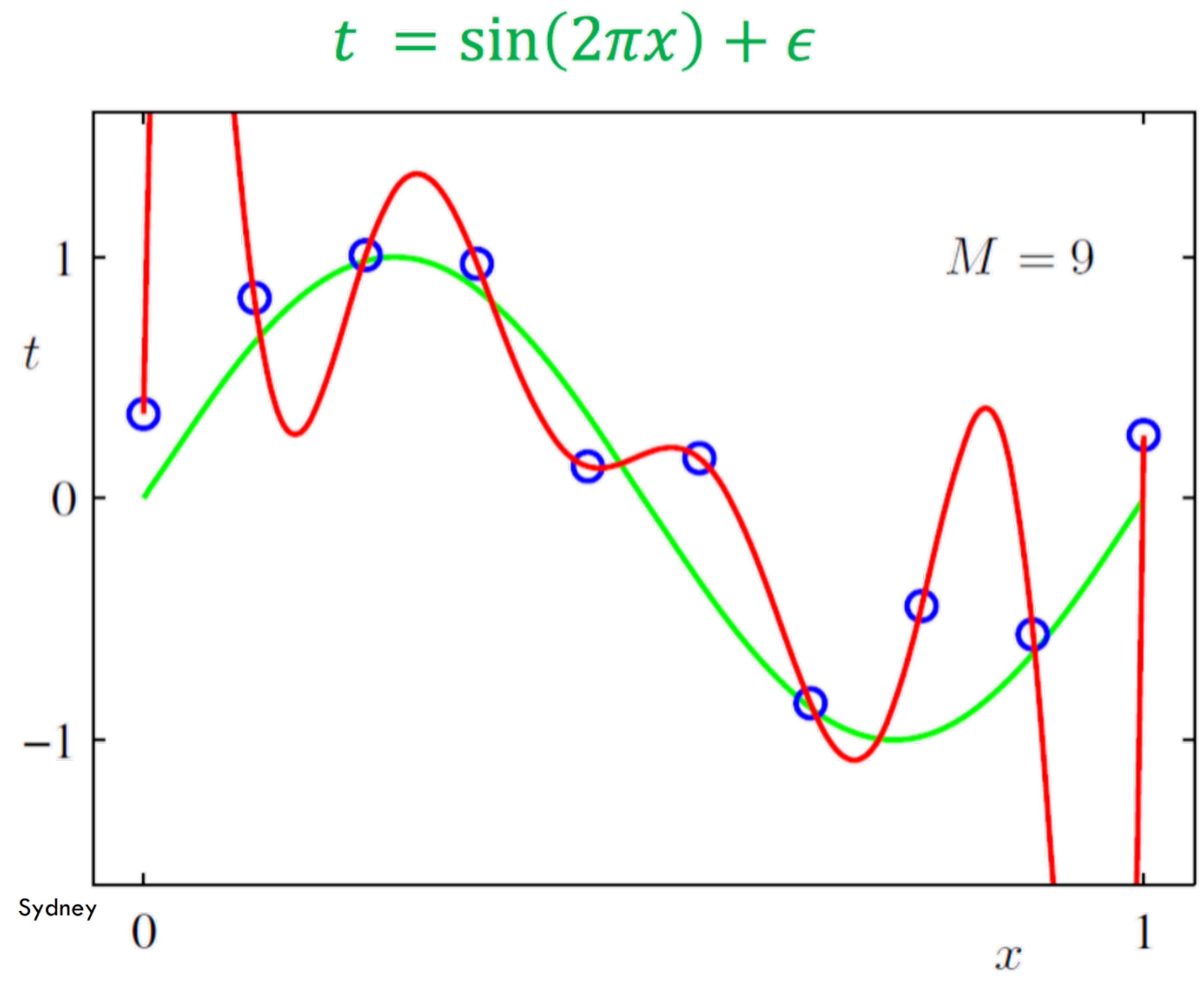
Week2 CNN Architectures

Tutor:
Email:
Tutorial:
Code: https://github.com/Jinxu-Lin/COMP5329

Overfitting

Overfitting

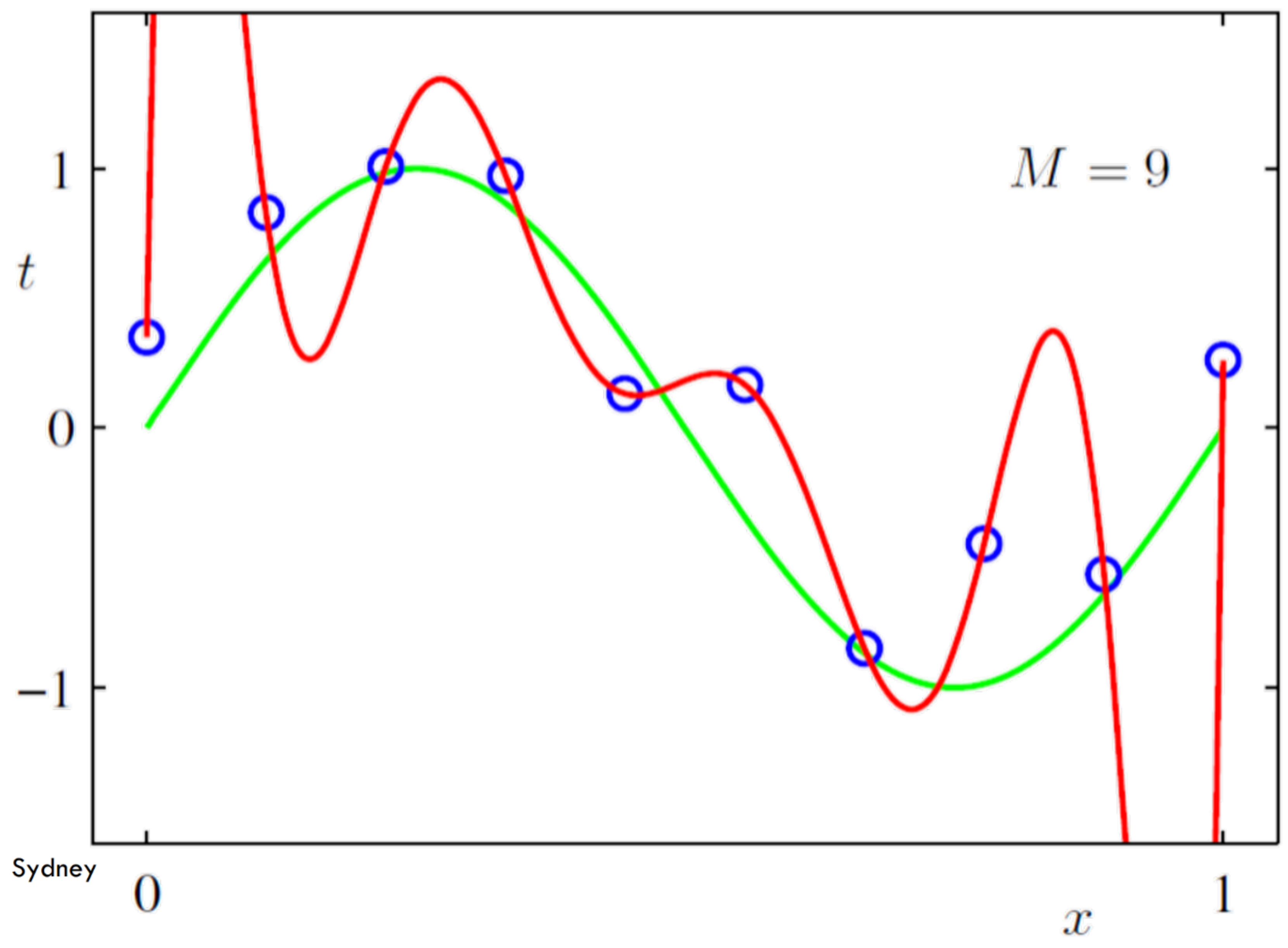
- Reasons:
  - Training data is too small to represent the whole dataset
  - Large amount of irrelavent information
  - Model complexity is too high
  - The model is trained for too long



Overfitting

- Solutions:
  - Increase training data (collect more data)
  - Data cleaning jobs
  - Reduce model complexity
  - Regularization (reduce the effect of model complexity)
  - Stop training early

$t = \sin(2\pi x) + \epsilon$



Techniques

L1/L2 Regularization

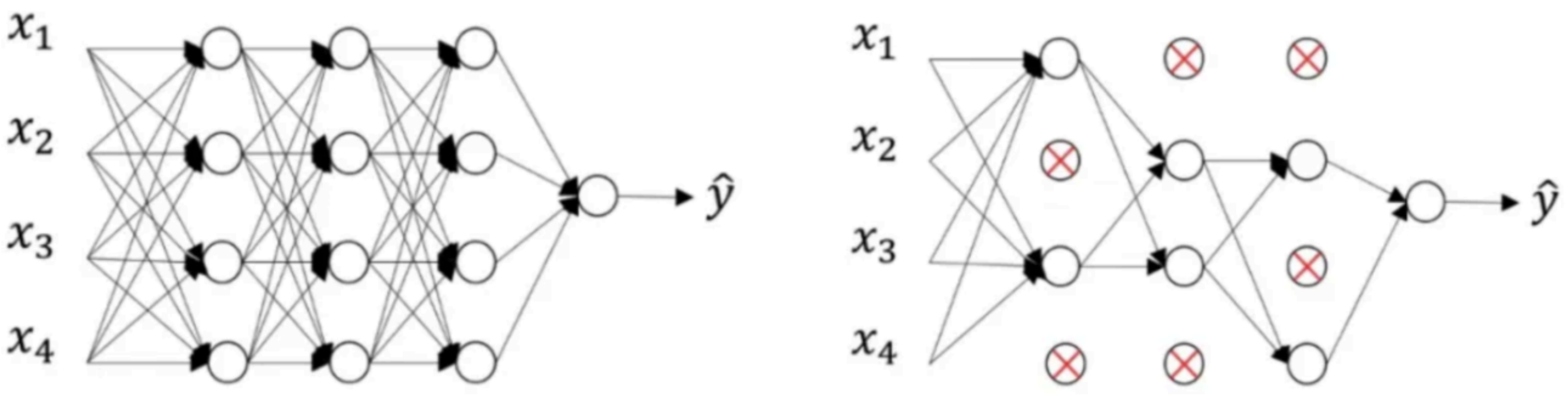
- Given loss $\hat{L}(\theta)$, Trainin objective:
- $\min_{\theta} \hat{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} l(\theta, x_i, y_i)$, s.t. $R(\theta) \leq r$, where l is the loss for each instance.
- L2 regularization: $R(\theta) = ||\theta||^2$
- L1 regularization: $R(\theta) = |\theta|$
- Bayesian regularization can actually be transformed to L2/L1 regularization. Refer to this link to see: https://towardsdatascience.com/a-bayesian-take-on-model-regularization-9356116b6457

Soft Regularization

- Use regularization as penalty, we have the new objective
- $\min_{\theta} = \frac{1}{n} \sum_{i=1}^{n} l(\theta, x_i, y_i) + \lambda * R(\theta)$
- However, the calculation of $R(\theta)$ is still very complicated in case of L2 regularization, we adopt the weight decay:
- $\theta = \theta - w_d * \theta$
- Normally, weight decay $w_d \to 0$. We can change this value during training

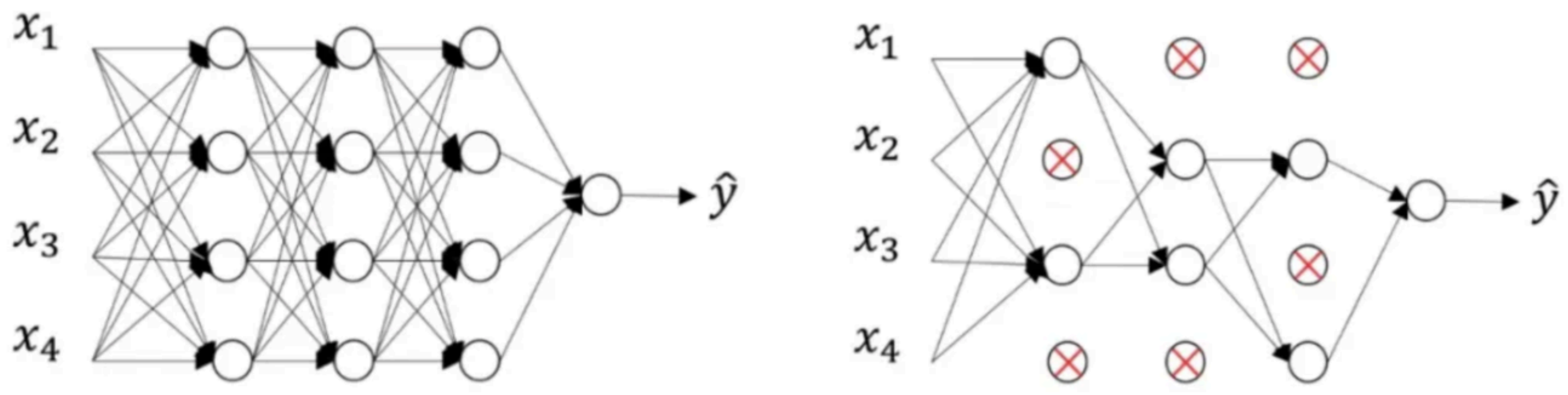
Dropout

- The basic idea is to add some random/surprising factors into the training process (only training/ deactivated during testing)
    - $o = f(W * x + b)$
- $o *$ mask, half of the mask to be 0.0, half of the mask to be 1



Dropout

- During the test time, you need to scale down or scale up in the training
- Behaviors of dropout during training and testing phase is different:
  - Magnitude of values
  - don't filter out values during testing.



Batch Normalization

- The previous regularization only helps to shrink/cut-off redundant information/ provide surprising factors to the training process.
- We need a more essential method to tackle the difference between training data and testing data.

Batch Normalization

- Training
    - Update mean value $\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$
    - Update variance value $\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2$
    - Update running mean value $\mu_x \leftarrow E[\mu_B]$ (implementation)
    - Update running var value $\sigma_x^2 \leftarrow E[\sigma_B^2]$ (implementation)
    - Calculate Normalized value $\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$
    - Update new neural value $x_i \leftarrow \gamma \hat{x}_i + \beta$

Batch Normalization

- Testing
  - Calculate Normalized value $\hat{x}_i \leftarrow \frac{x_i - \mu_x}{\sqrt{\sigma_x^2 + \epsilon}}$
  - Update new neural value $x_i \leftarrow \gamma \hat{x}_i + \beta$
- with γ, β are learned values, μ_B, σ_B^2 are accumulated during training.

Exam-Style Questions

Code

Code

- ./materials/Week7_Regularization/Week7_Regularization.ipynb
- ./ResNet/Models/batch_normalization.py
- ./ResNet/Models/dropout.py
- ./ResNet/Models/resnet.py
- ./ResNet/Optimizers/sgd.py: line26-28