More Multiclass Classification and Towards Generative Models

Alexander G. Ororbia II

Introduction to Machine Learning

CSCI-335

3/25/2026

Special Thanks: Sargur N. Srihari, Andrew Ng, Carlos Guestrin

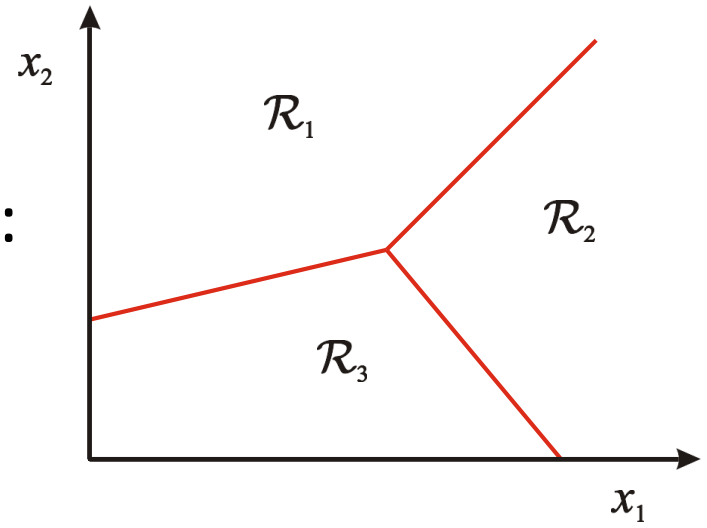
Discriminative Approach

Ex: Logistic regression, multinoulli regression

Estimate $P(Y|X)$ directly

(Or a discriminant function: e.g., a linear classifier or support vector machine)

Prediction (mode):
$\hat{y} = P(Y = y|X = x)$



Generative Approach

Ex: Naïve Bayes

Estimate $P(Y)$ and $P(X|Y)$

Prediction (mode):
$\hat{y} = \text{argmax}_y \, P(Y = y)P(X = x|Y = y)$

Discriminative Approach

Ex: Logistic regression, multinoulli regression

Estimate $P(Y|X)$ directly

(Or a discriminant function: e.g., a linear classifier or support vector machine)

Prediction (mode):
$\hat{y} = P(Y = y|X = x)$

Generative Approach

Ex: Naïve Bayes

Estimate $P(Y)$ and $P(X|Y)$

This is coming up soon…

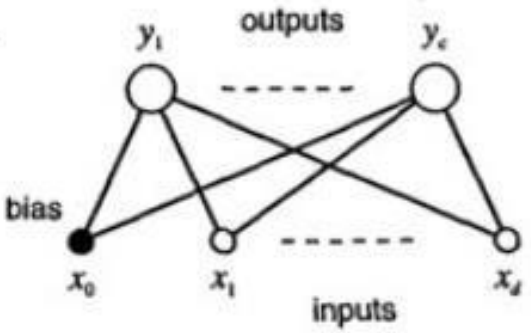
Prediction (mode):
$\hat{y} = \text{argmax}_y \, P(Y = y)P(X = x|Y = y)$

Multinoulli (Softmax) Regression

$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \sum_{l=0}^{m} w_l x_l = \mathbf{w}^T \mathbf{x}.$$

θ_z^T

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\left\{y^{(i)} = j\right\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right]$$



Also known as:

Categorical regression

Maximum entropy (classifier)

```
def softmax(X):

    exps = np.exp(X)

    return exps / np.sum(exps)
```

Note: Full generalization of logistic regression to handle Categorical distributions (over discrete categories/classes)

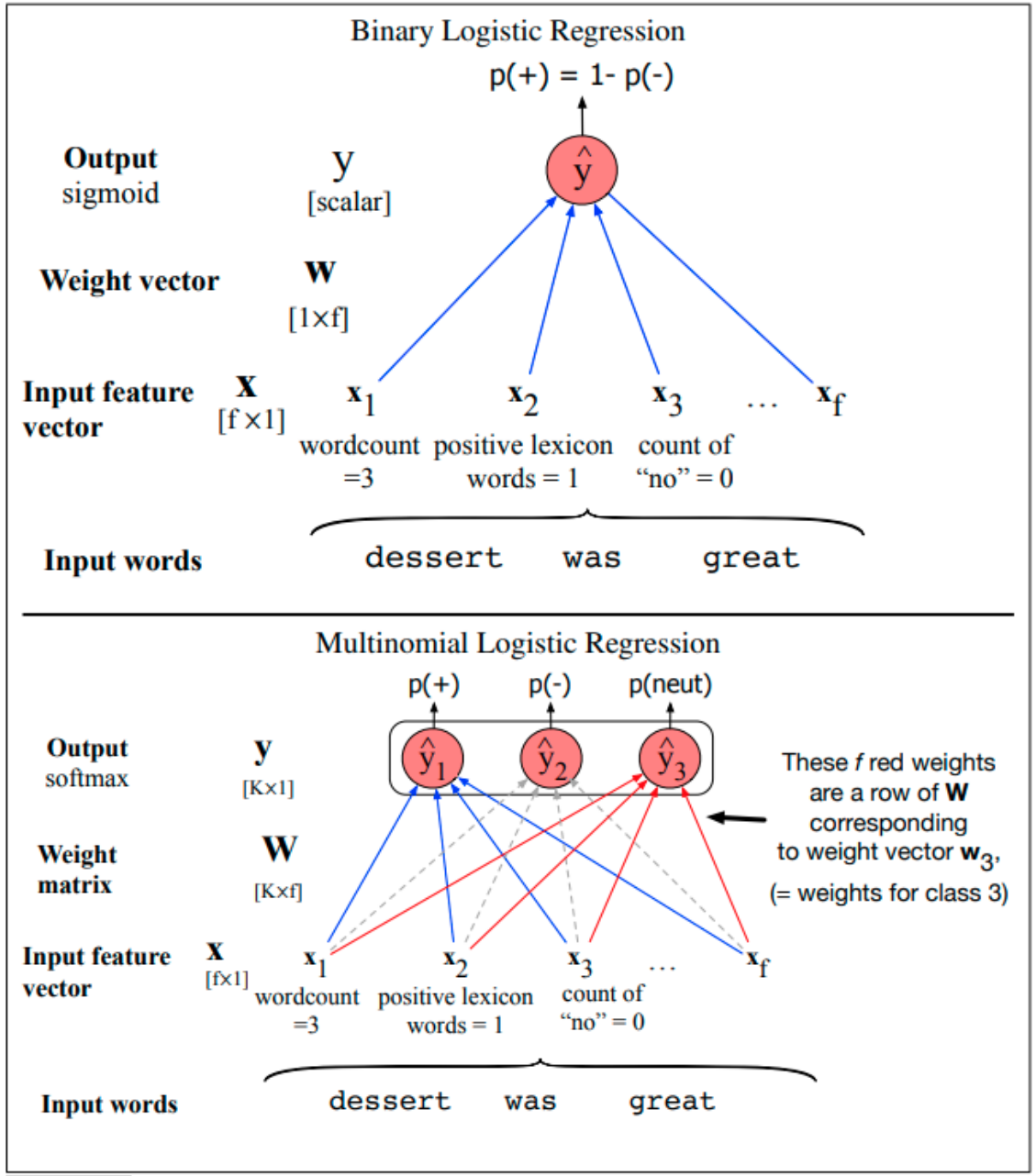


Figure 5.3 Binary versus multinomial logistic regression. Binary logistic regression uses a single weight vector $\mathbf{w}$, and has a scalar output $\hat{y}$. In multinomial logistic regression we have K separate weight vectors corresponding to the K classes, all packed into a single weight matrix $\mathbf{W}$, and a vector output $\hat{\mathbf{y}}$.

(Daniel Jurafsky & James H. Martin 2021)

Multinoulli: Connection to Logistic Regression

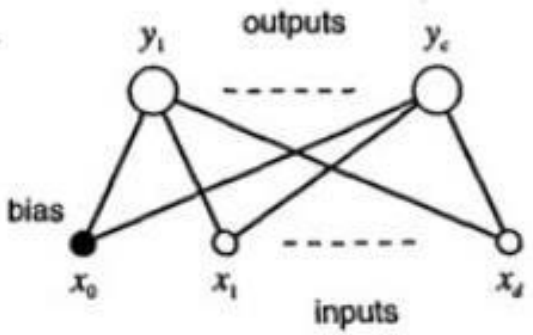
1) Derivation of 2-Class multinoulli regressor in terms of logistic regression!

2) Derivation of the multinoulli (log) likelihood in terms of Bernoulli (log) likelihood!

Multinoulli (Softmax) Regression

$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \sum_{l=0}^{m} w_l x_l = \mathbf{w}^T \mathbf{x}.$$

θ_j^T



$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\left\{y^{(i)} = j\right\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right]$$

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}(1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) + y^{(i)} \log h_\theta(x^{(i)})\right]$$

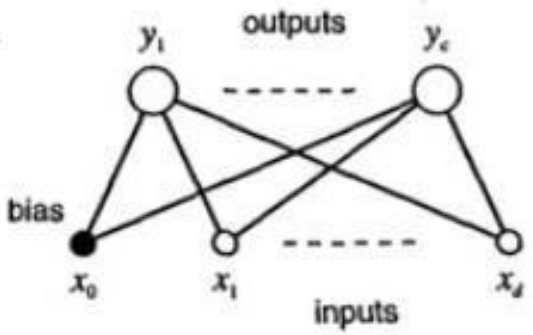
$$= -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=0}^{1} 1\left\{y^{(i)} = j\right\} \log p(y^{(i)} = j | x^{(i)}; \theta)\right]$$

$$\nabla_{\theta_j} J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[x^{(i)}\left(1\{y^{(i)} = j\} - p(y^{(i)} = j | x^{(i)}; \theta)\right)\right]$$

Multinoulli (Softmax) Regression

$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \sum_{l=0}^{m} w_l x_l = \mathbf{w}^T \mathbf{x}.$$

θ_z^T



$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\left\{y^{(i)} = j\right\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right]$$

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}(1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) + y^{(i)} \log h_\theta(x^{(i)})\right]$$

$$= -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=0}^{1} 1\left\{y^{(i)} = j\right\} \log p(y^{(i)} = j | x^{(i)}; \theta)\right]$$

$p(y^{(i)} = j | x^{(i)}; \theta)$

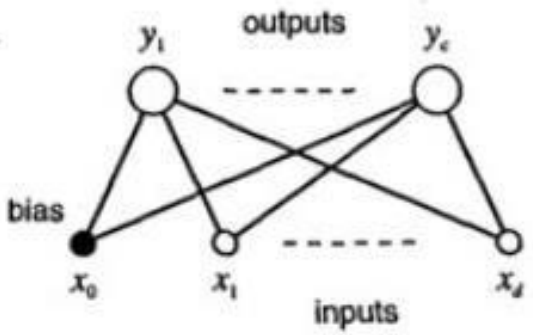
Again, the connection to logistic regression…

$$\nabla_{\theta_j} J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[x^{(i)}\left(1\{y^{(i)} = j\} - p(y^{(i)} = j | x^{(i)}; \theta)\right)\right]$$

Multinoulli (Softmax) Regression

$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \sum_{l=0}^{m} w_l x_l = \mathbf{w}^T \mathbf{x}.$$

θ_j^T



$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\left\{y^{(i)} = j\right\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right]$$

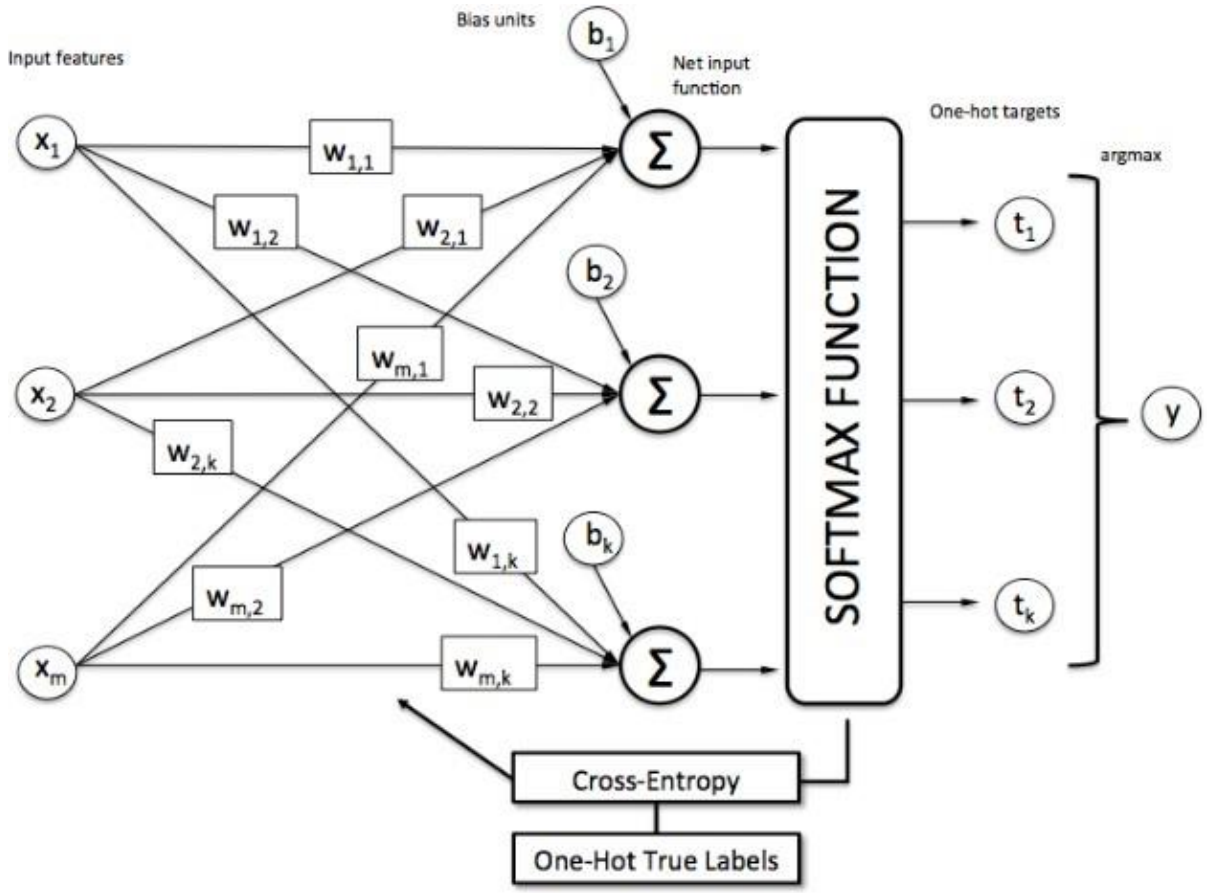
$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) + y^{(i)} \log h_\theta(x^{(i)})\right]$$

$$= -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=0}^{1} 1\left\{y^{(i)} = j\right\} \log p(y^{(i)} = j | x^{(i)}; \theta)\right]$$

$$\nabla_{\theta_j} J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[x^{(i)}\left(1\{y^{(i)} = j\} - p(y^{(i)} = j | x^{(i)}; \theta)\right)\right]$$

Derivation is left as an exercise for you ;-)

Multinoulli Regressor: Architecture



Multinoulli Regressor: Architecture

We will learn parameters by minimizing the negative log likelihood of our model's predictive posterior. If we say that $\mathbf{p}$ is our model's vector of normalized output probabilities, we define the negative log loss (cost) as follows:

$$\mathbf{p}_k = \frac{e^{\mathbf{f}_k}}{\sum_j e^{\mathbf{f}_j}}, \qquad \mathcal{J} = -\frac{1}{N}\sum_i \Big(\log(\mathbf{p}_{y_i})\Big) + \frac{\lambda}{2}\sum_d \sum_k (W_{d,k})^2$$

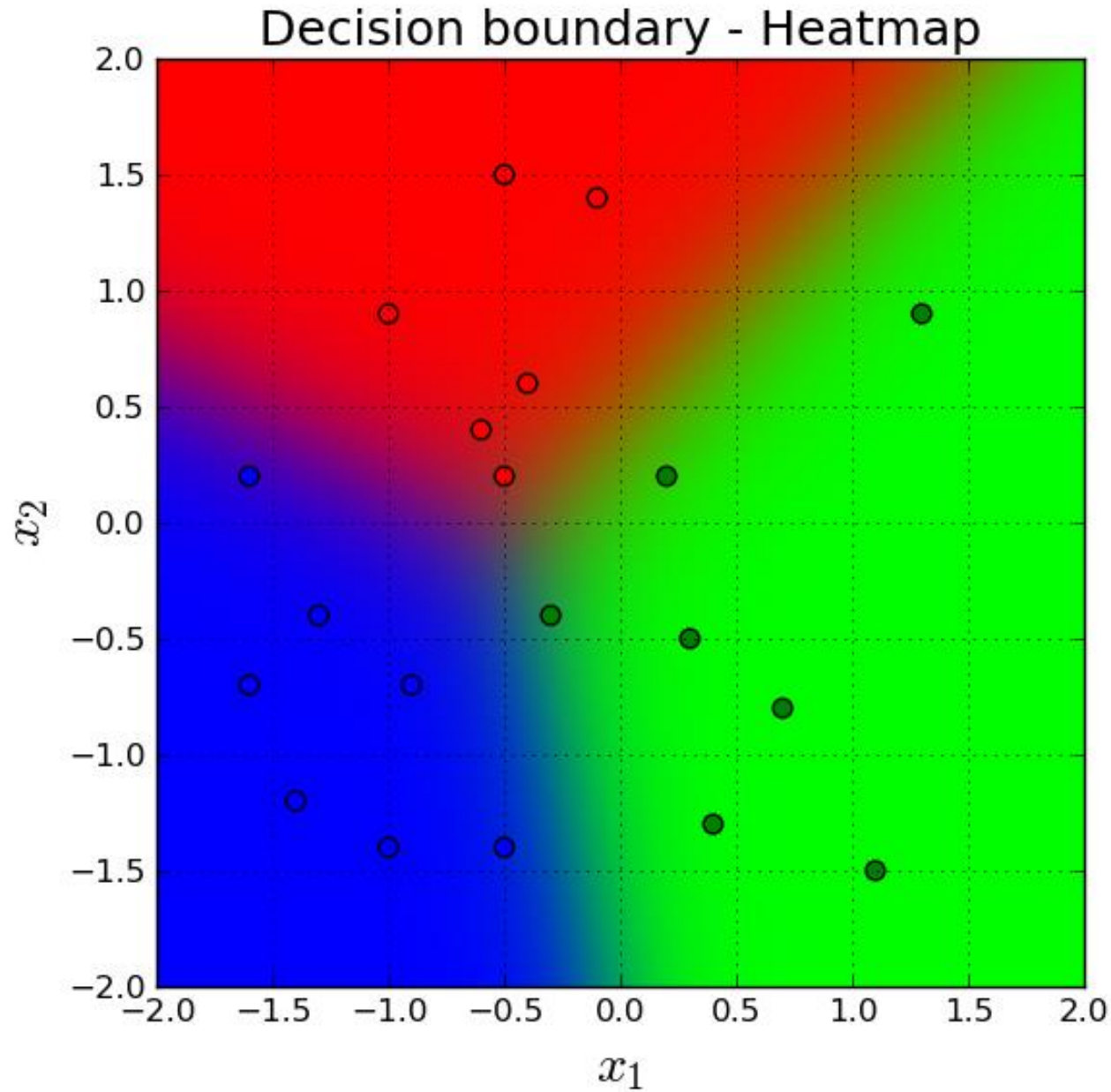
```
Inputs X:
 [[ 0.1  0.5]
 [ 1.1  2.3]
 [-1.1 -2.3]
 [-1.5 -2.5]]

[[ 1.  0.  0.]
 [ 0.  1.  0.]
 [ 0.  0.  1.]
 [ 0.  0.  1.]]
```

$$\frac{\partial \mathcal{J}_i}{\partial \mathbf{f}_k} = \mathbf{p}_k - \mathbf{1}_{y_i=k}$$

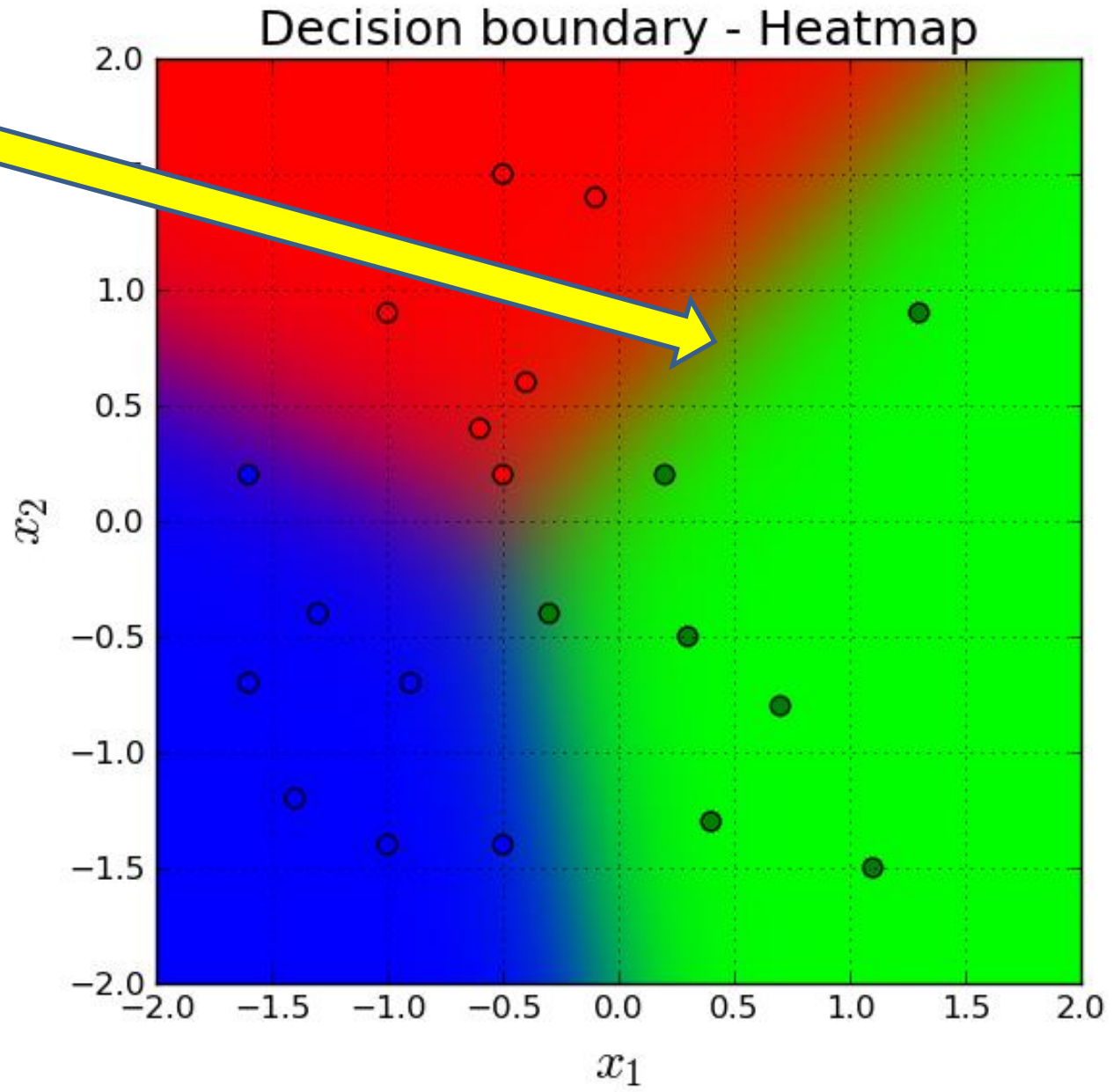
Recall: what is this term known as?

$$\frac{\partial \mathcal{J}}{\partial W} = \mathbf{X}^T * \left(\frac{1}{N}\frac{\partial \mathcal{J}}{\partial \mathbf{f}_k}\right) + \lambda(W) = \mathbf{X}^T * \left(\frac{1}{N}(\mathbf{p}_k - \mathbf{1}_{y_i=k})\right) + \lambda(W)$$

$$\frac{\partial \mathcal{J}}{\partial b} = \sum_{n=0}^{N} \frac{1}{N}\frac{\partial \mathcal{J}}{\partial \mathbf{f}_k} = \sum_{n=0}^{N} \frac{1}{N}(\mathbf{p}_k - \mathbf{1}_{y_i=k})$$


Decision boundary - Heatmap
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
x_2
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x_1

The boundaries



Notice that the data is linearly separable!

Could We Do Even Better?

- Yes, if we revisit artificial features / basis functions…
  - Generalize these to the notion of infinite-dimensional feature spaces (leading to the “kernel’ trick)
  - Incorporate the concept of “support vectors” and max-margin optimization
  - We get ***support vector machines (SVMs; not treated in this course)***
- Probabilistic graphical models (PGMs)
  - We introduce domain knowledge to craft graphs/relationships of variables with probability arcs/tables
- Deep learning
  - Through artificial neural networks or chains of nonlinear processing
  - We will get to this later

A Cautionary Example

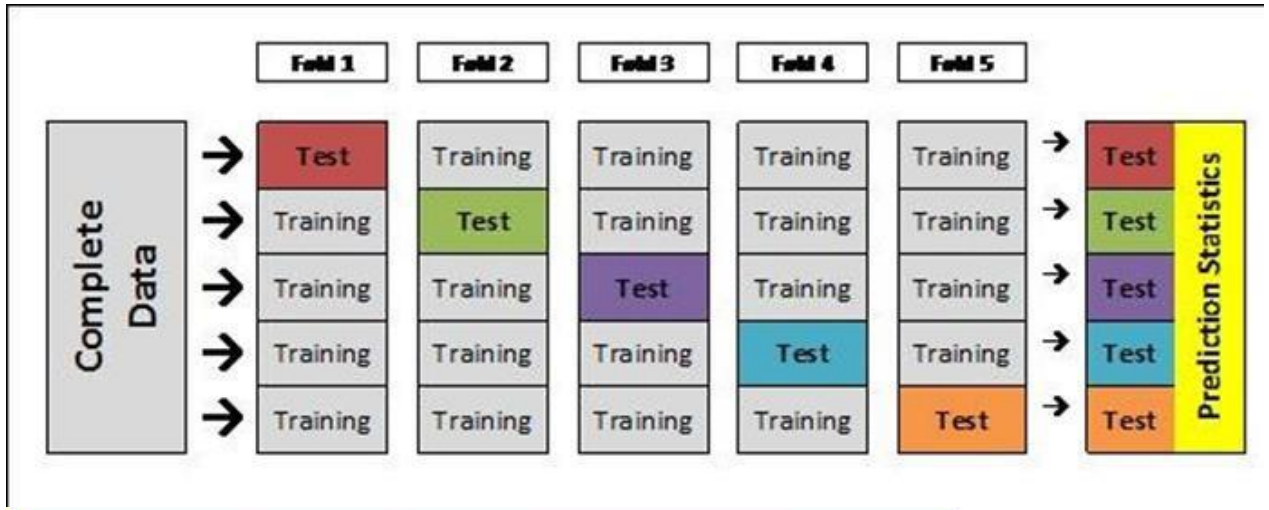
Image classification of tanks. Autofire when an enemy tank is spotted.
Input data: Photos of own and enemy tanks.
Worked really good with the training set used.
In reality, it failed completely. **Why?**

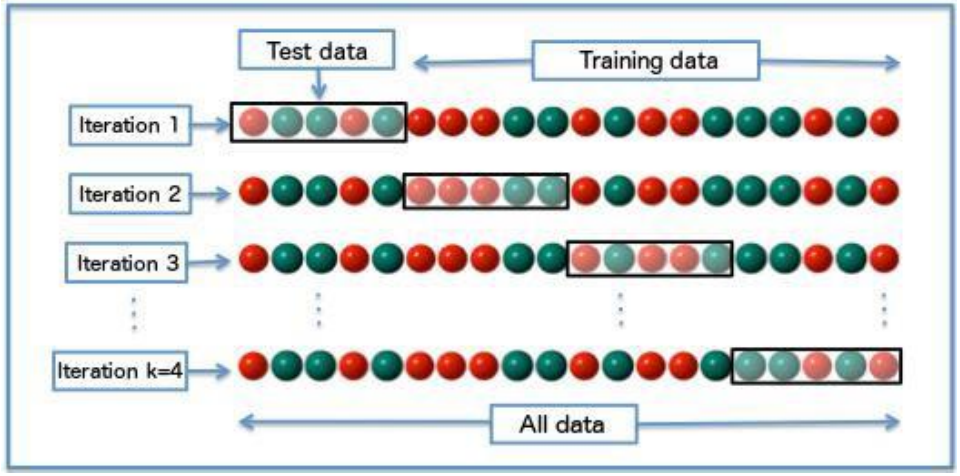
A Cautionary Example

Image classification of tanks. Autofire when an enemy tank is spotted.
Input data: Photos of own and enemy tanks.
Worked really good with the training set used.
In reality, it failed completely.

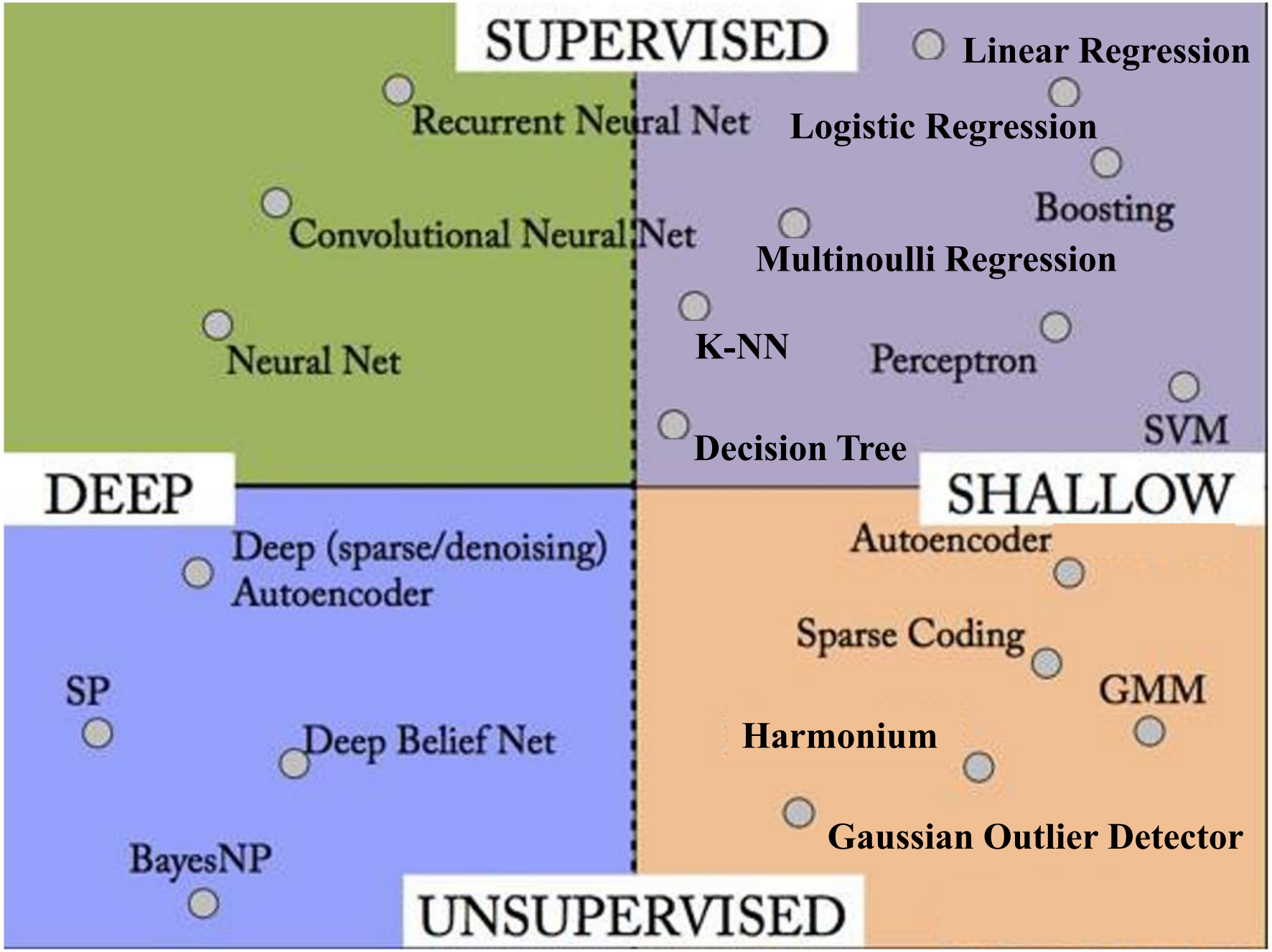
Reason: All enemy tank photos taken in the morning. All tanks in data shown at dawn
The classifier could NOT recognize dusk from dawn!!!!

Cross-Validation: Measuring Generalization





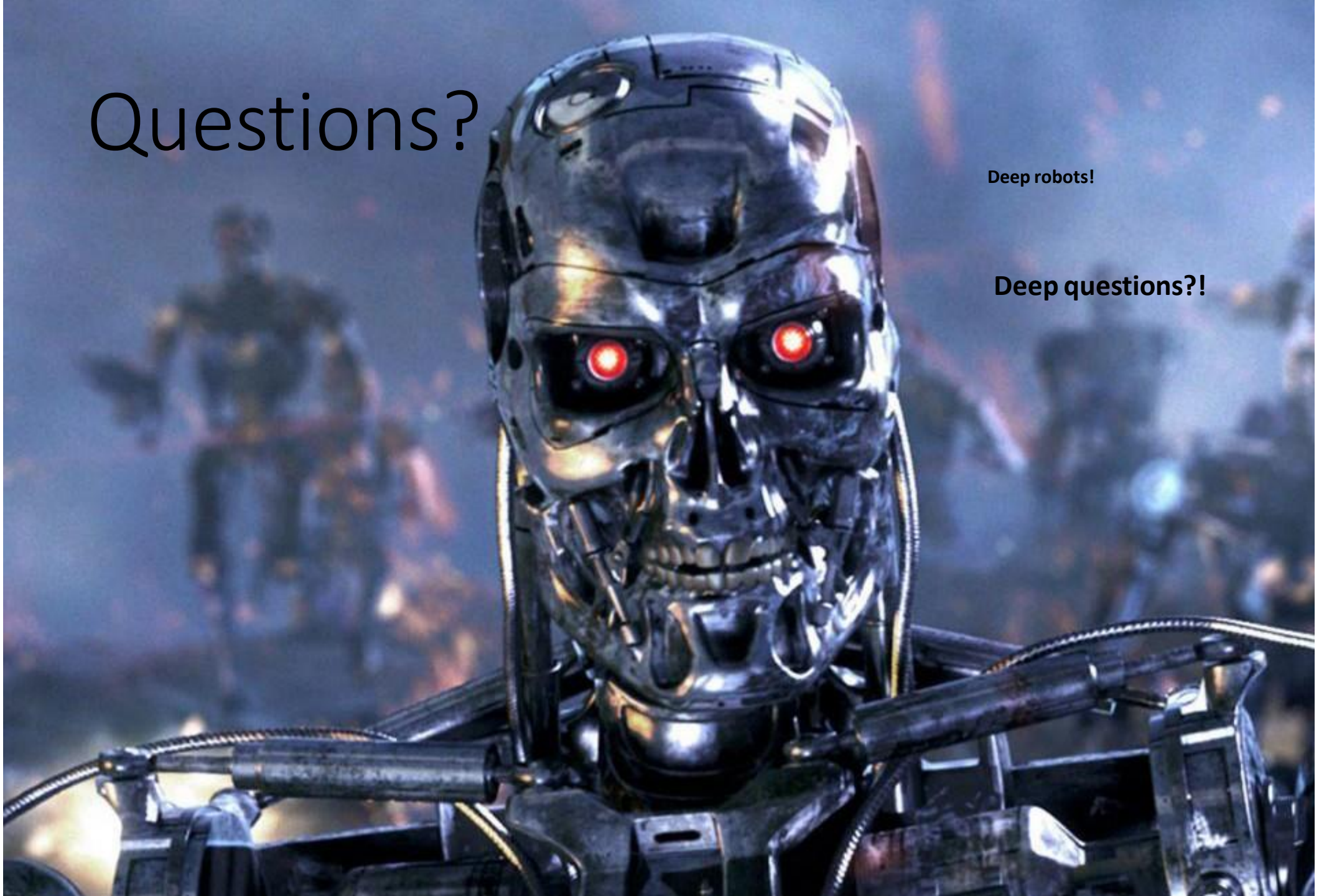
Good for when dataset is small! Average over trials/folds to get a biased estimator of generalization error.


SUPERVISED
Linear Regression
Recurrent Neural Net
Logistic Regression
Boosting
Convolutional Neural Net
Multinoulli Regression
K-NN
Neural Net
Perceptron
SVM
Decision Tree
DEEP
SHALLOW
Deep (sparse/denoising) Autoencoder
Autoencoder
Sparse Coding
SP
GMM
Deep Belief Net
Harmonium
Gaussian Outlier Detector
BayesNP
UNSUPERVISED

Getting Back to Bayes Thinking

Bayes Theorem Revisited!

Questions?



Deep robots!

Deep questions?!