under domain shift or concept drift
Statistical Inference and Hypothesis testing	Counterfactual prediction
Recommender System, Basket (item co-occurrence analysis)	**Zero and Few Shot Prediction**
Learning to Rank (Ordinal Regression)	**Semi-Supervised Learning**
Generative Modelling: Conditional and Unconditional	Weakly-supervised and multiple instance learning
Multi-task Prediction	Causal Learning, Inference, Discovery & Counterfactual prediction
Multi-Label Prediction	Active Learning
Survival Prediction (Churn Prediction or Failure Prediction)	Meta Learning
Adaptive Prediction Sets & Conformal Prediction	Curriculum Learning
Meta-Learning: Learning to learn and learning to optimize	Transfer Learning
Representation Learning	Contrastive and self-taught Learning
Open Set Recognition	Online and Continuous Learning and Unlearning
Subset Discovery	Reinforcement learning
Domain Specific tasks - CV: Object detection, localization, counting, instance segmentation, semantic segmentation, image to image regression. NLP: Tokenization, Embeddings, Next word prediction…	Structured Output Learning **Topic Modeling**, Machine Translation, Community discovery, graph learning, time series forecasting, …

Performance Evaluation

- Objective
  - How good is my ML model pipeline?
  - What parameters should I pick?
  - What am I doing wrong?
- Cross-validation
- Metrics
  - Accuracy, Balanced Accuracy
  - AUC-ROC, AUC-PR
- All metrics have assumptions and limitations
  - Try understanding those!

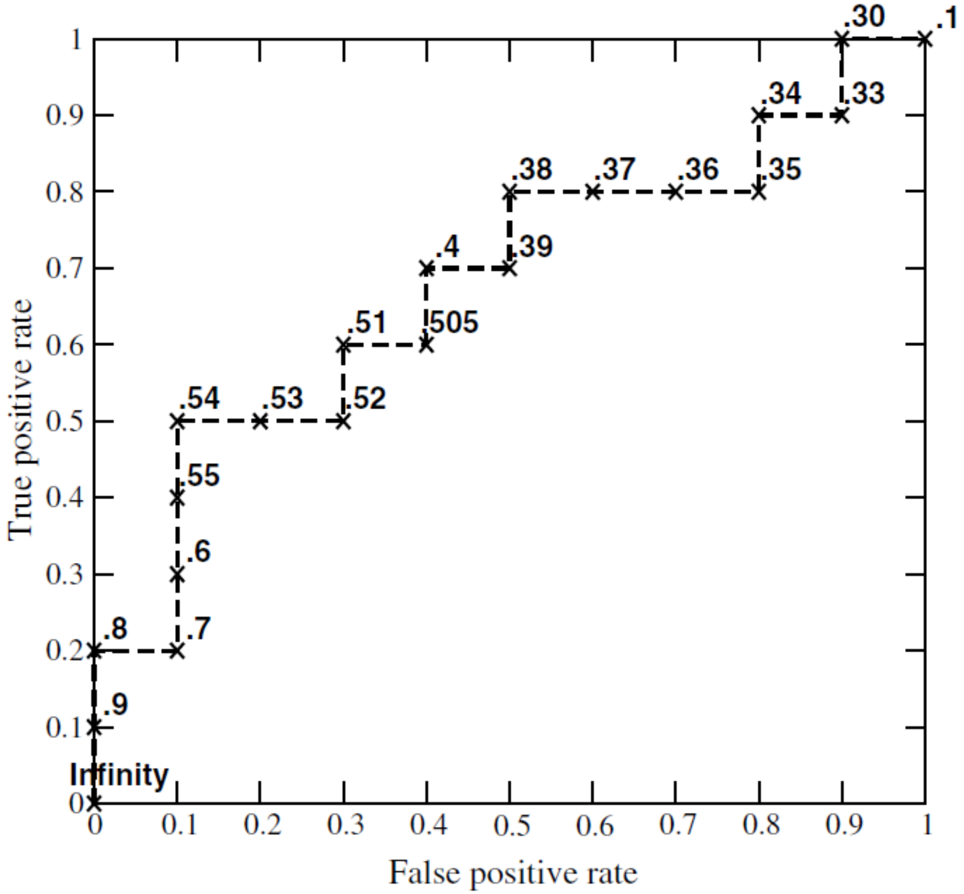
Confusion Matrix

		True condition			
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive** (Type I error)	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$
	Predicted condition negative	**False negative** (Type II error)	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$
	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	True positive rate (TPR), Sensitivity, Recall $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$	

https://en.wikipedia.org/wiki/Sensitivity_and_specificity

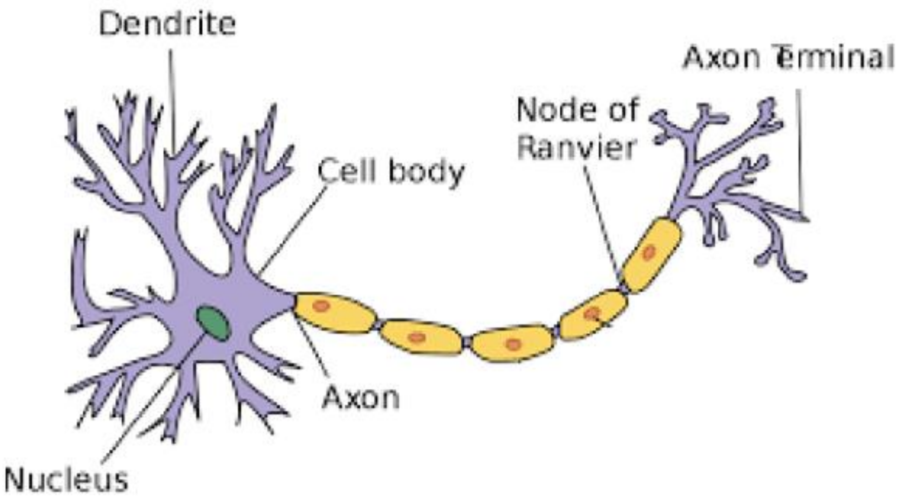
Making the ROC Curve

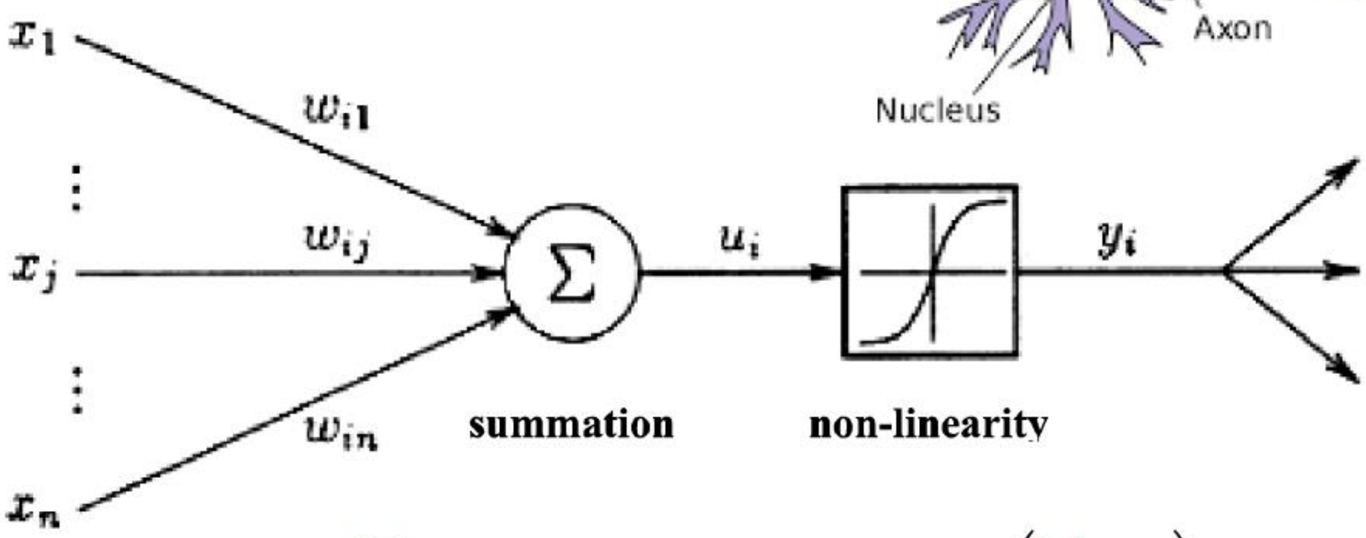
Inst#	Class	Score	Inst#	Class	Score
1	**p**	.9	11	**p**	.4
2	**p**	.8	12	**n**	.39
3	**n**	.7	13	**p**	.38
4	**p**	.6	14	**n**	.37
5	**p**	.55	15	**n**	.36
6	**p**	.54	16	**n**	.35
7	**n**	.53	17	**p**	.34
8	**n**	.52	18	**n**	.33
9	**p**	.51	19	**p**	.30
10	**n**	.505	20	**n**	.1



Neural Networks

$$y_i = f(u_i) = \sum_j w_{ij} x_j$$



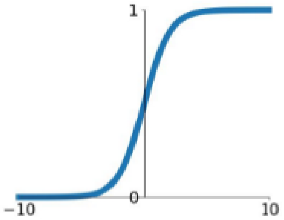


$$u_i = \sum_j w_{ij} x_j = \mathbf{w}_i^T \mathbf{x} \qquad y_i = f(u_i) = f\left(\sum_j w_{ij} x_j\right)$$

Activation Functions

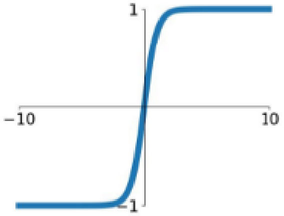
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



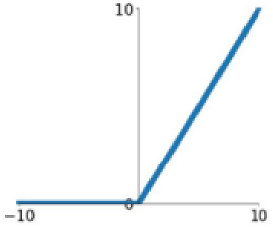
tanh

$\tanh(x)$



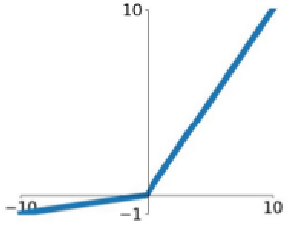
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

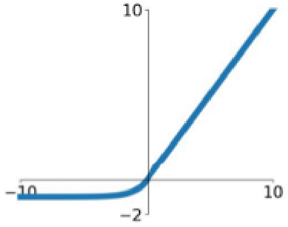


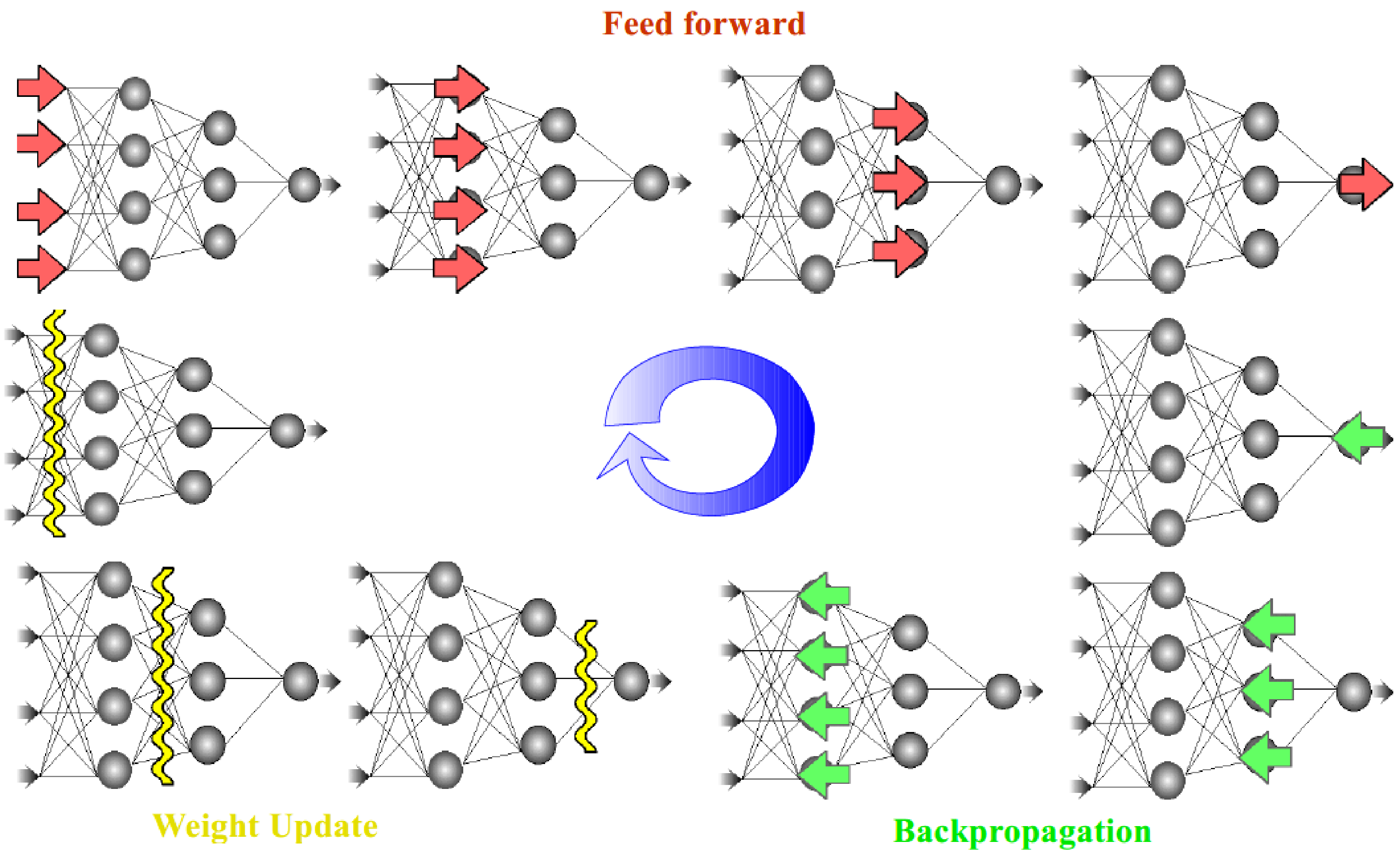
Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$




Feed forward
Weight Update
Backpropagation

By the Chain rule, we have:

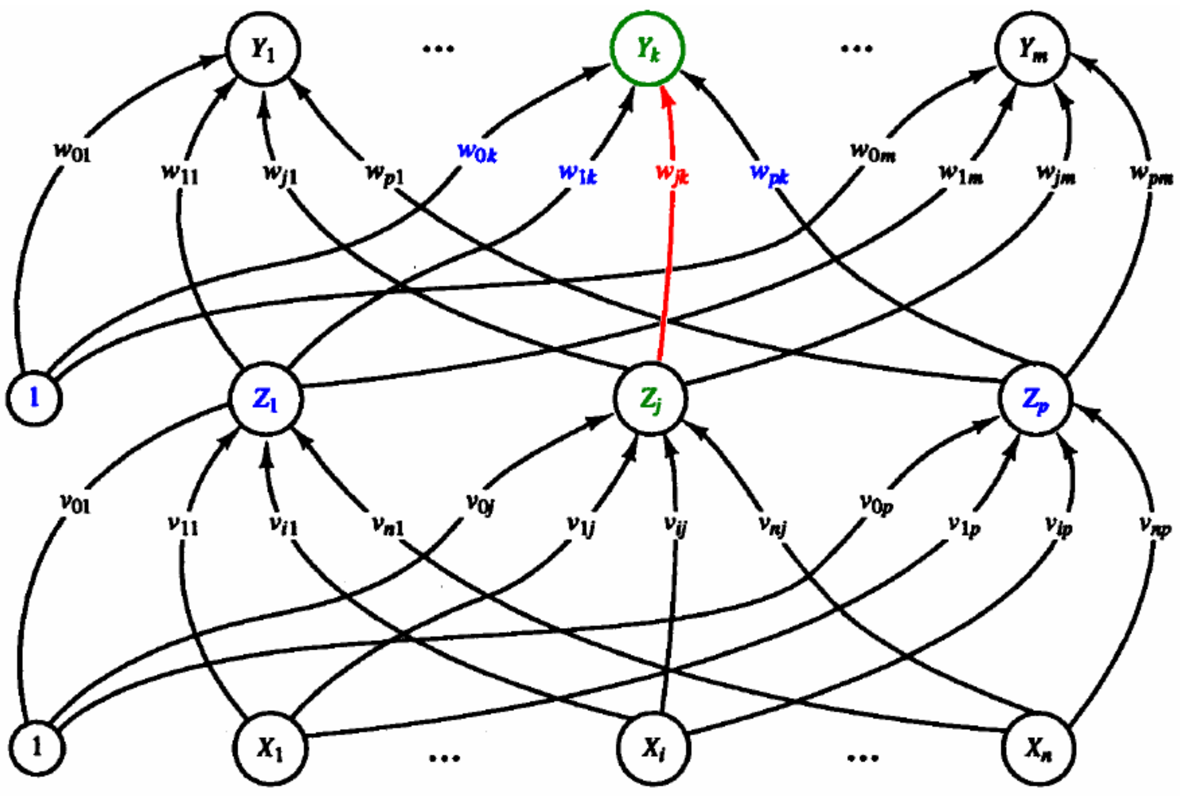
$$E = 0.5\sum_k (t_k - y_k)^2$$

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} 0.5\sum_k (t_k - y_k)^2$$
$$= \frac{\partial}{\partial w_{jk}} 0.5(t_k - y_k)^2$$

Change in w_{jk} affects only y_k

$$= -(t_k - y_k)\frac{\partial}{\partial w_{jk}} y_k$$
$$= -(t_k - y_k)\frac{\partial}{\partial w_{jk}} f(y\_in_k)$$
$$= -(t_k - y_k)f'(y\_in_k)\frac{\partial}{\partial w_{jk}} y\_in_k$$
$$= -(t_k - y_k)f'(y\_in_k)\frac{\partial}{\partial w_{jk}} \sum_{j=0}^{p} z_j w_{jk}$$
$$= -(t_k - y_k)f'(y\_in_k)z_j = -\delta_k z_j$$

With $\delta_k = (t_k - y_k)f'(y\_in_k)$



$z_j = f(z\_in_j), z_{in_j} = \sum_{i=0}^{n} x_i v_{ij}, x_0 = 1, j = 1 \dots p$

$y_k = f(y\_in_k), y_{in_k} = \sum_{j=0}^{p} z_j w_{jk}, z_0 = 1, k = 1 \dots m$

Use of Gradient Descent Minimization

$$\Delta w_{jk} = -\alpha\frac{\partial E}{\partial w_{jk}} = \alpha\delta_k z_j$$

$$\frac{\partial E}{\partial v_{ij}} = \frac{\partial}{\partial v_{ij}} 0.5 \sum_k (t_k - y_k)^2$$

$$= 0.5 \sum_k \frac{\partial}{\partial v_{ij}} (t_k - y_k)^2$$

Change in v_{ij} affects all $Y_{1..m}$

$$= \sum_k (t_k - y_k) \frac{\partial}{\partial v_{ij}} (-y_k)$$

$$= -\sum_k (t_k - y_k) \frac{\partial}{\partial v_{ij}} f(y_{in_k})$$

$$= -\sum_k (t_k - y_k) f'(y_{in_k}) \frac{\partial}{\partial v_{ij}} y_{in_k}$$

$$= -\sum_k \delta_k \frac{\partial}{\partial v_{ij}} \sum_{j=0}^{p} z_j w_{jk}$$

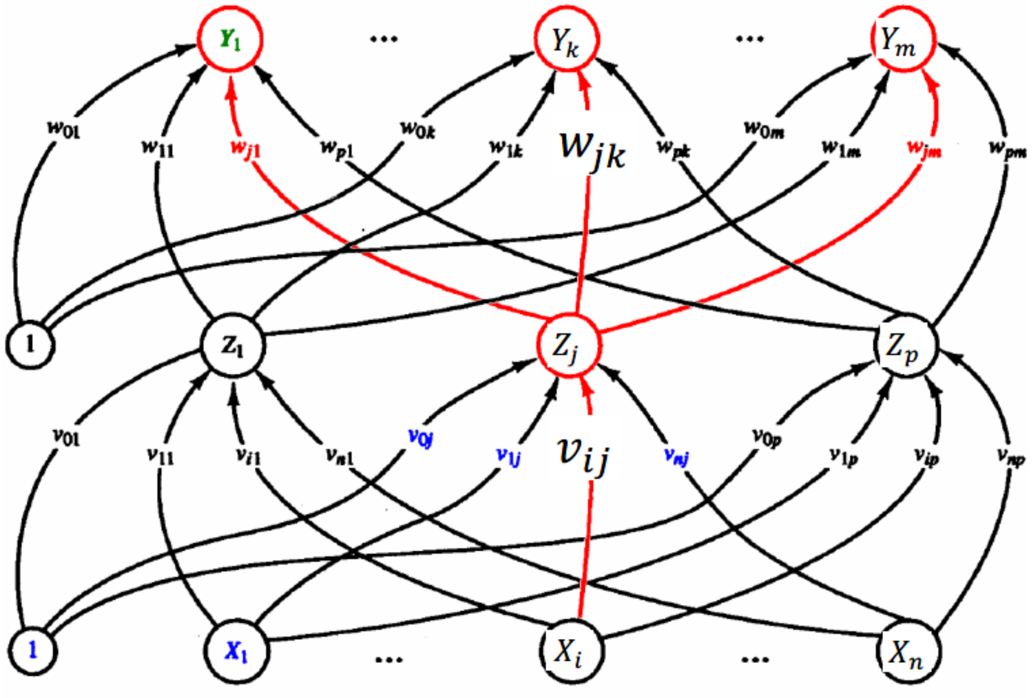
Change in v_{ij} affects only z_j

$$= -\sum_k \delta_k \frac{\partial}{\partial v_{ij}} z_j w_{jk} = -\sum_k \delta_k w_{jk} \frac{\partial}{\partial v_{ij}} f\left(z_{in_j}\right)$$

$$= -\sum_k \delta_k w_{jk} f'\left(z_{in_j}\right) \frac{\partial}{\partial v_{ij}} z_{in_j}$$

$$= -\sum_k \delta_k w_{jk} f'\left(z_{in_j}\right) \frac{\partial}{\partial v_{ij}} \sum_{i=0}^{n} x_i v_{ij}$$

$$= -\sum_k \delta_k w_{jk} f'\left(z_{in_j}\right) x_i = -\delta_j x_i$$



$z_j = f(z\_in_j), z_{in_j} = \sum_{i=0}^{n} x_i v_{ij}, x_0 = 1, j = 1 \dots p$

$y_k = f(y\_in_k), y_{in_k} = \sum_{j=0}^{p} z_j w_{jk}, z_0 = 1, k = 1 \dots m$

With $\delta_k = (t_k - y_k) f'(y\_in_k)$

Use of Gradient Descent Minimization

$$\Delta v_{ij} = -\alpha \frac{\partial E}{\partial v_{ij}} = \alpha \delta_j x_i$$

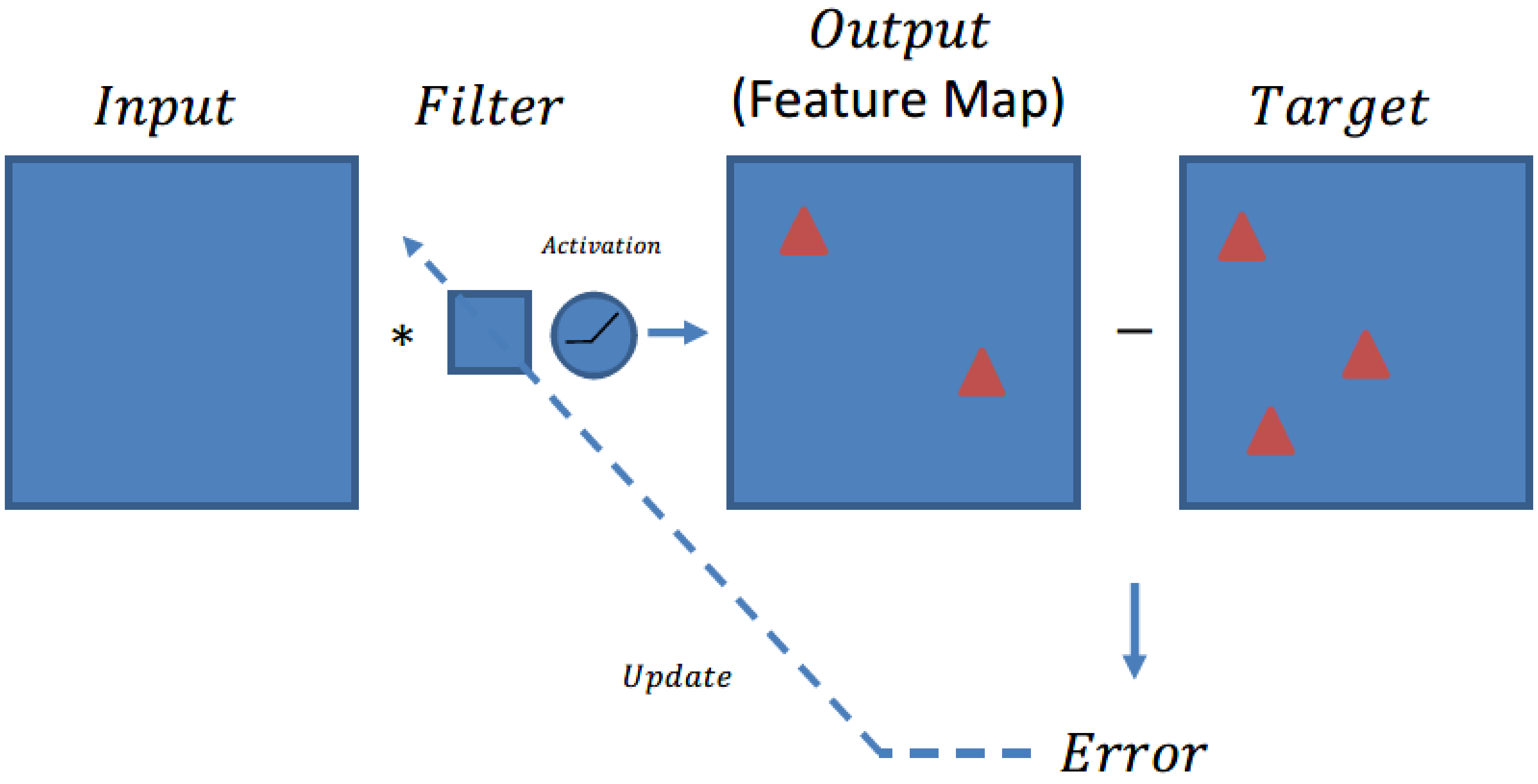
Understanding NNs

- Understanding the role of
  - Gradients
  - Inputs
  - Activation functions (esp output layer activation functions such as softmax)
  - Learning rate
  - Number of layers
  - Number of neurons
  - Los

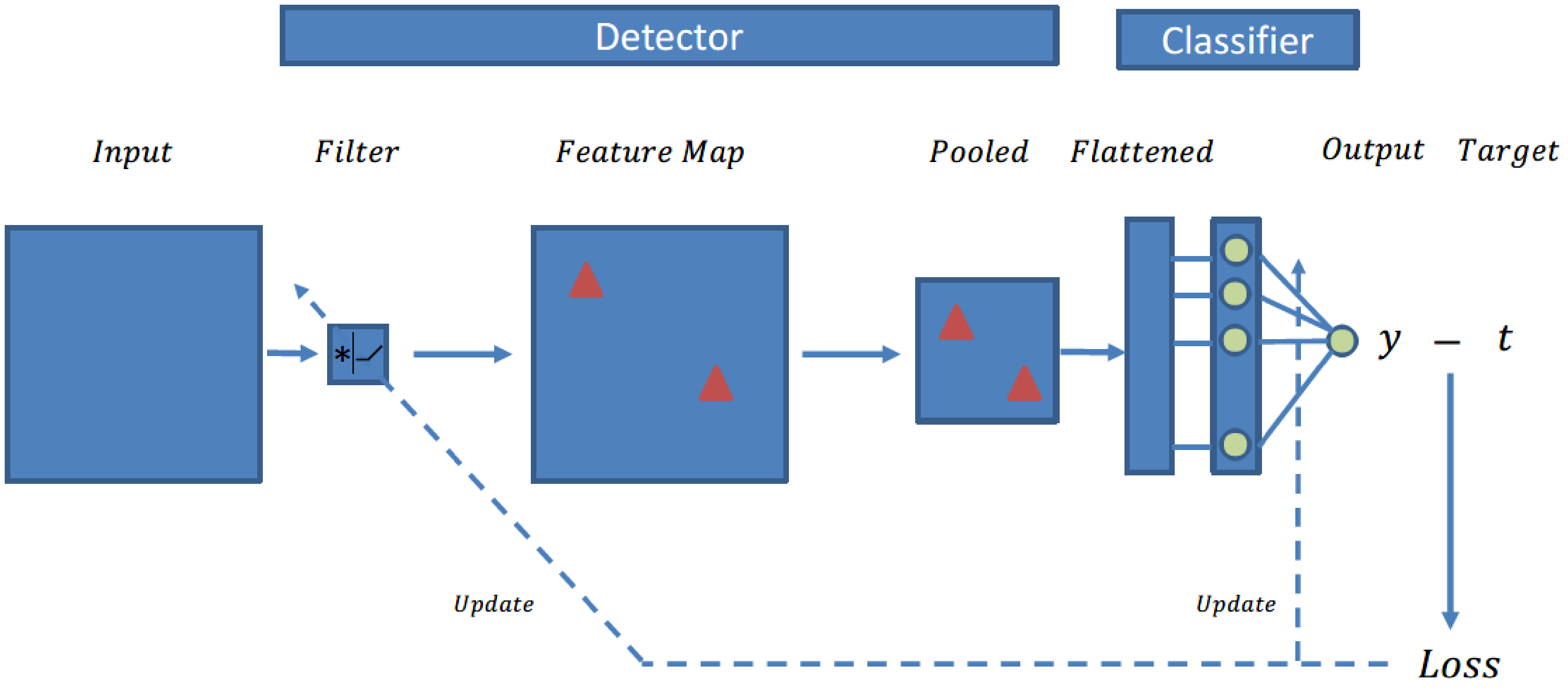
$$\Delta v_{ij} = \alpha x_i f'(\boldsymbol{v}_j^T \boldsymbol{x}) \sum_{k=1}^{m} w_{jk} \left(t_k - f\left(\sum_{j=0}^{p} w_{jk} f(\boldsymbol{v}_j^T \boldsymbol{x}) \right) \right) f'\left(\sum_{j=0}^{p} w_{jk} f(\boldsymbol{v}_j^T \boldsymbol{x}) \right)$$

Convolution Neural Networks

- Filter based object detection



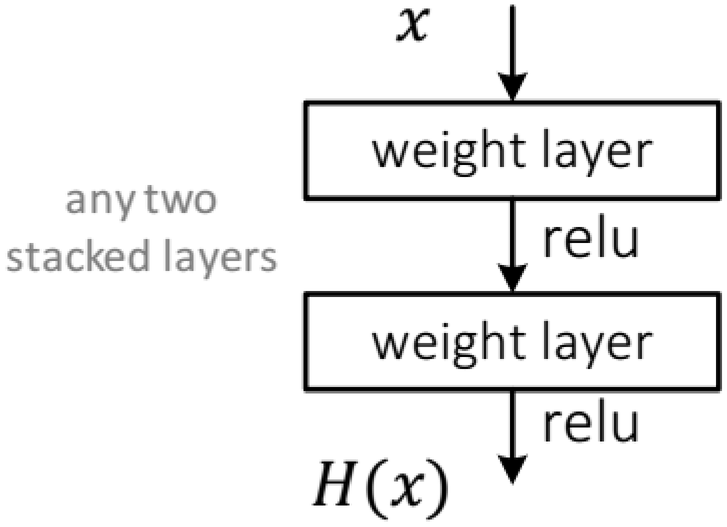
CNNs



Understanding why CNNs work!

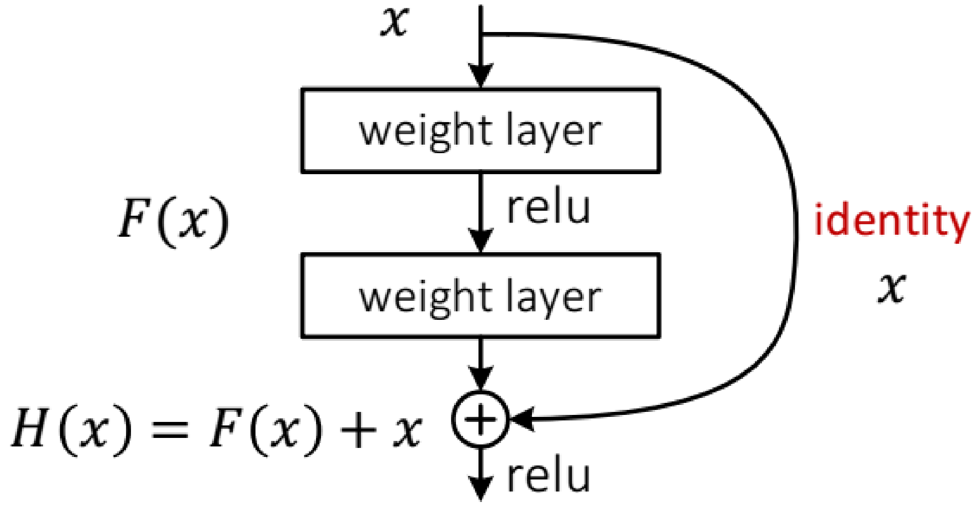
Residual Learning: skip connections

Plain Network



H(x) is any desired mapping
Hope the 2 weight layers fit *H(x)*

Residual Network



H(x) is any desired mapping
Hope the 2 weight layers fit *F(x)*

The network learns fluctuations *F(x)=H(x)-x*
Easier!

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

Conditional GAN

Training data: (c, x), (condition, desired output), e.g., (text, image)

Condition c

Prior distribution z

Generator

x

Learn to approximate $p(x|c)$

For Generator:
It wants the discriminator to classify $(c, G(c))$ as positive

Condition c

Discriminator

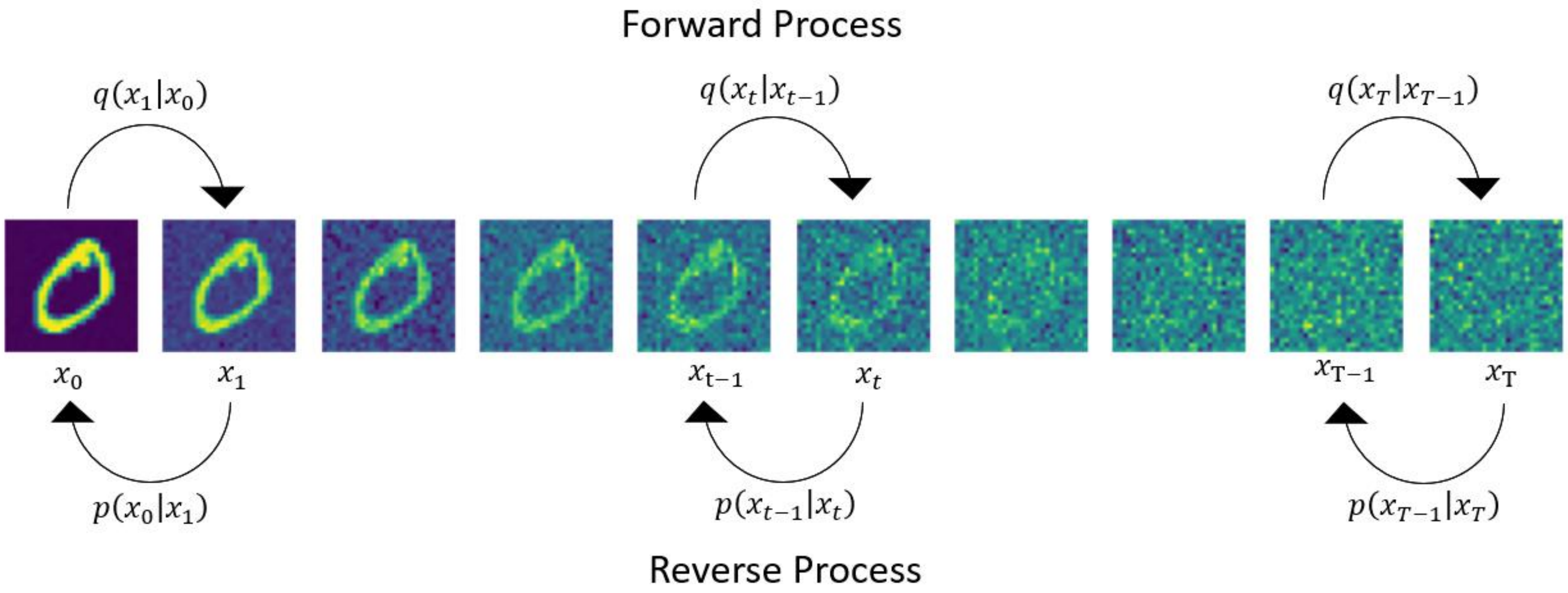
scalar

For Discriminator:
Positive example: (c, x), e.g., the original (text, image) pair

Negative examples: $(c, G(c))$, e.g., (text, generated image) pair

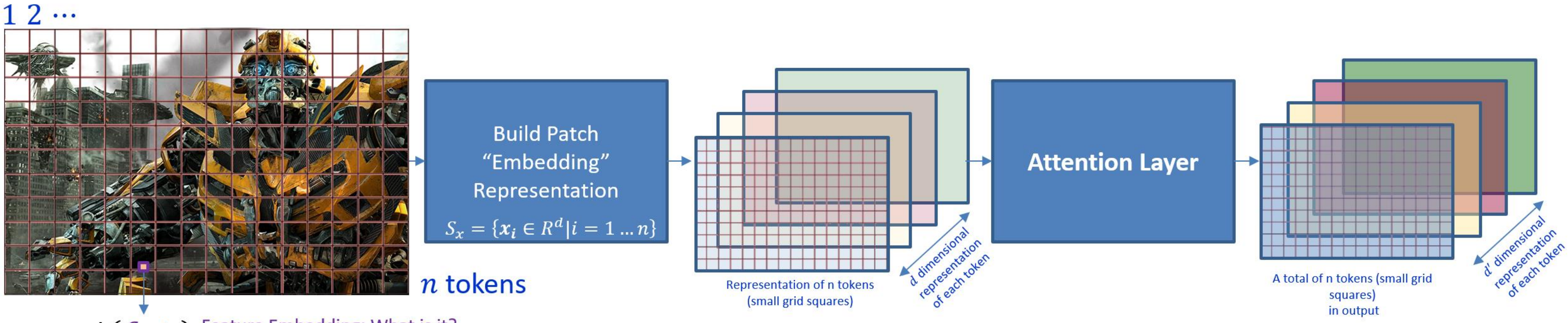
(c', x), e.g., (arbitrary text, original image) pair

Other Topics: Diffusion Models (Optional)



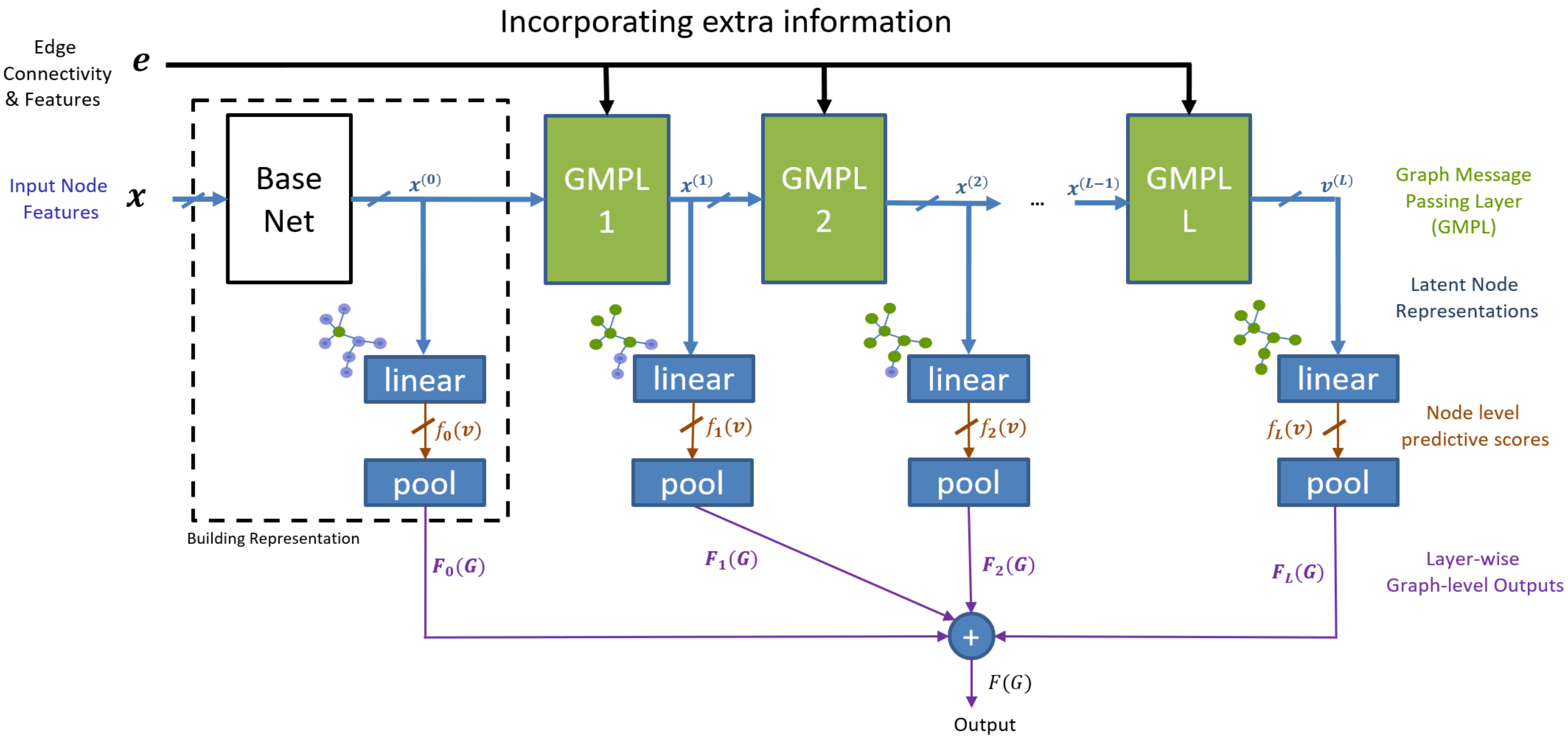
https://github.com/wgrgwrght/Simple-Diffusion/blob/main/SimpleDiffusion.ipynb

Other Topics: Transformers



$\boldsymbol{x_i} \equiv \phi(\boldsymbol{f_i}, \boldsymbol{t_i})$ Feature Embedding: What is it?
Positional Embedding: Where is it?

Message Passing Based Graph Neural Networks



Code: https://tia-toolbox.readthedocs.io/en/latest/_notebooks/jnb/inference-pipelines/slide-graph.html

Fayyaz Minhas, Whole Slide Images Are Graphs, 2020. https://www.youtube.com/watch?v=Of1u0i7roS0.

Exam Philosophy and Types of Questions

- Testing the student's ability to generalize and cross-connect
- Types of questions
  - Solution
    - Solve or Calculate
  - Conceptual
    - Why does …
  - Book work
    - What is ..
  - Application
    - How to ..

Exam Structure

- Attempt four out of 5 questions

- Past papers (No solutions)
  - https://warwick.ac.uk/services/exampapers?q=cs909&department=&year=
  - https://warwick.ac.uk/services/exampapers?q=cs429&department=&year=