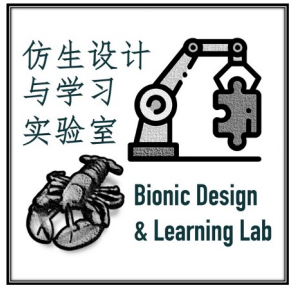
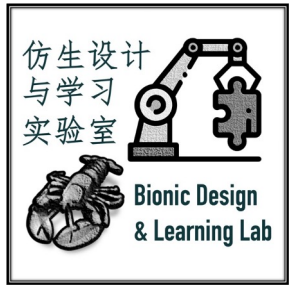
Lecture 08
Network Tuning I



AncoraSIR.com

SUSTech
Southern University
of Science and Technology

Regularization for Deep Learning



AncoraSIR.com

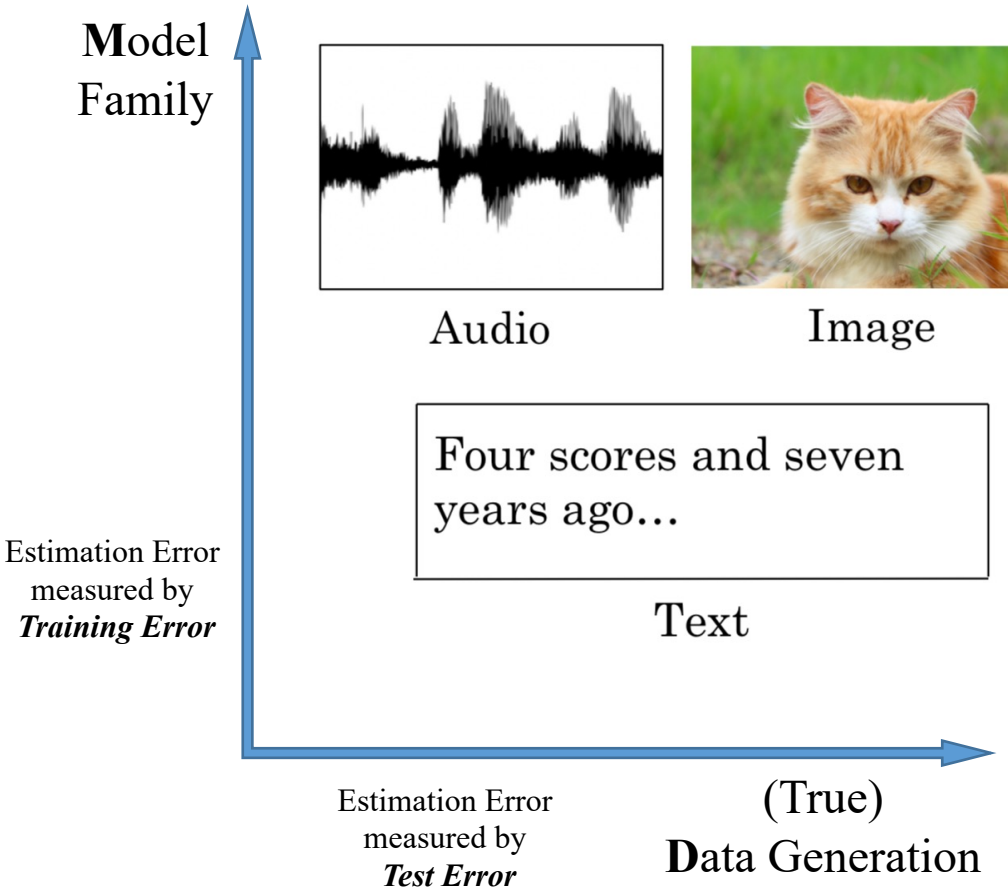
SUSTech
Southern University
of Science and Technology

Regularization in General

Any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error

Regularization of a mismatch

- **Data generation**: almost never have access to the true data generation process
  - *Measured by **test error**, to be reduced ultimately when dealing with new inputs*
- **Model representation**: not sure if our model family covers the data generation or not
  - *Measured by **training error**, can be minimized by exploiting the data, not the purpose*
- The problem can be extremely complicated
  - Image, audio, text, etc.
  - **Reduce the test error at the expense of increasing the training error**



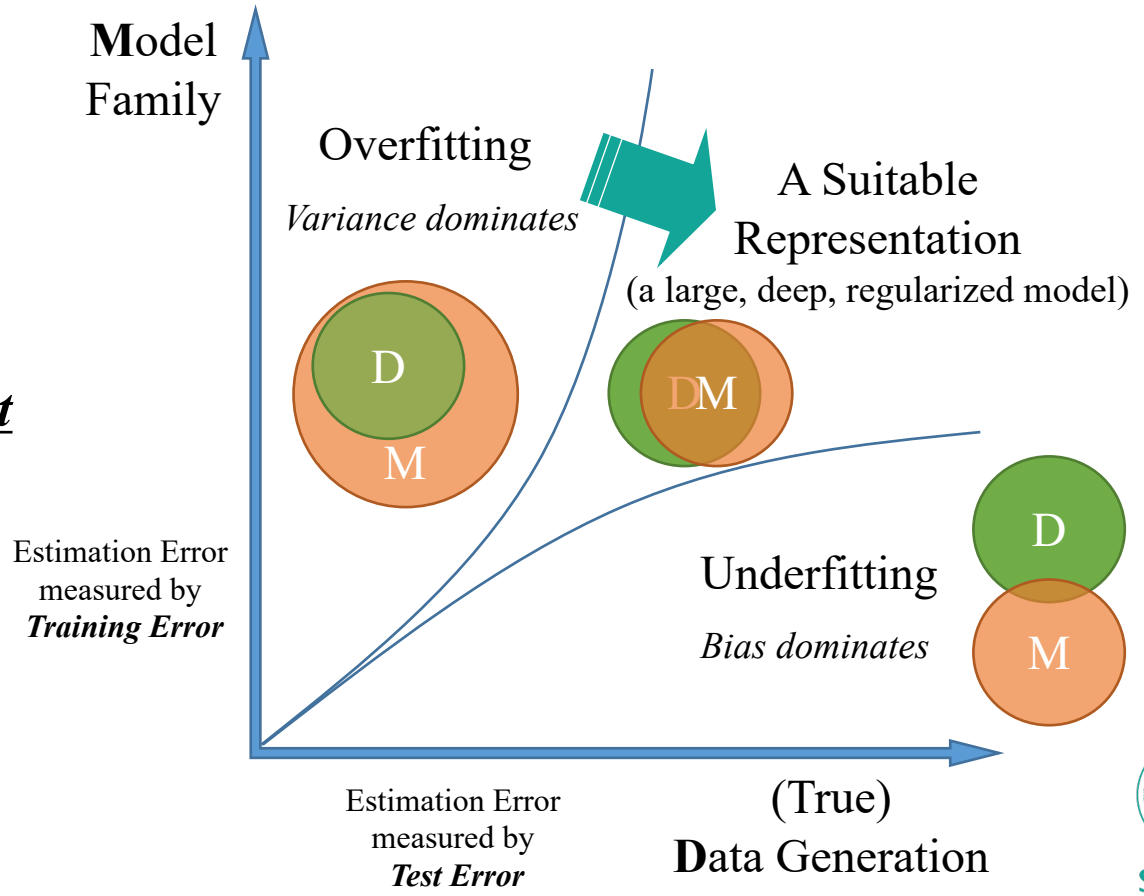
AncoraSIR.com

Regularization in General

Any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error

Regularization of a mismatch

- **Data generation**: almost never have access to the true data generation process
- **Model representation**: not sure if our model family covers the data generation or not
- More memorization capacity naturally tends to ***overfit***
  - Limited memorization capacity won't be able to learn the mapping, causing ***underfitting***
- The best fitting model is a large model that has been ***regularized appropriately***



AncoraSIR.com

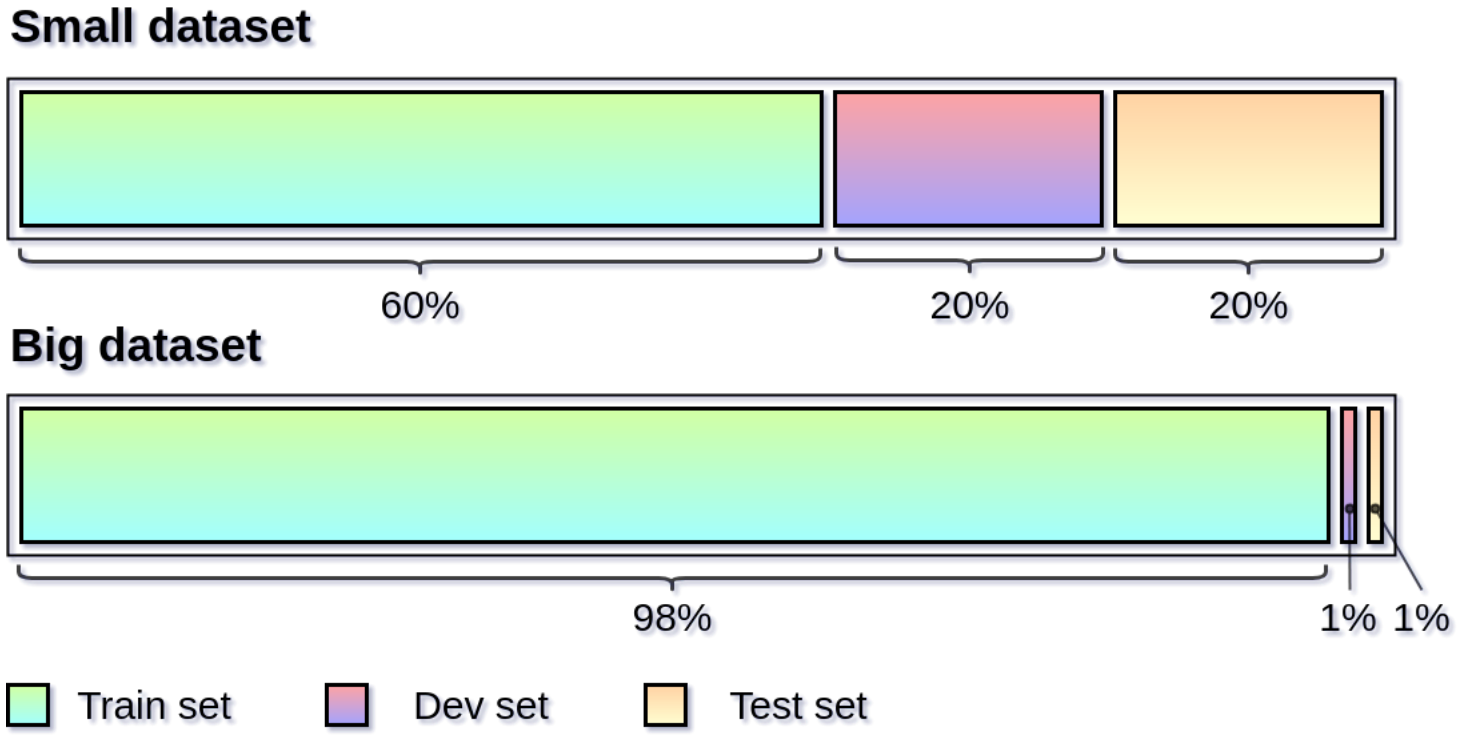
Training, Validation, and Test Dataset

It is a good practice to divide our dataset into three parts to reproduce the real conditions as much as possible

- Make sure that your cross-validation and test set come from the same distribution as well as that they accurately reflect data that we expect to receive in the future

- *dev* and *test* sets should be simply large enough to give us high confidence in the performance of our model.

- **(Common) Small dataset**: 60:20:20
- **(Very) Big dataset**: 98:1:1



Traing set: the data our mode learns from
Dev set: track our progress and draw conclusions to optimise the model
Test set: use at the end of the training process to evaluate the performance of our model

AncoraSIR.com

Data Preprocessing

How to preprocess image data?

Mean subtraction

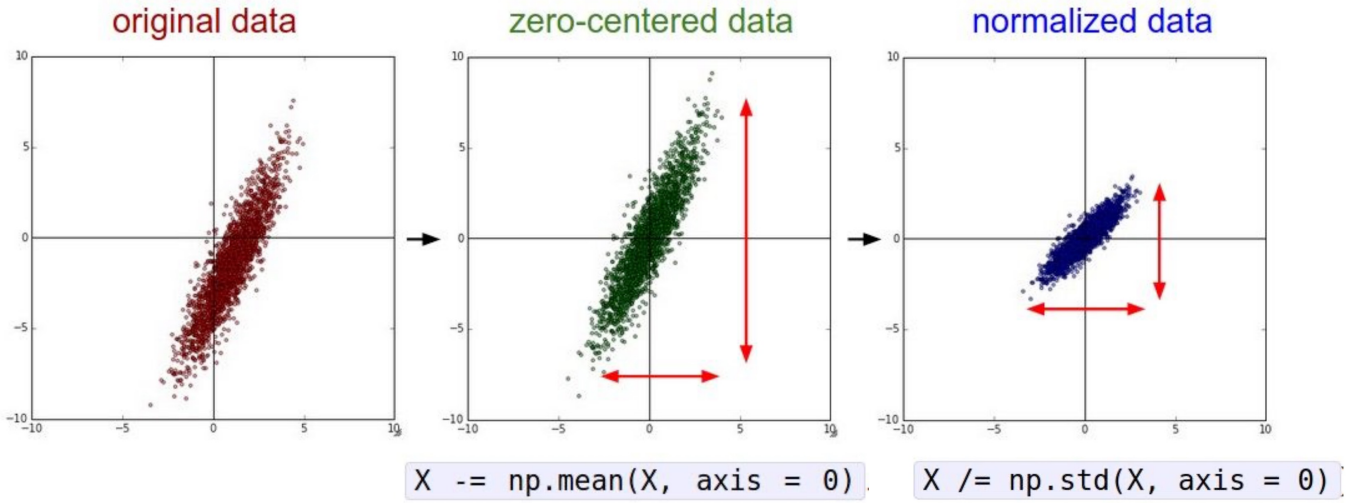
- Subtracting the mean across every individual feature in the data

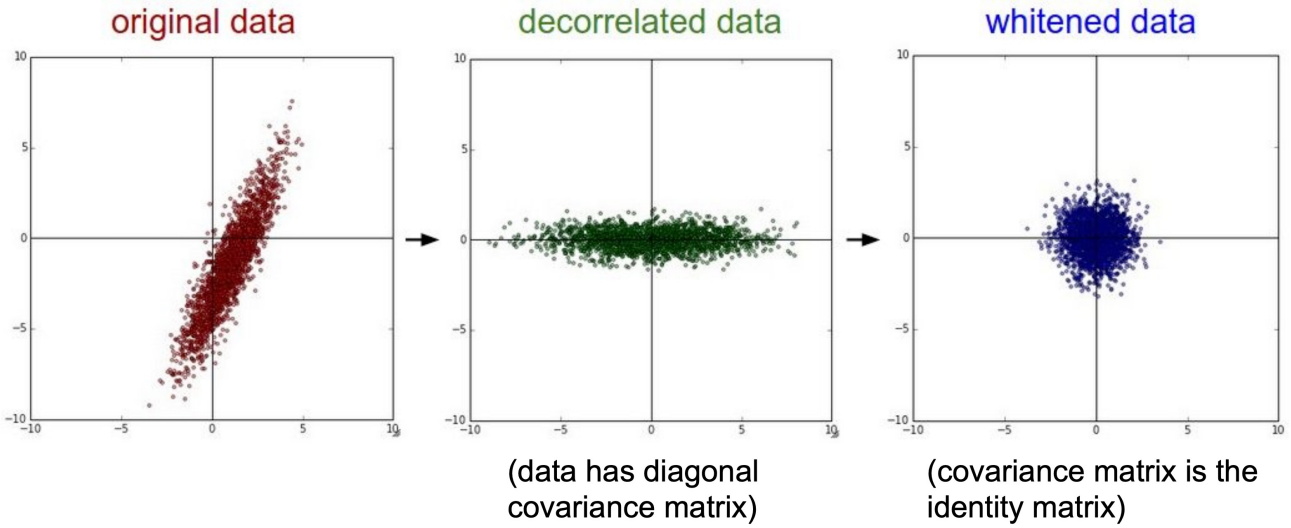
Normalization

- Normalizing the data dimensions so that they are of approximately the same scale.
  - One is to divide each dimension by its standard deviation, once it has been zero-centered.
  - Another is to normalize each dimension so that the min and max along the dimension is -1 and 1 respectively.

PCA and Whitening

- In this process, the data is first centered as described above.
- Then, we can compute the covariance matrix that tells us about the correlation structure in the data





AncoraSIR.com

Batch Normalization

Forcing the activations throughout a network to take on a unit gaussian distribution at the beginning of the training

- Common practice of setting up the initial weights (small but not destructive)
  - Small random numbers => $W = 0.01 * np.random.randn(D, H)$
    - Setup the neurons to be all random and unique in the beginning, so they will compute distinct updates and integrate themselves as diverse parts of the full network
    - $randn$ samples from a zero mean, unit standard deviation gaussian
    - Any problem with ***backpropagation***?

- **How to properly initializing neural networks?**
  - Batch Normalization
    - When used with mini-batches in stochastic gradient training, each mini-batch produces estimates of the mean and variance of each activation
  - Fully Connected Layer => BatchNorm Layer => Non-linear Activation
    - Doing preprocessing at every layer of the network,
      but integrated into the network itself in a differentiable manner.

https://arxiv.org/pdf/1502.03167.pdf

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

AncoraSIR.com

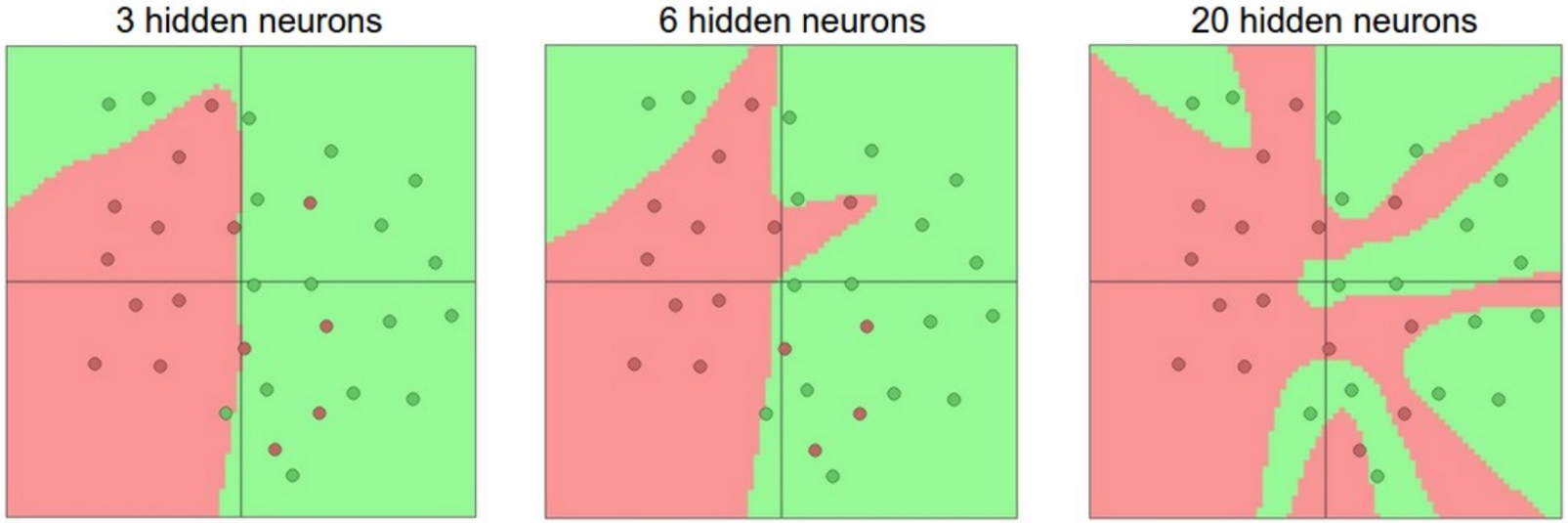
SUSTech Southern University of Science and Technology

Reducing the Network's Size

The simplest way to prevent overfitting

- Reduce the size of the model
  - the number of learnable parameters in the model (the model's ***capacity***)
  - (the number of layers + the number of units per layer)
- Deep learning models tend to be good at fitting to the training data,
  - but the real challenge is generalization, not fitting

Larger Neural Networks can represent more complicated functions



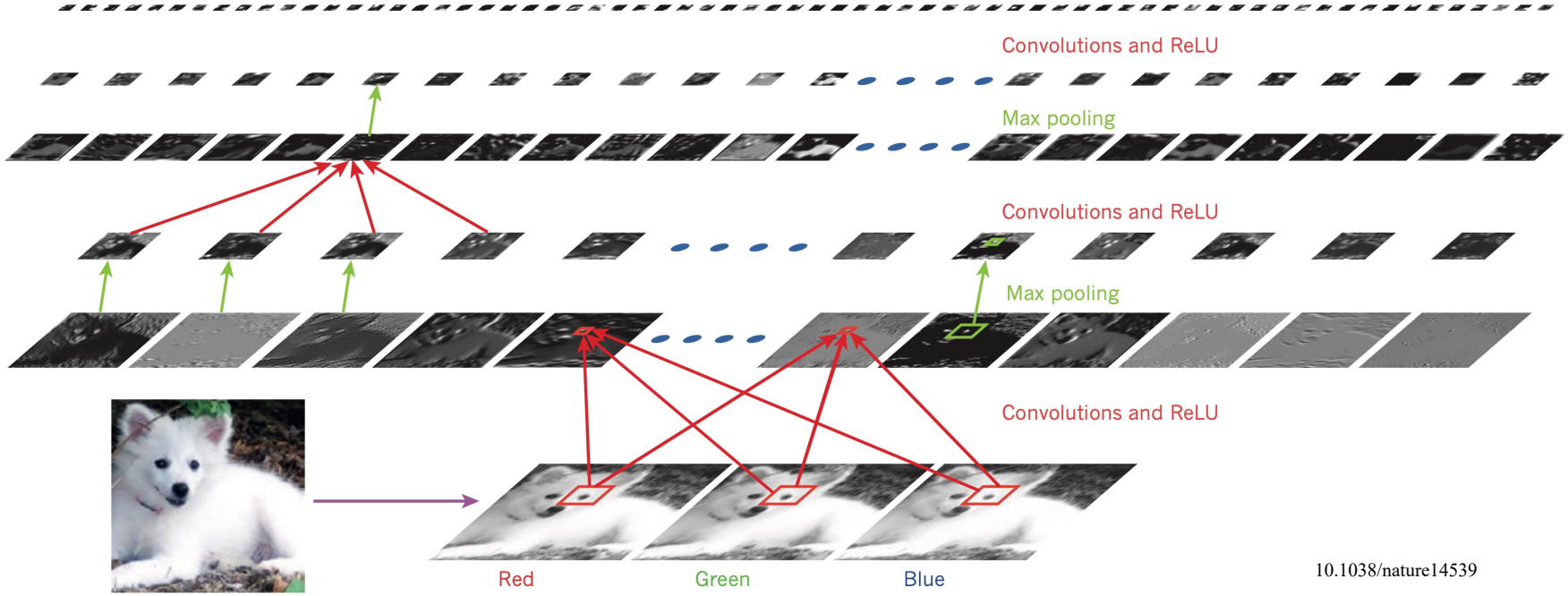
AncoraSIR.com

Overfitting occurs when a model with high capacity fts the noise in the data instead of the (assumed) underlying relationship

To Find an Appropriate Model Size

A General Workflow

- Start with relatively few layers and parameters
- Increase the size of the layers or add new layers
- Until you see diminishing returns with regard to validation loss.

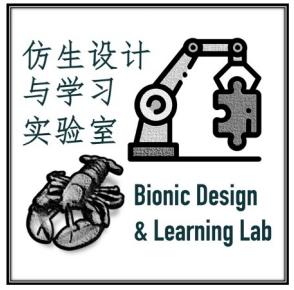


10.1038/nature14539

AncoraSIR.com

SUSTech
Southern University of Science and Technology

A Few Regularization Methods



AncoraSIR.com

SUSTech
Southern University
of Science and Technology

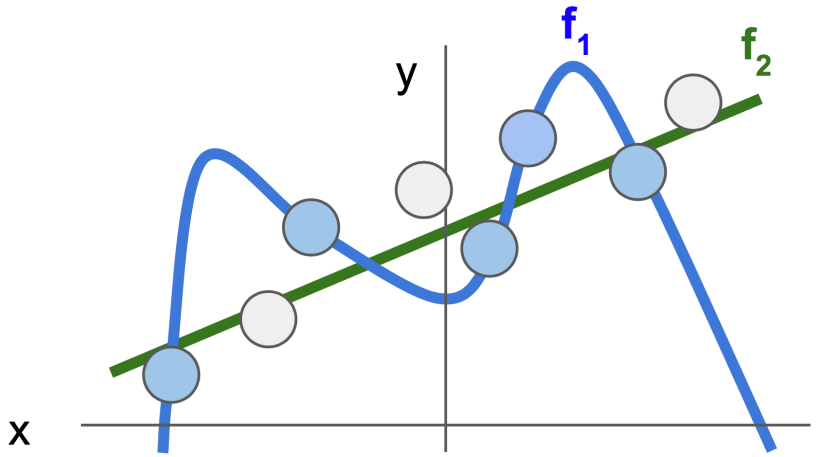
Norm Penalties as Constrained Optimization

Weight Regularization

- **Intuition**: Occam's razor
  - The simplest solution is most likely the right one—the one that makes fewer assumptions

- **Formulation**: A simple model is always preferred
  - A model where the distribution of parameter values has less entropy (or a model with fewer parameters)

- **Method**: Constrain the network complexity
  - Forcing its weights to take only small values
  - Adding a cost associated with having large weights to the loss function of the network



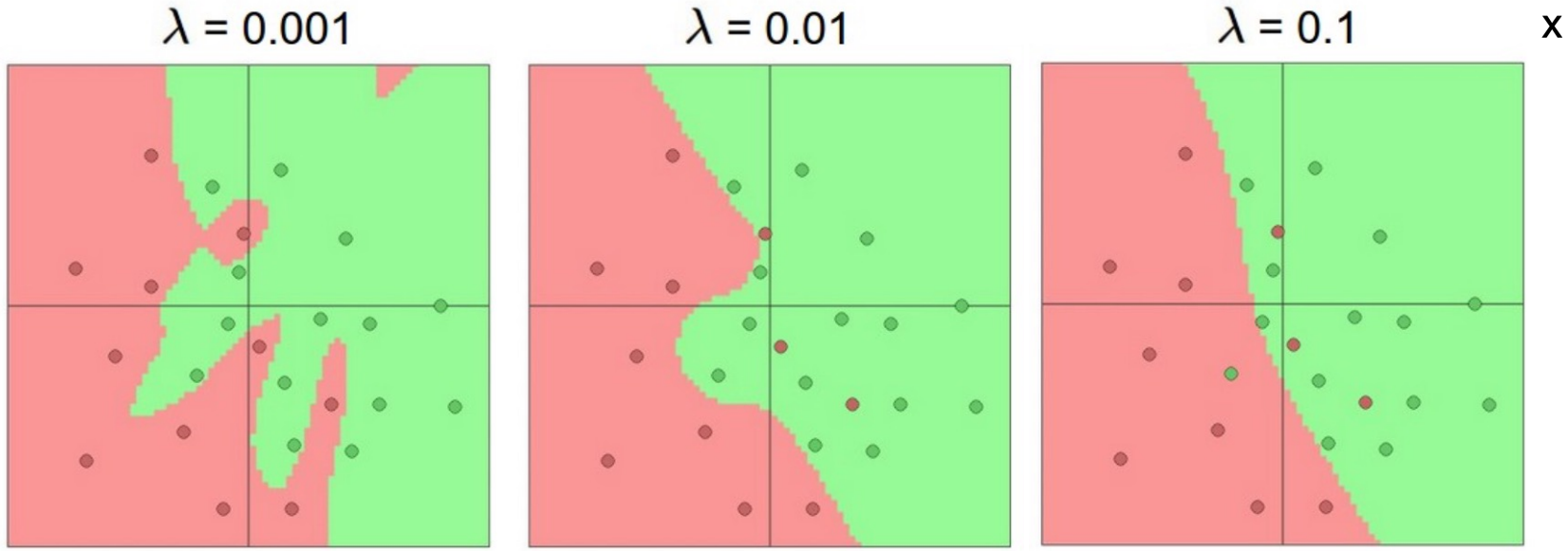
Regularization pushes against fitting the data *too* well so we don't fit noise in the data

AncoraSIR.com

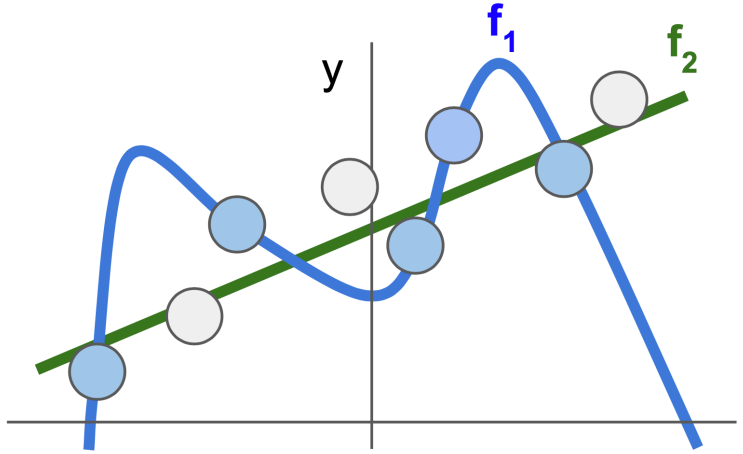
Norm Penalties as Constrained Optimization

Weight Regularization

- **Weight Regularization**
  - L1 regularization: $R(W) = \sum_k \sum_l |W_{k,l}|$
    - The cost added is proportional to the *absolute value of the weight coefficients*
  - L2 regularization: $R(W) = \sum_k \sum_l W_{k,l}^2$
    - The cost added is proportional to the *square of the value of the weight coefficien*



AncoraSIR.com



Regularization pushes against fitting the data *too* well so we don't fit noise in the data

$$L(W) = \frac{1}{N}\sum_{1}^{N} L_i(\hat{y}_i, y_i) + \lambda R(W)$$

λ as strength of Regularization (*hyperparameter*)

Data loss
Model predictions should match training data

Regularization
Prevent the model from doing too well on training data

Elastic Net (L1+L2)

- $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

Dataset Augmentation

Create fake data and add it to the training set, particularly effective for object recognition

- We are always limited by the amount of data available for generalization
- Why Images?
  - high dimensional
  - include an enormous variety of factors, many of which can be easily simulated

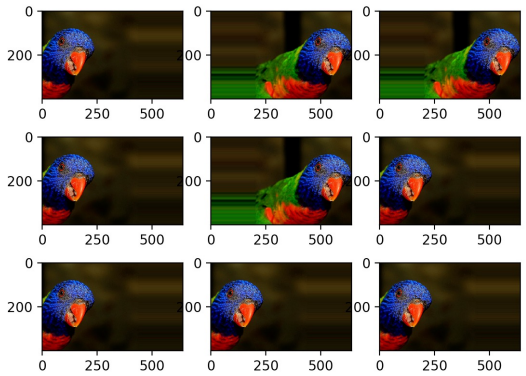
https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

AncoraSIR.com

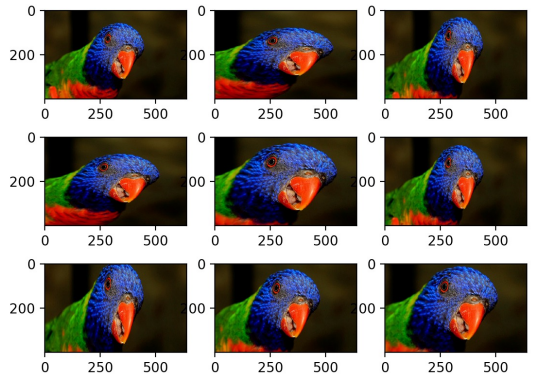
Dataset Augmentation

Create fake data and add it to the training set, particularly effective for object recognition

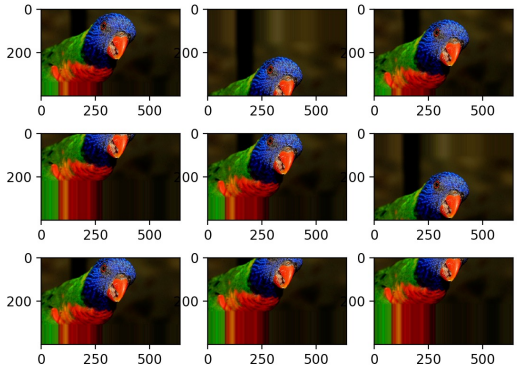
Horizontal Shift



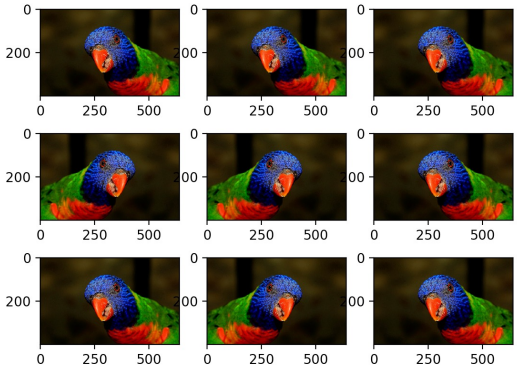
Random Zoom (Noise)



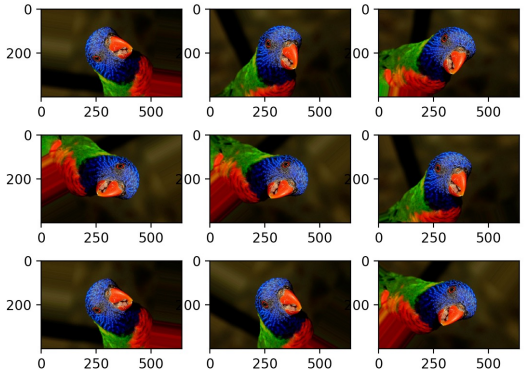
Vertical Shift



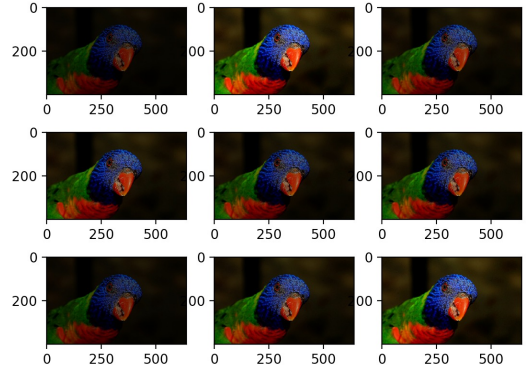
Flip with H/V Shift



Rotation Shift



Brightness Shift (Noise)

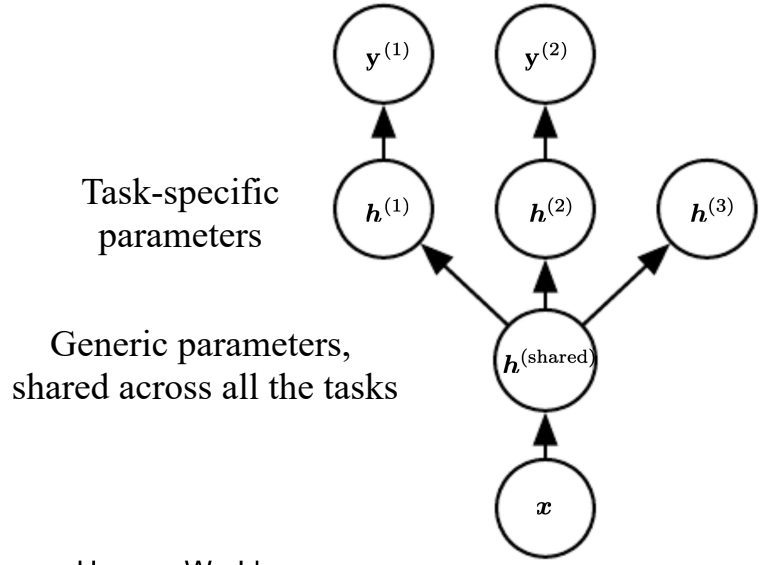


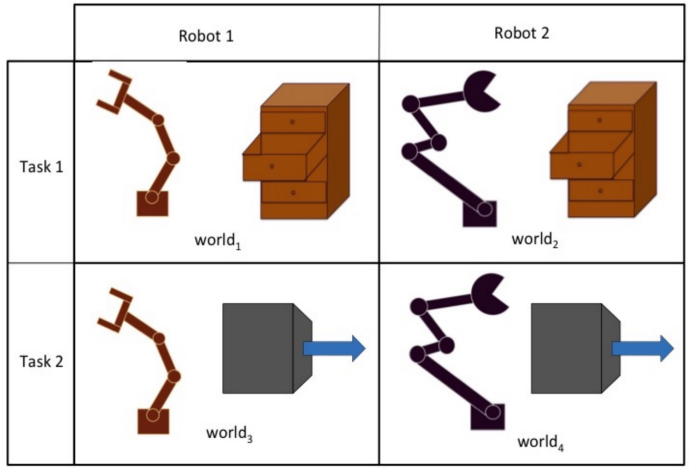
AncoraSIR.com

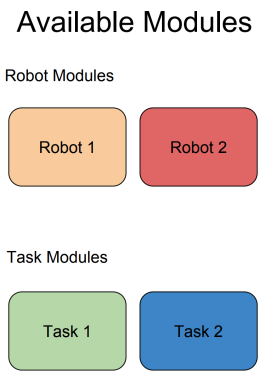
Multi-Task Learning

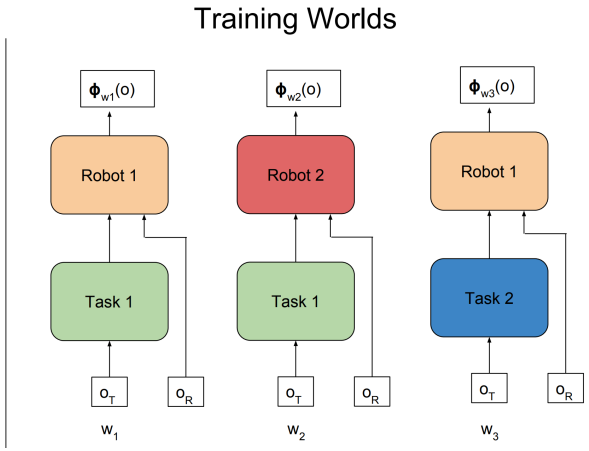
Improve generalization by pooling the examples arising out of several tasks.

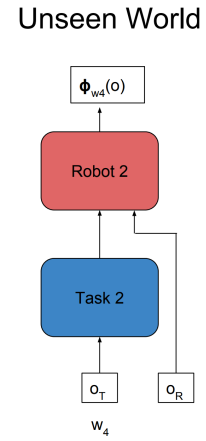
- A common situation where
  - the tasks share a common input
  - but involve different target random variables
- The underlying assumption
  - there exists a common pool of factors that explain the variations in the input $\boldsymbol{x}$,
  - while each task is associated with a subset of these factors

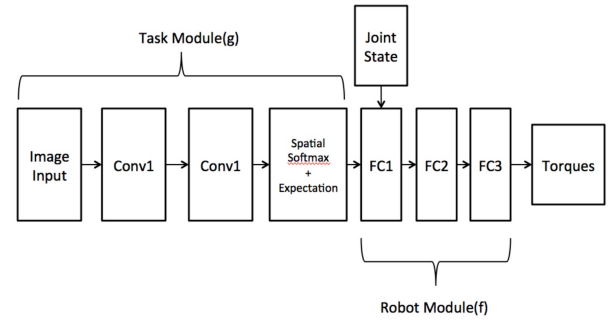










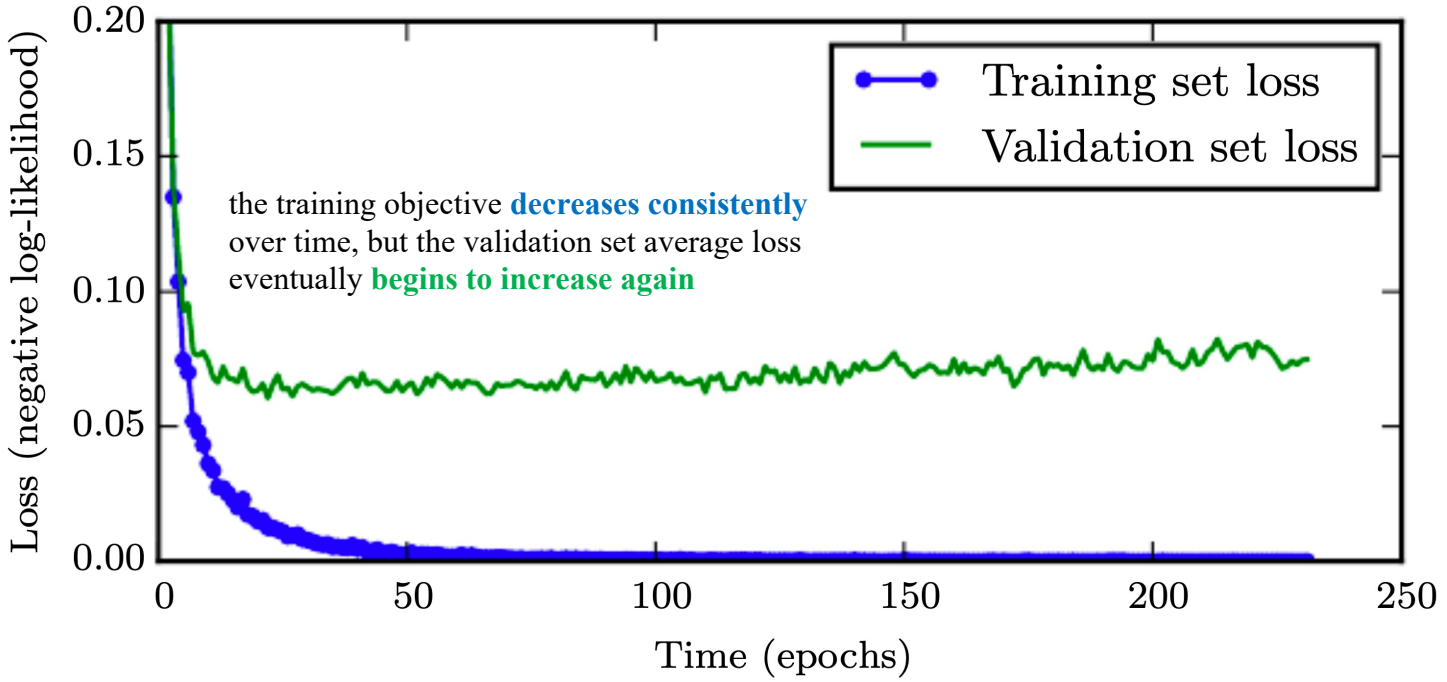


https://arxiv.org/pdf/1609.07088.pdf

Early Stopping

Due to its simplicity and effectiveness, it is probably the most commonly used form of regularization in deep learning

- We can obtain a model with better validation set error (and thus, hopefully better test set error)
  - By returning to the parameter setting at the point in time with the lowest validation set error.
- Every time the error on the validation set improves, we store a copy of the model parameters
- As a very efficient hyperparameter selection algorithm
  - The number of training steps is just another hyperparameter



Learning curves showing how
the negative log-likelihood loss changes over time

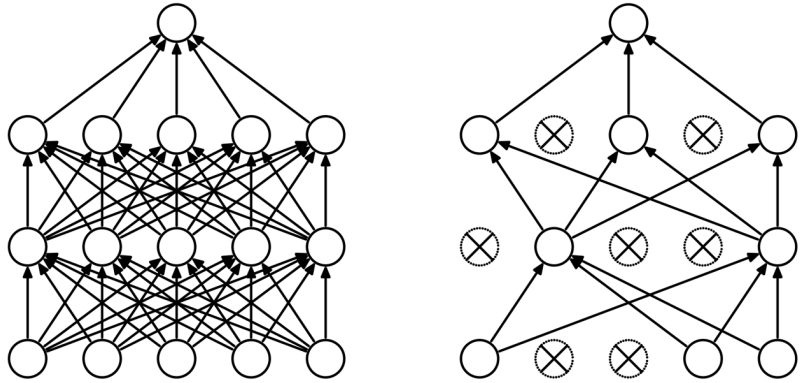
(***epochs***: the number of training iterations over the dataset)

AncoraSIR.com

Dropout

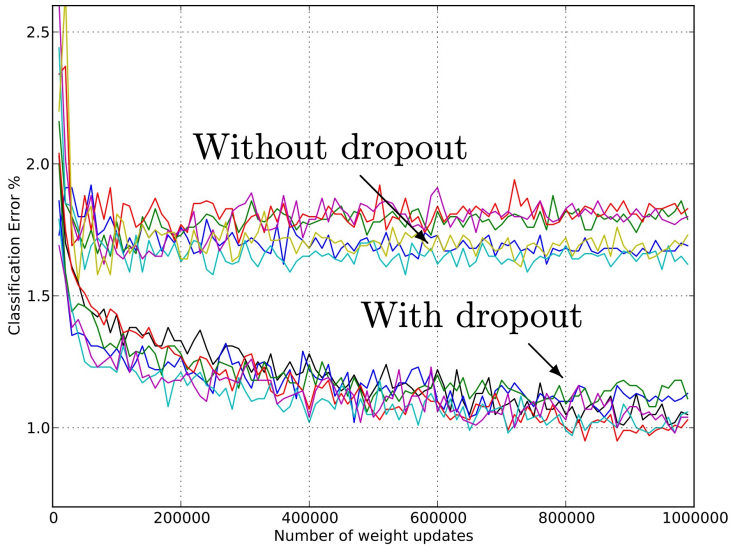
Randomly drop units (along with their connections) from the neural network during training

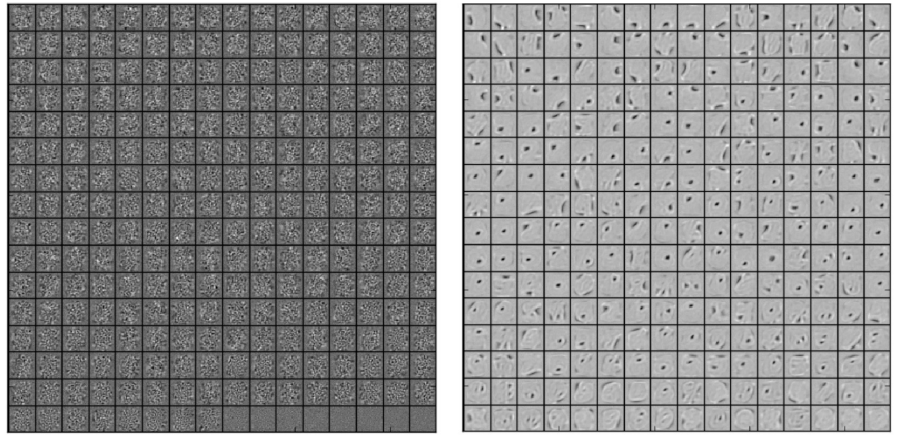
- The dropout rate
  - The fraction of the features that are zeroed out;
  - Usually set between 0.2 and 0.5.
- Dropout improves the performance of neural networks on supervised learning tasks significantly



(a) Standard Neural Net (b) After applying dropout.

Method	Unit Type	Architecture	Error %
Standard Neural Net (Simard et al., 2003)	Logistic	2 layers, 800 units	1.60
SVM Gaussian kernel	NA	NA	1.40
Dropout NN	Logistic	3 layers, 1024 units	1.35
Dropout NN	ReLU	3 layers, 1024 units	1.25
Dropout NN + max-norm constraint	ReLU	3 layers, 1024 units	1.06
Dropout NN + max-norm constraint	ReLU	3 layers, 2048 units	1.04
Dropout NN + max-norm constraint	ReLU	2 layers, 4096 units	1.01
Dropout NN + max-norm constraint	ReLU	2 layers, 8192 units	0.95
Dropout NN + max-norm constraint (Goodfellow et al., 2013)	Maxout	2 layers, (5 × 240) units	0.94
DBN + finetuning (Hinton and Salakhutdinov, 2006)	Logistic	500-500-2000	1.18
DBM + finetuning (Salakhutdinov and Hinton, 2009)	Logistic	500-500-2000	0.96
DBN + dropout finetuning	Logistic	500-500-2000	0.92
DBM + dropout finetuning	Logistic	500-500-2000	**0.79**



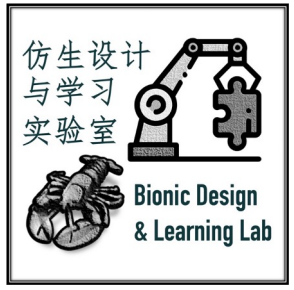


(a) Without dropout (b) Dropout with $p = 0.5$.

AncoraSIR.com http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

SUSTech Southern University of Science and Technology

Optimization for Deep Models



AncoraSIR.com

SUSTech
Southern University
of Science and Technology

Optimization in General

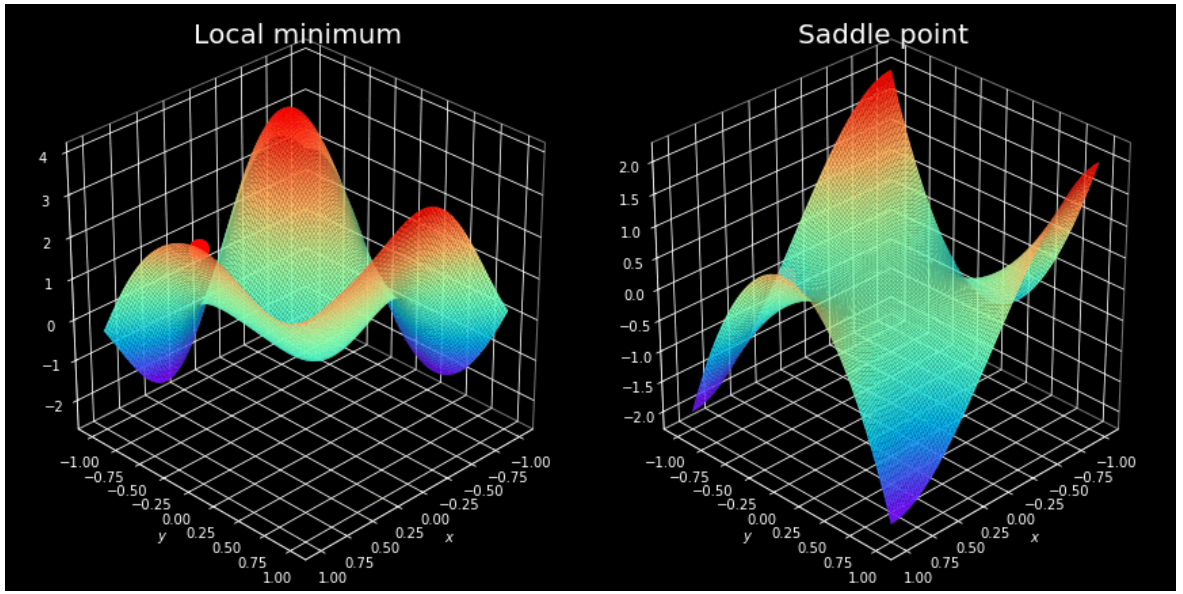
Optimization is a process of searching for parameters that minimize or maximize our functions

- A Design Challenge with Indirect Optimzation

- Accuracy, precision or recall …
  - How well our model solves a given problem?
  - ***Things we really care about***

- Sum of Squared Error, Maximum-Likelihood …
  - Optimizing a different cost function $J(\theta)$ and hope that minimizing its value will improve metric we care about
  - ***Things we are actually computing***

Advanced algorithms are usually needed
to find a minimum of non-convex cost functions



Points where function takes a minimum value, but only in a given region

Plateaus where the cost function is almost constant (the gradient is almost ***zeroed*** in all directions making it impossible to escape)

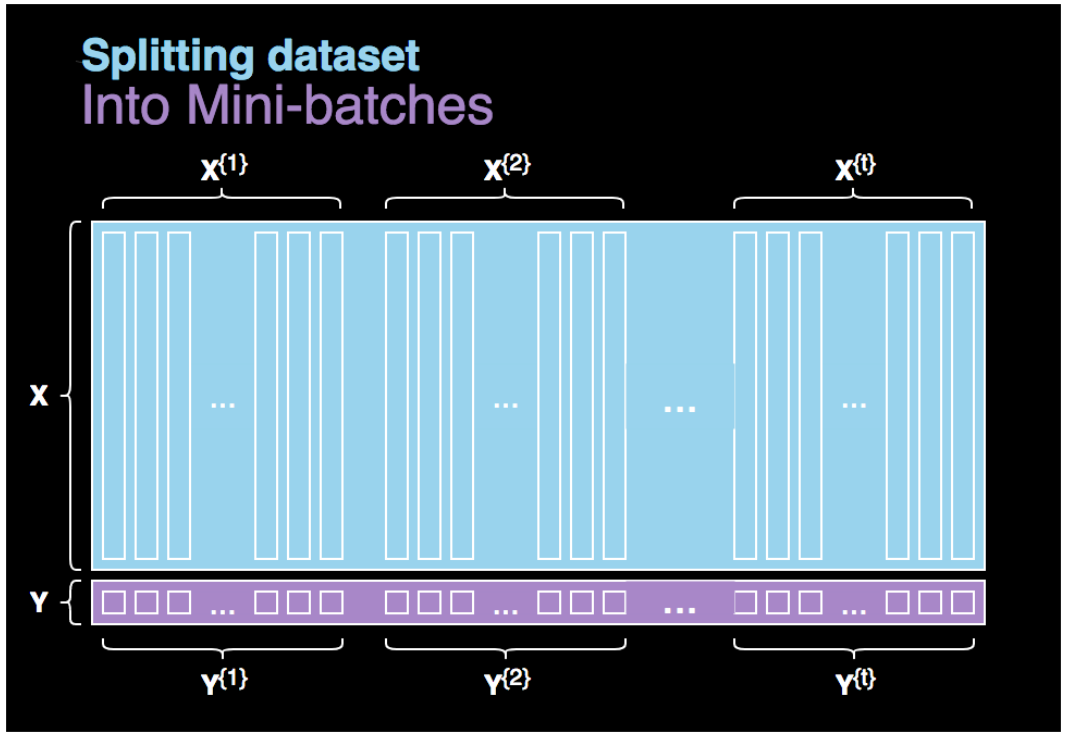
AncoraSIR.com

Mini-batch Gradient Descent

Improve the efficiency of gradient processing

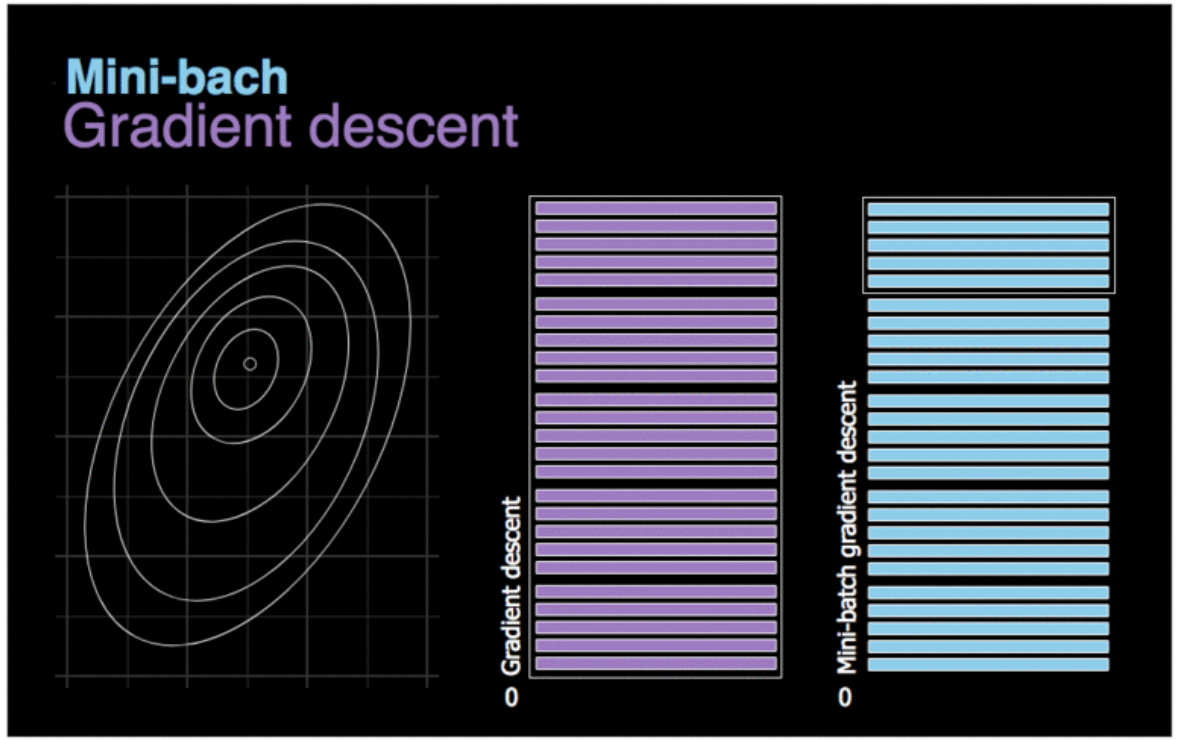
Vectorization: handling many training examples at once

Mini-batches: further split the dataset for iterative training



Gradient descent: takes longer to process with a smoother path

Mini-bach gradient descent: much faster but also noisy



Stochastic Gradient Descent is also an effective alternative (GD is K=1, Overfit if K=N)

AncoraSIR.com

Stochastic Gradient Descent

In practice, it is necessary to gradually decrease the learning rate over time

Algorithm 8.1 Stochastic gradient descent (SGD) update at training iteration k

Require: Learning rate ϵ_k.

Require: Initial parameter $\boldsymbol{\theta}$

while stopping criterion not met **do**

Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.

Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow +\frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$

Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \hat{\boldsymbol{g}}$

end while

$\boldsymbol{w}$ in our case

Introduces a source of noise that does not vanish even when we arrive at a minimum.

In practice, it is common to decay the learning rate linearly until iteration τ:

$$\epsilon_k = (1 - \alpha)\epsilon_0 + \alpha\epsilon_\tau \qquad \alpha = \frac{k}{\tau}$$

After iteration τ , it is common to leave ϵ constant.

The "art" of choosing a learning rate

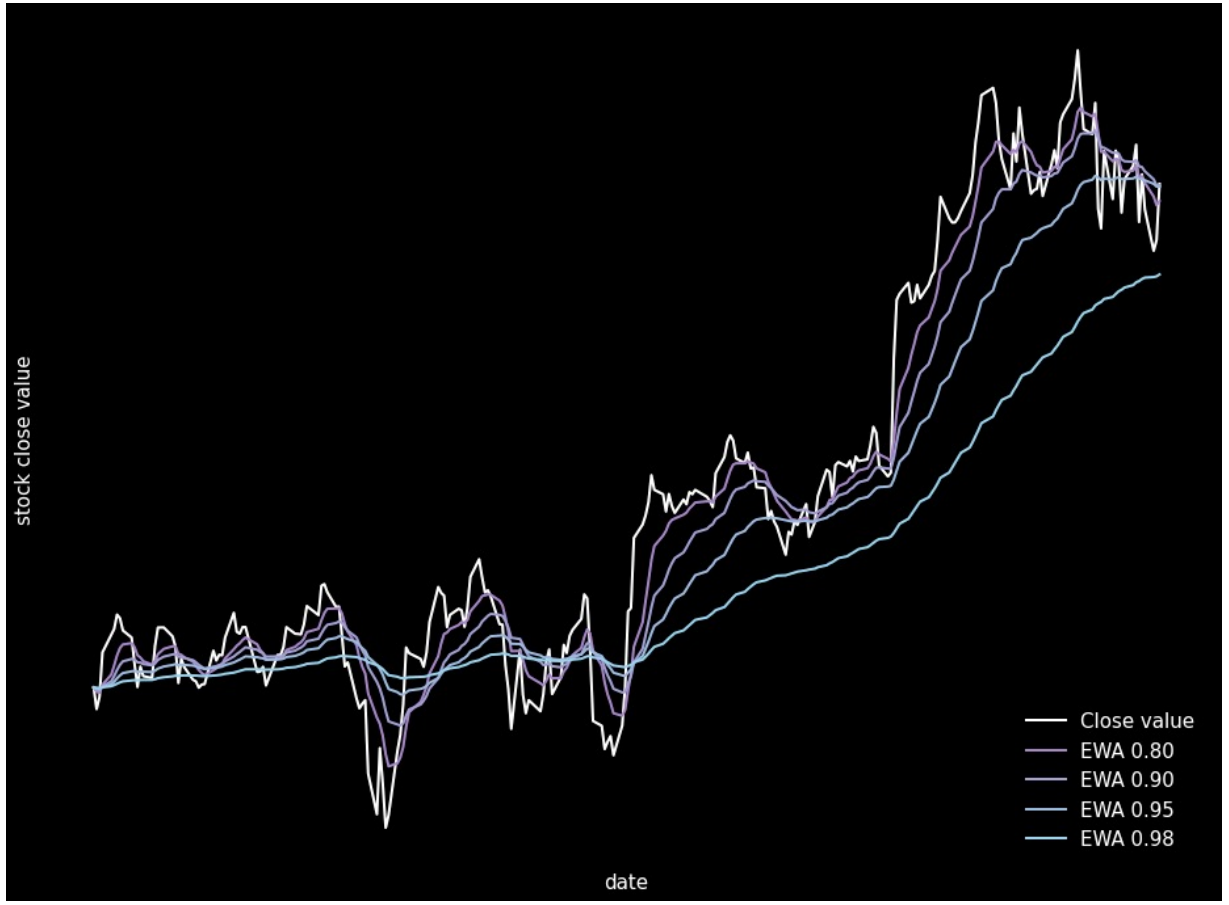
- by trial and error
- monitor learning curves that plot the objective function as a function of time.
- ϵ_0: it is usually best to monitor the first several iterations and use a learning rate that is higher than the best-performing learning rate at this time, but not so high that it causes severe instability.

AncoraSIR.com

Exponentially Weighted Averages

Averaging many previous values in order to become independent of local fluctuations and focus on the overall trend

- $v_t = \beta v_{t-1} + (1-\beta)\theta_t$
  - β: **control the range of values to be averaged**
- $v_0 = 0$ (initialize)
- $v_1 = \beta v_0 + (1-\beta)\theta_1 = (1-\beta)\theta_1$
- $v_2 = \beta v_1 + (1-\beta)\theta_2 = (1-\beta)(\beta\theta_1 + \theta_2)$
- $v_3 = \beta v_2 + (1-\beta)\theta_3 = (1-\beta)(\beta^2\theta_1 + \beta\theta_2 + \theta_3)$
- …
- $v_t = \beta v_{t-1} + (1-\beta)\theta_t = \frac{1}{1/(1-\beta)}\sum_{i=1}^{t}\beta^{t-i}\theta_i$
- Intuition:
  - Can be considered as a weighted sum of $1/(1-\beta)$ samples before the current time instance
    - *If* $\beta = 0.9$, v_t *average the previous* $\frac{1}{(1-\beta)} = 10$ *days*
    - *If* $\beta = 0.98$, v_t *average the previous* $\frac{1}{(1-\beta)} = 500$ *days*
  - Highly effective when applied to deep learning



AncoraSIR.com

SUSTech
Southern University of Science and Technology

Gradient Descent with Momentum

To gain momentum, so that even if the local gradient is zero, we still move forward relying on the previously calculated values

Compute an **E**xponentially **W**eighted **A**verage of your gradients, and then use that gradient to update your weights instead

One iteration t

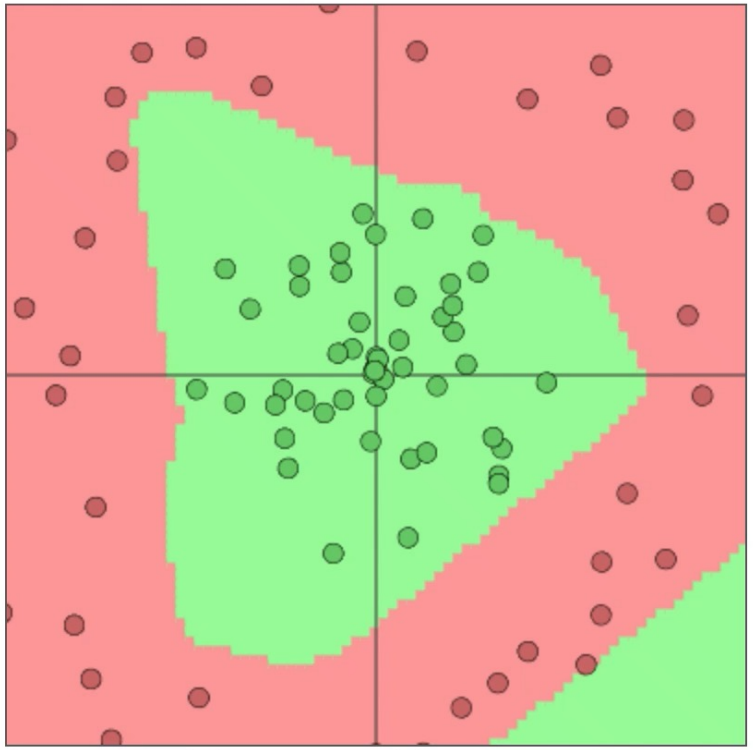
- Compute dW and db on the current mini-batch
- $v_{dW} = \beta v_{dW} + (1-\beta) \cdot dW$
- $v_{db} = \beta v_{db} + (1-\beta) \cdot db$
- $W = W - \alpha v_{dW}$
- $b = b - \alpha v_{db}$
- $(\theta \; as \; a \; collection \; of \; W \; and \; b)$

AncoraSIR.com

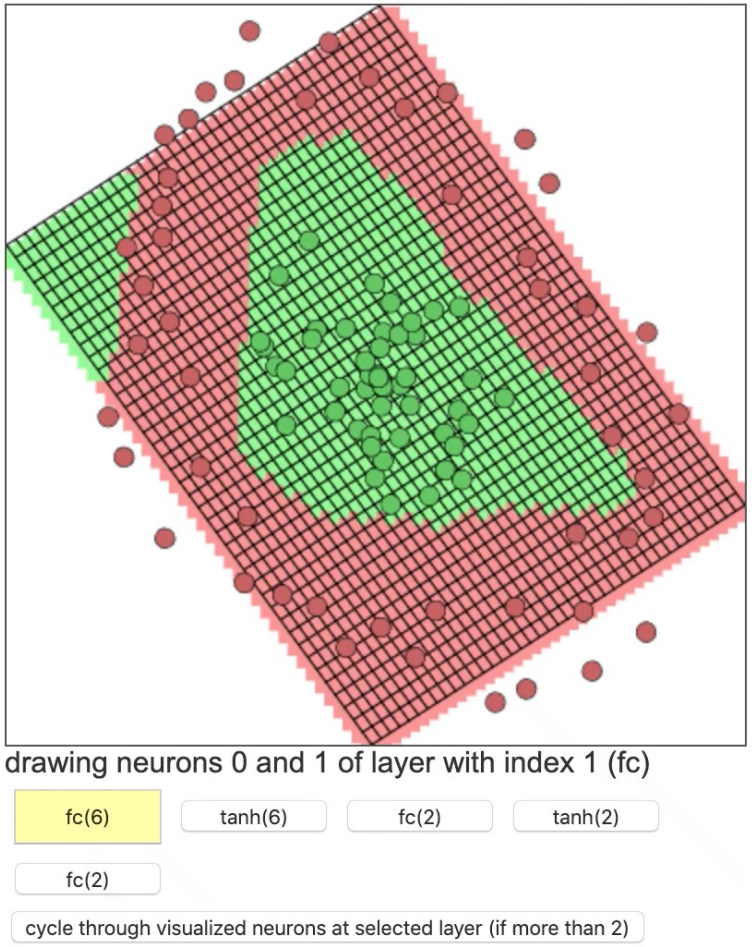
Standard gradient descent vs. GD with momentum

Classify Toy 2-D data with a Neural Network

https://cs.stanford.edu/people/karpathy/convnetjs/



Controls:
CLICK: Add red data point
SHIFT+CLICK: Add green data point
CTRL+CLICK: Remove closest data point



Thank you~

songcy@sustech.edu.cn