Deep Learning (BEV033DLE)
Lecture 7. Regularization

Alexander Shekhovtsov

Czech Technical University in Prague

Regularization Methods

✦ **Problem: Overfitting to a Finite Training Data**

- Can classify the training data with 100% accuracy
- Test error is substantially larger than training error

✦ **Explicit Regularization:**

- Parameter Regularization (L1, L2, General norms)
- Data Augmentation (random transforms, noises)
- Dropout
- Cross-validation (hyper parameters, early stopping, model selection)
- ...

✦ **Implicit Regularization (Side Effects):**

- SGD, Initialization, Batch Normalization

✦ **Everything Else:**

- Network Architecture (invariances, parameter sharing, depth, attention)
- Loss function, task formulation, transfer learning

Parameter Regularization

General Setup

- Regularized training objective:

  $\min_\theta \mathcal{L}(\theta) + \lambda R(\theta) = \min_\theta \sum_i l_i(\theta) + \lambda R(\theta)$

  - $l_i(\theta)$ - loss function for the data point (x_i, y_i)
  - $R(\theta)$ - function not depending on data
  - λ - regularization strength
- Can be interpreted as maximum a posteriori (MAP) parameter estimation:
  - $p(D|\theta)$ - likelihood of the data given parameters
  - $p(\theta) \propto \exp(-\lambda R(\theta))$ - prior on the model parameters
  - $p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$ - Bayesian posterior over parameters
  - $\hat{\theta}_{\text{MAP}} = \operatorname{argmax}_\theta p(D|\theta)p(\theta) = \operatorname{argmax}_\theta \log p(D|\theta) + \log p(\theta)$

  [RPZ lecture 3:(Parameter Estimation: Maximum a Posteriori (MAP))]

- In practice also commonly appears in the form independent of the amount of data:

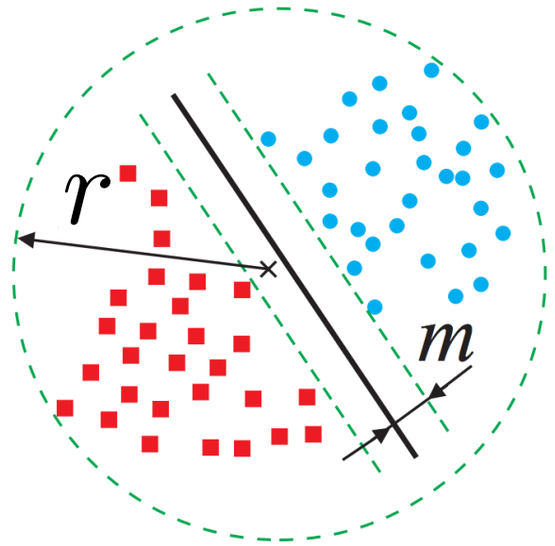
  $\min_\theta \frac{1}{n} \sum_i l_i(\theta) + \lambda R(\theta)$

  - λ is tuned for a given dataset with cross-validation

L_2 Regularization (Weight Decay)

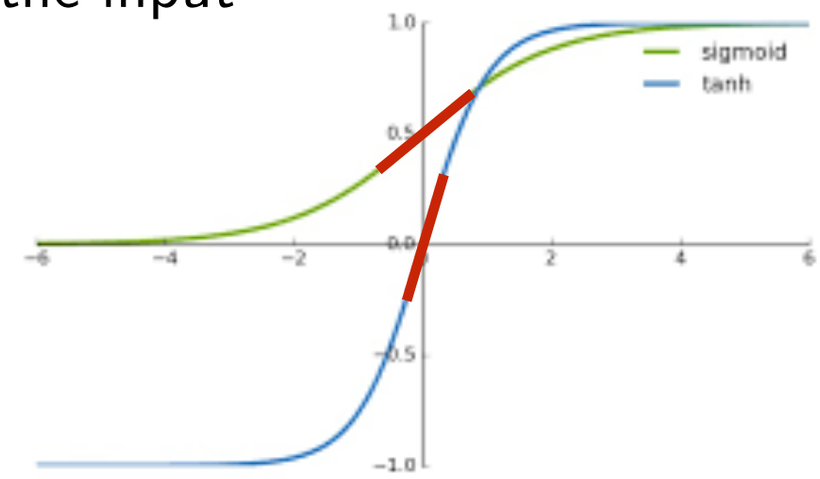
- L_2-regularization (l_2, weight decay): $R(\theta) = \frac{1}{2}\|\theta\|^2$
- In **SVMs**:
  - Maximize margin and minimize slacks:

    $\min_{w,b} \sum_i \mathrm{ReLU}(1 - (w^\mathsf{T} x_i + b) y_i) + \lambda \frac{1}{2}\|w\|^2$
  - Generalization bounds tighter with larger margin, independent of dimensionality



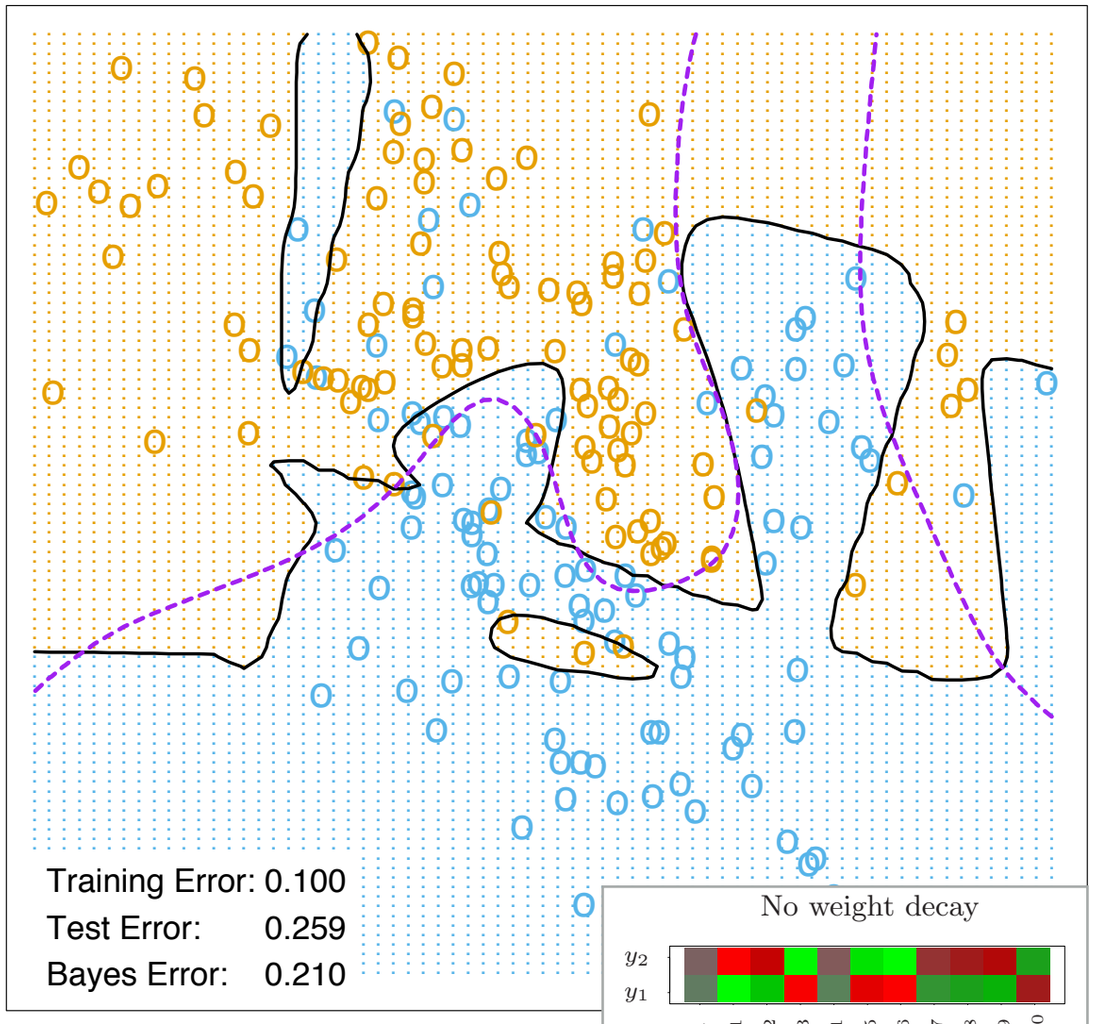
- In **Linear Regression**:
  - Ridge regression: $\min_\theta \sum_i (\theta^\mathsf{T} x_i - y_i)^2 + \frac{\lambda}{2}\|\theta\|^2$
  - Solution: $\hat{\theta} = (X X^\mathsf{T} + \lambda I)^{-1} X^\mathsf{T} y$
  - Helps with underdetermined problems
  - Equivalent to *multiplicative noise* $\mathcal{N}(1, \lambda^2)$ on the input
  - Smoothing effect (reduces variance of $\hat{\theta}$)
- **Sigmoid NNs**:
  - Small $\theta \rightarrow$ sigmoid outputs are close to linear

    $\rightarrow$ smoother classification boundary

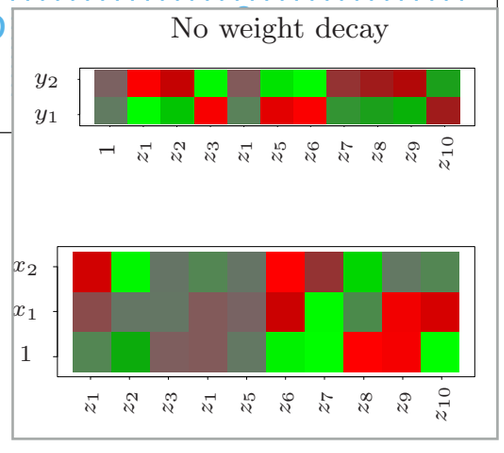


Simulated Data Example

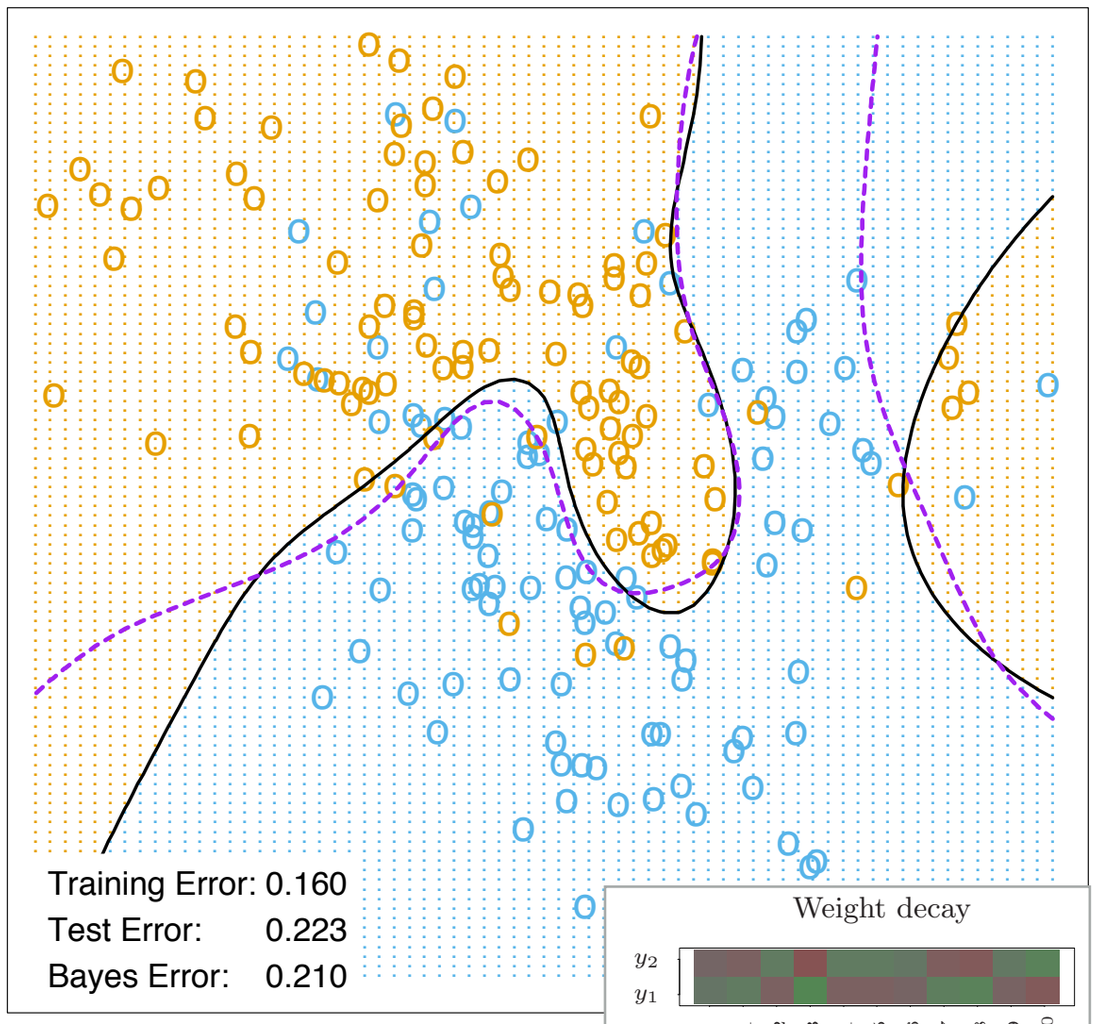
Neural Network - 10 Units, No Weight Decay



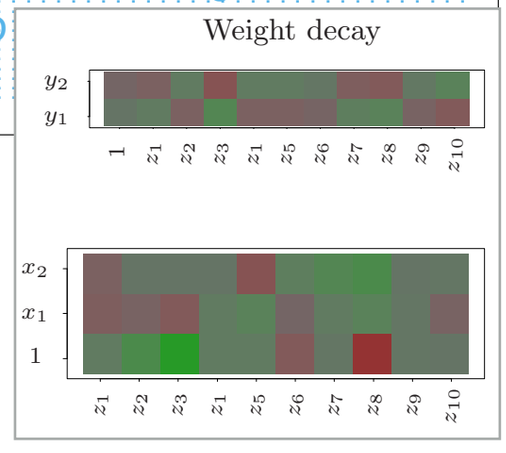
weights

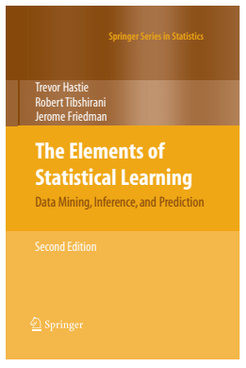


Neural Network - 10 Units, Weight Decay=0.02



weights



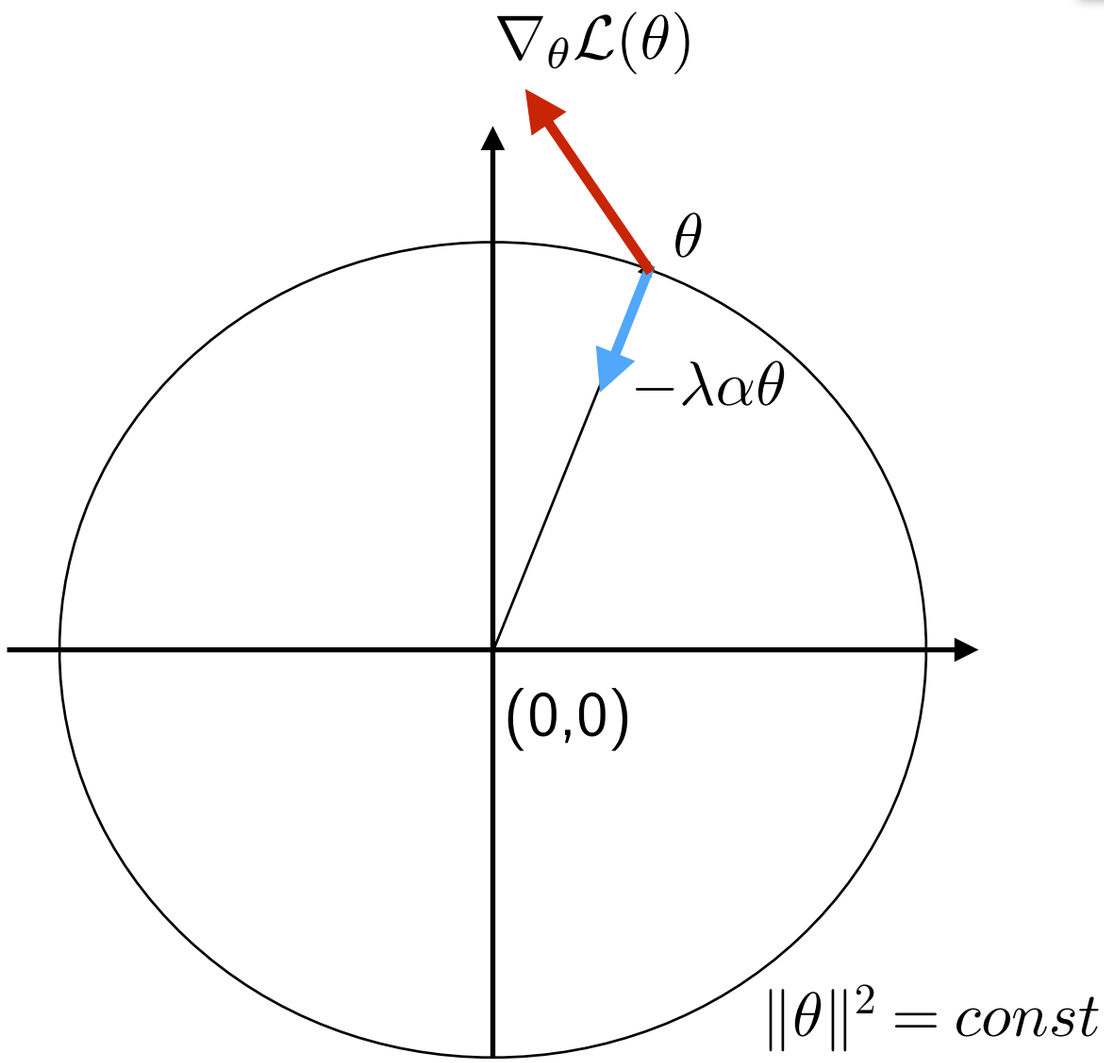


Hastie, Tibshirani and Friedman: The Elements of Statistical Learning

https://web.stanford.edu/~hastie/ElemStatLearn/

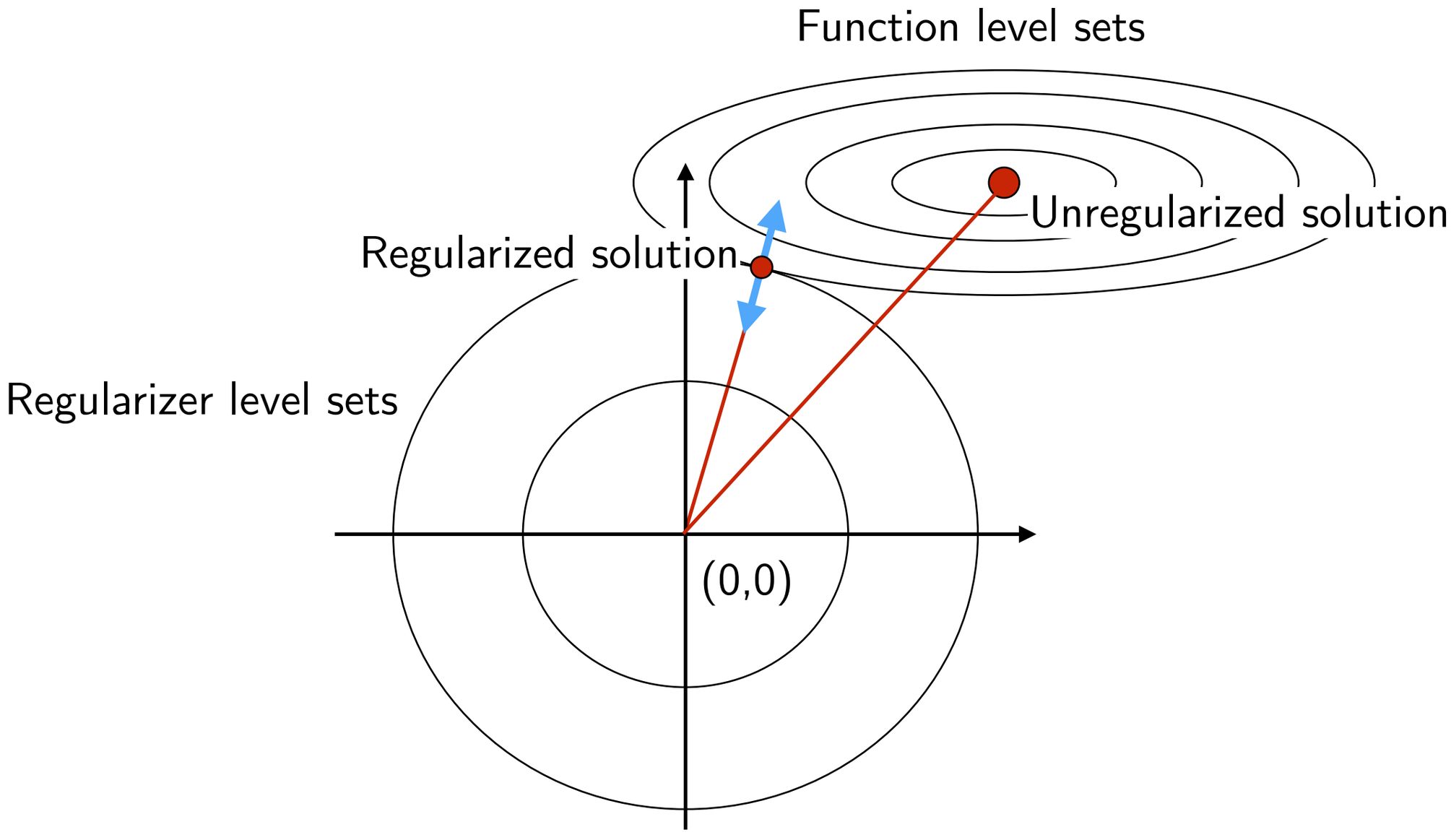
Weight Decay

- $\min_\theta \mathcal{L}(\theta) + \frac{\lambda}{2}\|\theta\|^2$
- Gradient descent:
  - $g^t := \nabla_\theta \mathcal{L}(\theta) + \lambda\theta$
  - $\theta^{t+1} = \theta^t - \alpha g^t$
  - $\theta^{t+1} = \underbrace{(1-\alpha\lambda)}_{\text{decay}}\theta^t - \alpha\nabla_\theta\mathcal{L}(\theta)$



- In neural networks:
  - There is typically a manifold of optimal solutions, small regularization of order 10^{-5} may have effect
  - It may be desirable to exclude biases from regularization

L2 Regularization Effect

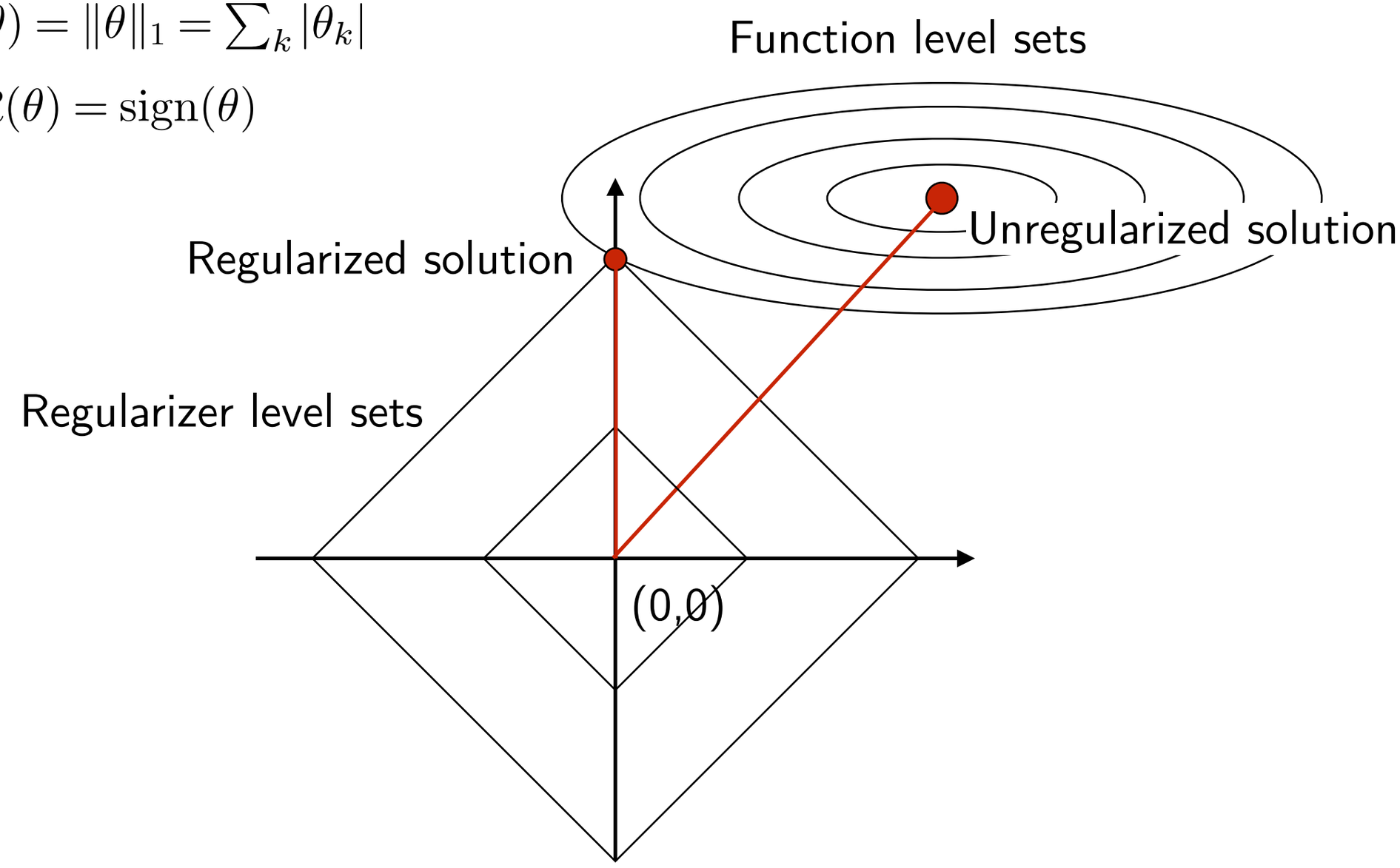


- L_2 regularization effect (approximately quadratic loss):
  - Parameter shrink along eigenvectors of the loss Hessian by $\frac{s_i}{s_i+\lambda}$
  - s_i – eigenvalue, curvature along i'th eigenvector

L1 Regularization

- $R(\theta) = \|\theta\|_1 = \sum_k |\theta_k|$

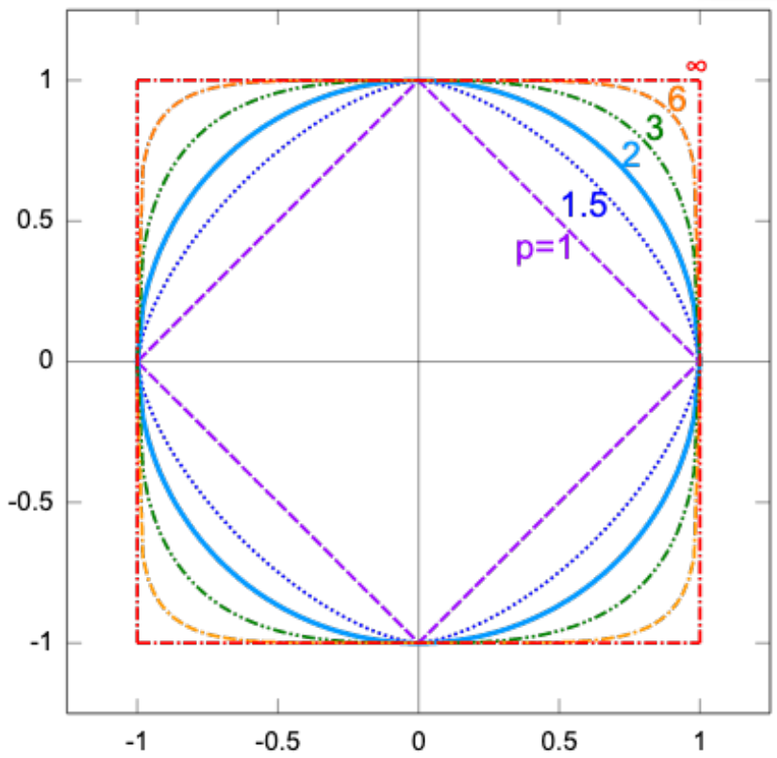
  $\nabla R(\theta) = \text{sign}(\theta)$



- L1 regularization effect:
  - promotes sparsity
  - for better generalization we typically do not want sparsity (= less parameters)

Other Variants

- **More General Norms**:
  - L_p norm: $R(\theta) = \|\theta\|_p = \left(\sum_i |\theta_i|^p\right)^{\frac{1}{p}}$
  - $p < 1$ is closer to counting non-zero weights, *i.e.* sparsity
  - $p = \infty$ results in $R(\theta) = \max_i \theta_i$
  - $L_{p,q}$ norm: $R(W) = \|W\|_{p,q} = \left(\sum_j \left(\sum_i |W_{ij}|^p\right)^{\frac{q}{p}}\right)^{\frac{1}{q}}$



- **Constrained Optimization Form**:

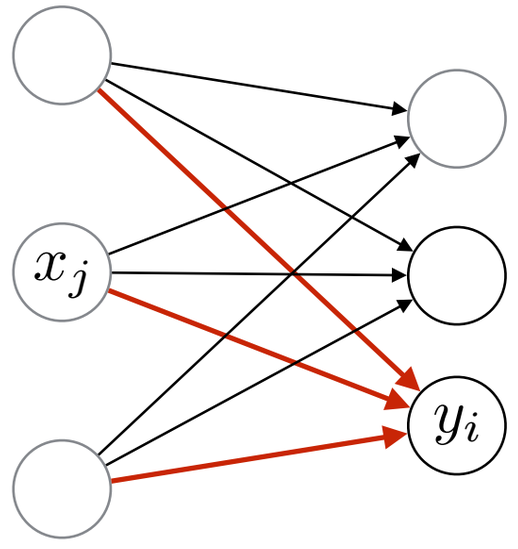
$\min_W L(W)$ s.t. $R(W) \leq s$

  - Does not makes weights small,

    but prevents them from growing too high
  - Can use projected SGD to solve
  - In particular L_2 norm on each column: $\forall i \quad \sum_j W_{ij}^2 \leq s$

    called **max-norm** appears useful

$y_i = \sum_j W_{ij} x_j$

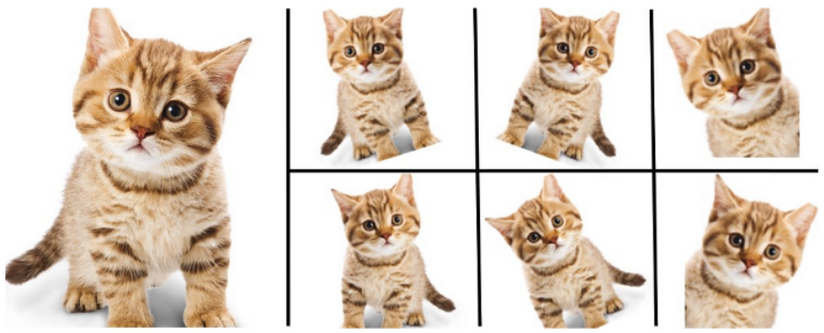


Data Augmentation

Data Augmentation (Image Classification)

✦ **Random transforms:**

- the transformed input should be as likely in the data distribution
- the class label should stay the same
- original image should be kept with sufficient probability



Geometric: scale, crop, rotation, non-rigid

Photometric: brightness, contrast

Filters: blur, sharpen, low-pass

Simulation: acquisition noise, fully synthetic

✦ **Pros:**

- Can improve generalization
- Can improve model robustness / enforce invariances of features

✦ **Cons:**

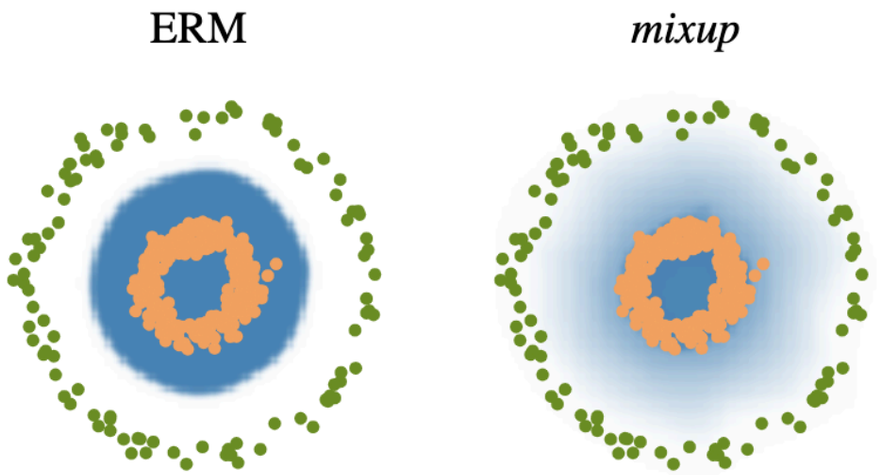
- altering the true data distribution too much could worsen performance (e.g. too much noise, synthetic-real gap)
- training could be slower

Data Augmentation (Image Classification)

- **Mixup** (Zhang et al .2018)**:**

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \qquad \text{where } x_i, x_j \text{ are raw input vectors}$$
$$\tilde{y} = \lambda y_i + (1-\lambda)y_j, \qquad \text{where } y_i, y_j \text{ are one-hot label encodings}$$

Inductive bias towards linear predictors -- smoother confidences and decision boundaries



Dataset	Model	ERM	*mixup*
CIFAR-10	PreAct ResNet-18	5.6	**4.2**
	WideResNet-28-10	3.8	**2.7**
	DenseNet-BC-190	3.7	**2.7**
CIFAR-100	PreAct ResNet-18	25.6	**21.1**
	WideResNet-28-10	19.4	**17.5**
	DenseNet-BC-190	19.0	**16.8**

Powerful idea to augment an image: use other images (labelled or not)

($\star$) Cross-entropy is linear in the target $y \Rightarrow$ reduces to regular data augmentation

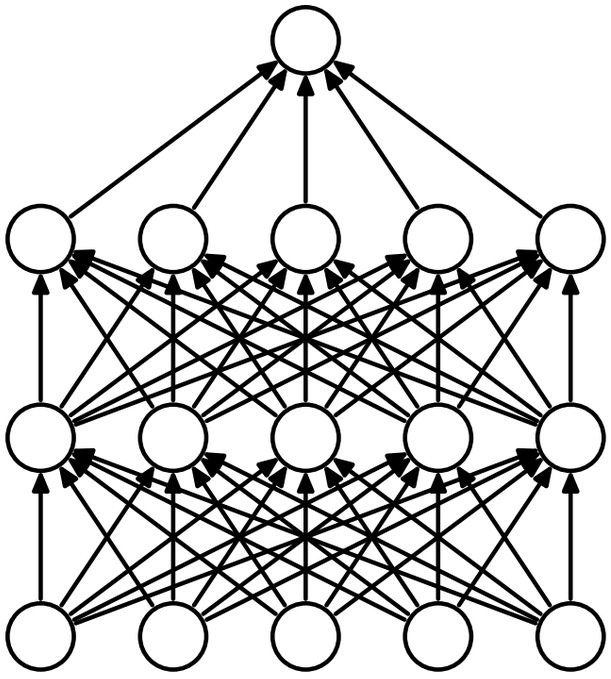
Injected Noises / Dropout

✦ **Injected Noises:**

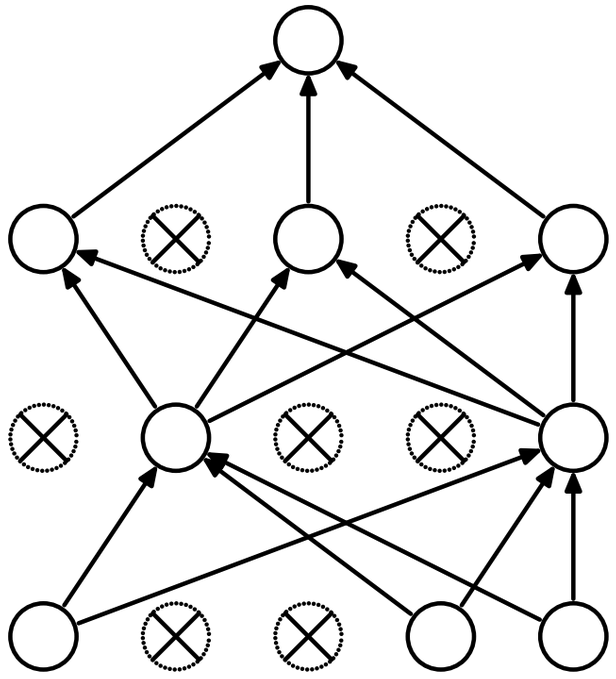
- input
- deep features
- parameters
- gradient updates

Bayesian learning, robust local minima

Dropout



(a) Standard Neural Net



(b) After applying dropout.

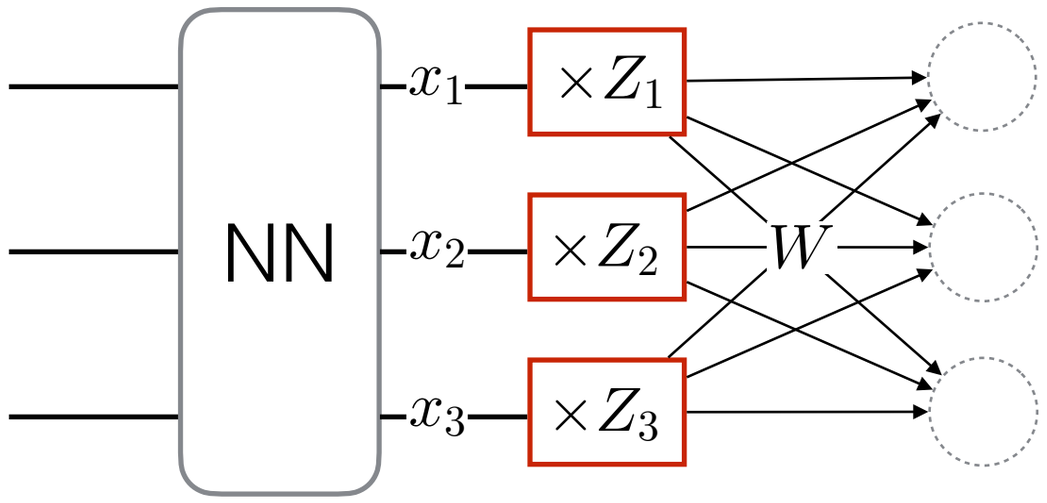
[Hinton et al. (2012) Improving neural networks by preventing co-adaptation of feature detectors]

[Srivastava et al. (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting]

- During training:
  - Randomly, "drop" some neurons -- set their outputs to zero
  - This results in the associated weights not being used and we obtain a (random) subnetwork
  - When learning, the network develops robustness to units being dropped
- During testing:
  - Use all units -- ensemble of random subnetworks trained on differently sampled data

Mathematical Model

- What does it mean mathematically?
  - Introduce random Bernoulli variables $Z_i = \begin{cases} 1, & \text{with probability } p, \\ 0, & \text{with probability } 1-p, \end{cases}$ multiplying outputs of the preceding layer
  - Can interpret outputs multiplied with 0 as dropped
  - Drop probability $q = 1 - p$
  - Next layer activations: $a = W(x \odot Z)$



$Z_i \sim \text{Bernoulli}(0.3)$

- Prediction is random now?
  - Denote the network output as $f(x, Z; \theta)$
  - We have two choices how to make predictions:
    - **Randomized predictor:** $p(y|x, Z) = f(x, Z; \theta)$
    - **Ensemble:** $p(y|x) = \mathbb{E}_Z[f(x, Z; \theta)] = \sum_z p(z) f(x, z; \theta)$
- We use randomized predictor for training (easier)
- We use ensemble (or its approximation) for testing

Note: Gaussian multiplicative $\mathcal{N}(1, \sigma^2)$ noises work as well (Gaussian Dropout)

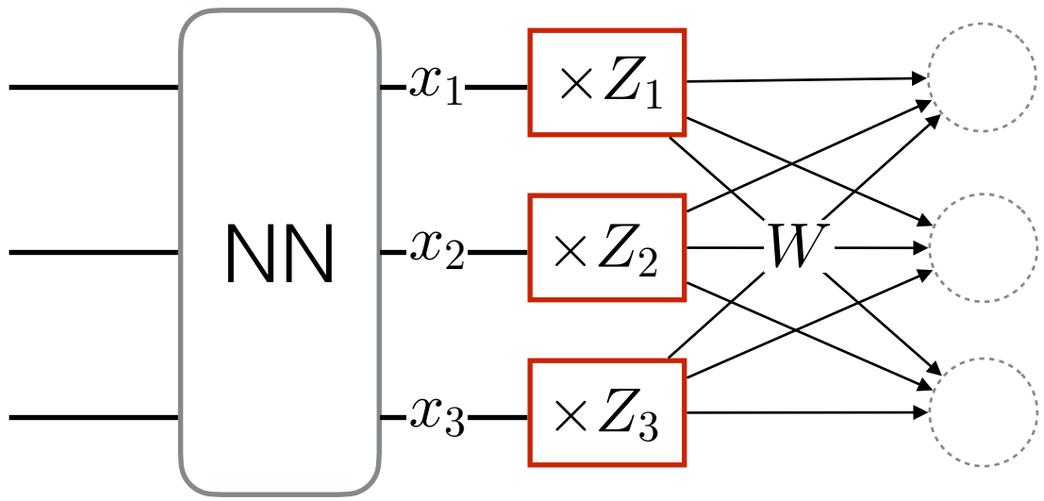
Training

- Expected loss of randomized predictor:
  - Double expectation in noises and data: $\mathbb{E}_Z\Big[\mathbb{E}_{(x,y)\sim\text{data}}\Big[l(y, f(x, Z; \theta))\Big]\Big]$
  - Same as: $\mathbb{E}_{Z\sim\text{Bernoulli}(q),\ (x,y)\sim\text{data}}\Big[l(y, f(x, Z; \theta))\Big]$
- What it means practically:
  - Draw a batch of data
  - For each data point i independently sample noises z_i
  - Unbiased loss estimate using a batch of size M:
    $\frac{1}{M}\sum_{i=1}^{M} l(y_i, f(x_i, z_i; \theta))$
  - Compute forward and backward pass
  - Will have increased variance of the stochastic gradient

Testing

- Use approximation (common default):
  - $\mathbb{E}_Z\Big[f(x, Z; \theta)\Big] \approx f(x, \mathbb{E}_Z[Z]; \theta)$
  - Since $\mathbb{E}_Z[Z] = p$, we have

    $a = W(x \odot \mathbb{E}[Z]) = (pW)x$
  - i.e. need to scale down the weights



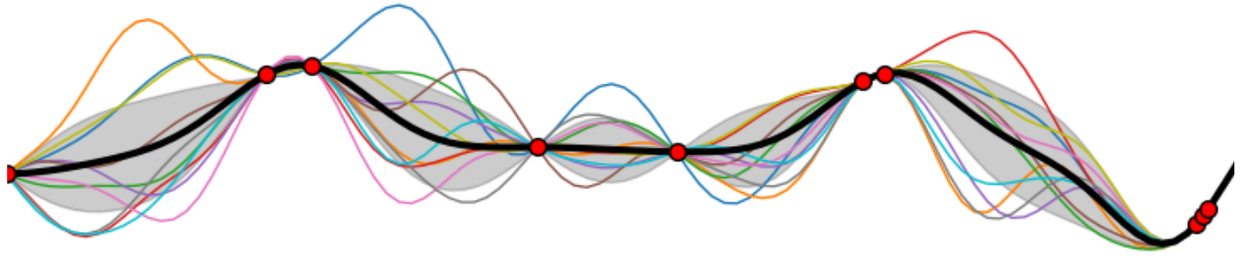
$Z_i \sim$ Bernoulli(0.3)

$E[Z] = p$

- Use sampling:
  - $\mathbb{E}_Z\Big[f(x, Z; \theta)\Big] \approx \frac{1}{M}\sum_{i=1}^{M} f(x_i, z_i; \theta)$
  - Generalizes slightly better than the above
  - Can be used to also estimate model uncertainty

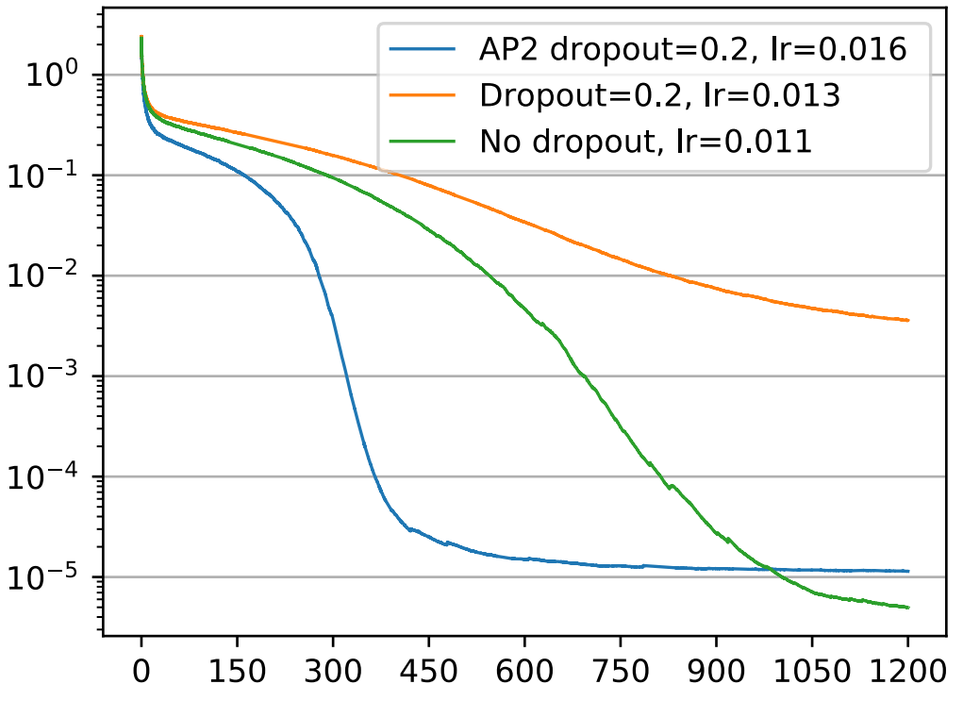
- Both variants achieve a "committee" or "ensembling" effect

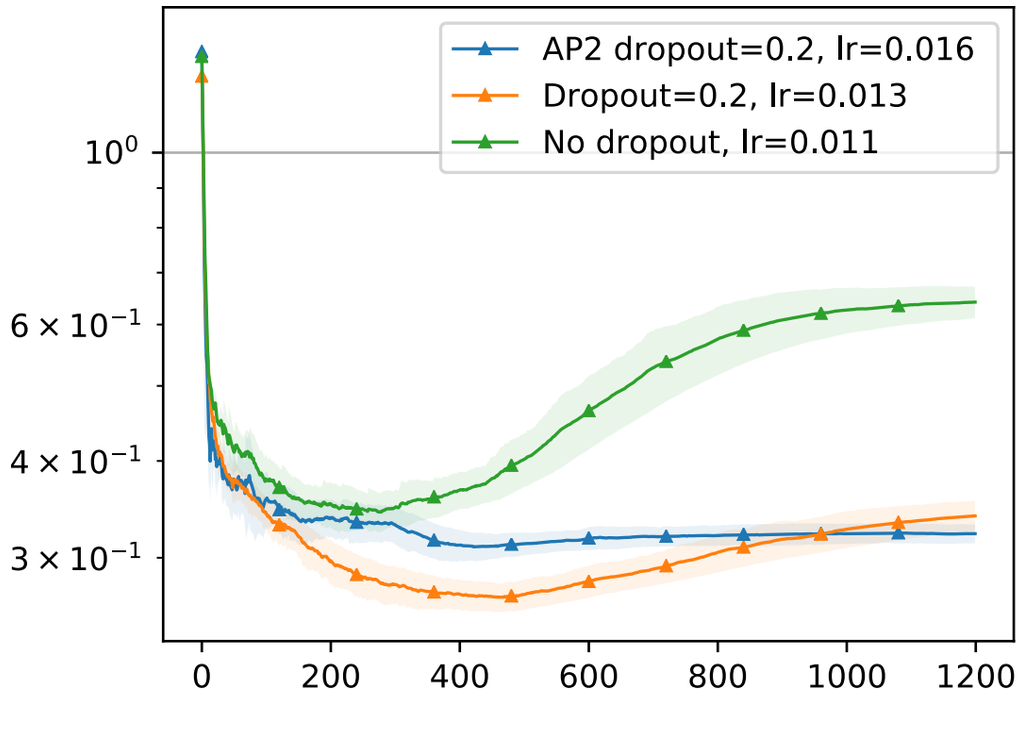
averaging of many well fitting models:



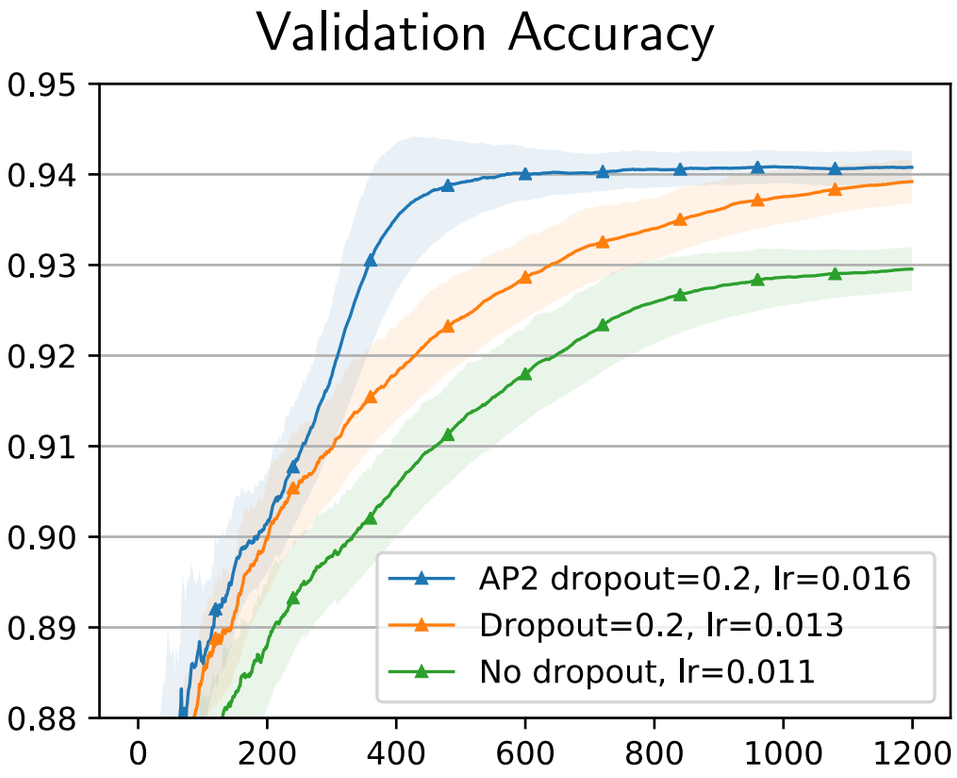
- More accurate analytic approximations than the first option are possible

CIFAR10 Example: Dropout



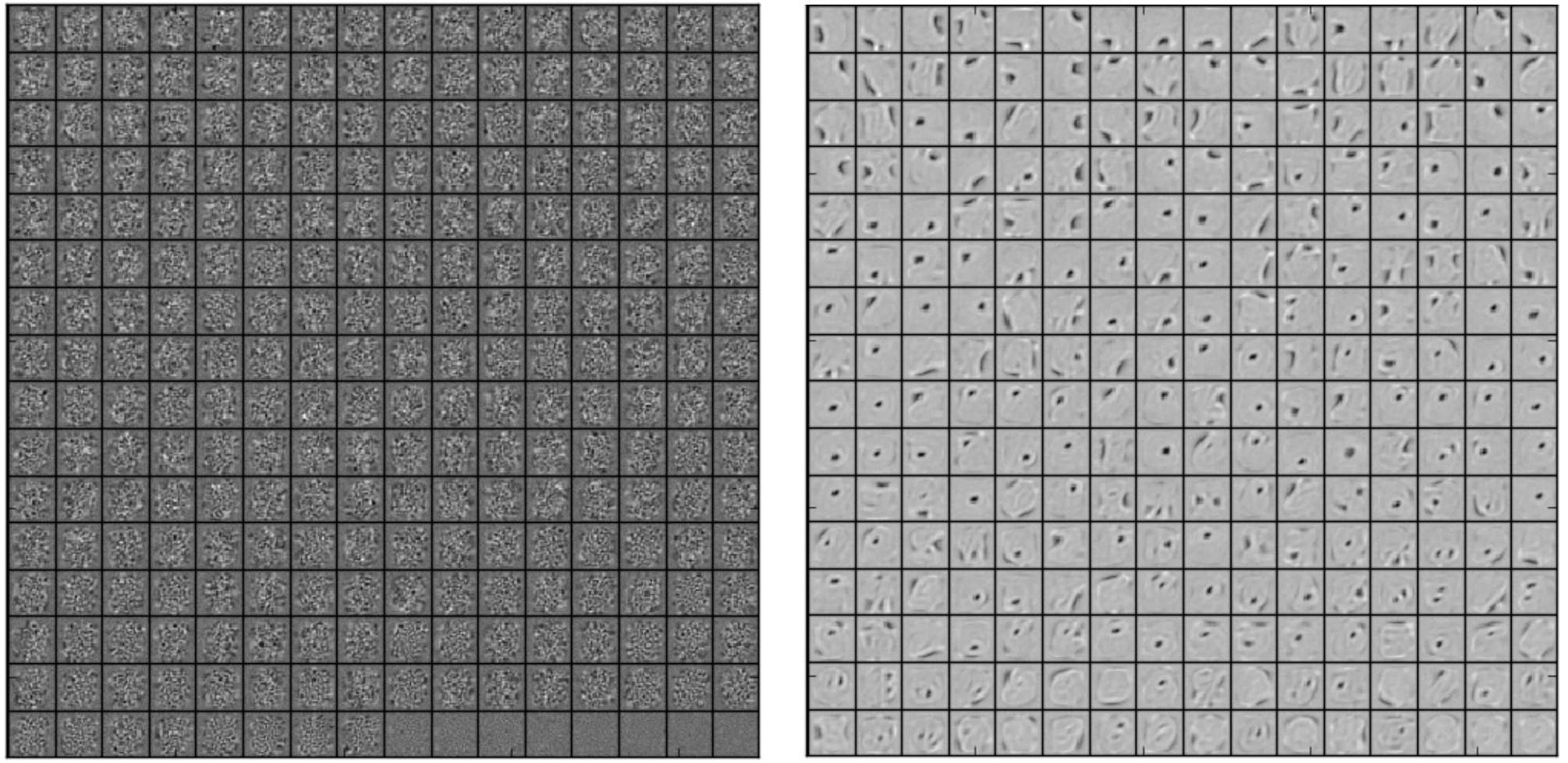


- Typically need to train longer due to higher gradient variance
  - There are techniques to approximate the effect analytically:
    Fast Dropout, Analytic Dropout (AP2)



Effect on Features

- Experiment:
  - MNIST auto encoder with 1 fully-connected hidden layer of 256 units



(a) Without dropout

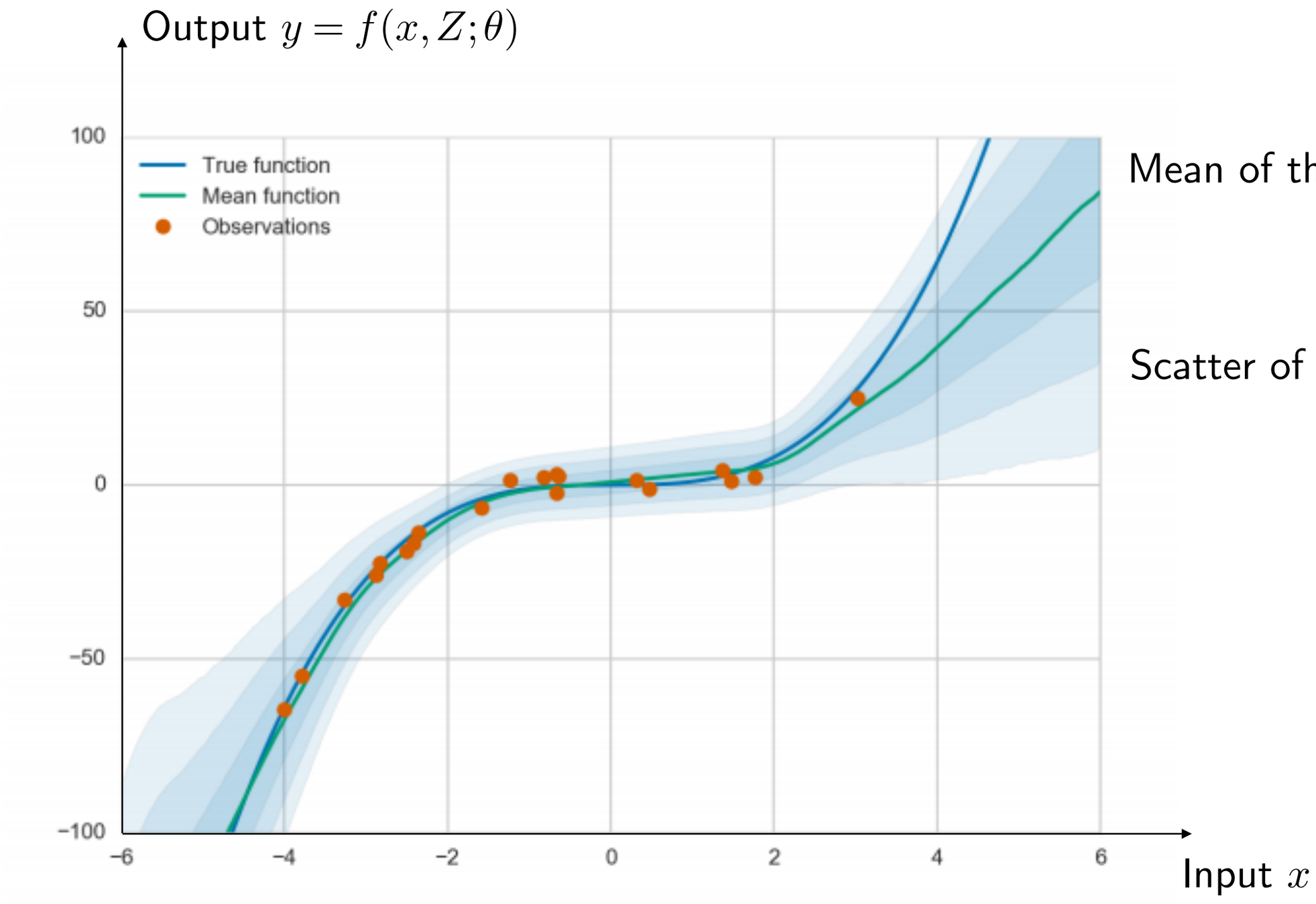
(b) Dropout with $p = 0.5$.

[Srivastava et al. (2014)]

- Hypothehis: dropout prevents co-adaptation and learns simpler features
- Further interesting studies in the paper: effect on activation sparsity, connection to ridge regression, etc.

Model Uncertainty with Dropout

[Louizos and Welling 2017]



Mean of the ensemble

Scatter of the ensemble

Implicit Regularization

CIFAR10 Example: BN

- BN has a strong regularization effect!
  - It depends on a randomly formed batch -> injecting specific structured noises
  - The normalization bends the parameter space -> different behavior of SGD