

Deep Learning (BEV033DLE)

Lecture 7. Regularization

Alexander Shekhovtsov

Czech Technical University in Prague

◆ 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](#) (L_1 , L_2 , general norms)
- [Data Augmentation](#) (random transforms, noises)
- Injected Noises ([Dropout](#))

◆ Implicit Regularization (Inductive Bias):

- Network Architecture (activations, structure, normalization, attention)
- Initialization, SGD dynamics, Batch Normalization
- Loss function, task formulation, transfer learning

Parameter Regularization

- ◆ 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
 - $\hat{\theta}_{\text{MAP}} = \underset{\theta}{\operatorname{argmax}} p(D|\theta)p(\theta) = \underset{\theta}{\operatorname{argmax}} \log p(D|\theta) + \log p(\theta)$

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

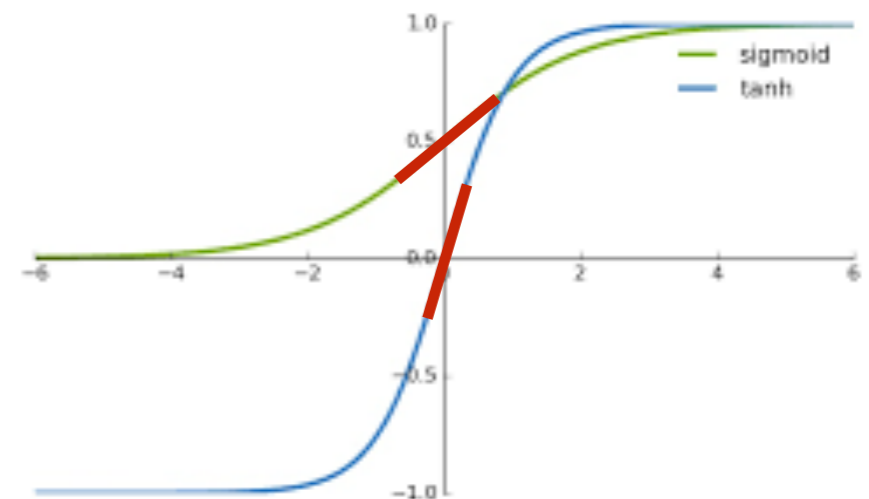
- ◆ In practice also commonly appears in the form independent of the amount of data:

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

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

L₂ Regularization

- ◆ L_2 -regularization: $R(\theta) = \frac{1}{2}\|\theta\|^2$
- ◆ **Linear Regression:**
 - Ridge regression: $\min_{\theta} \sum_i (\theta^\top x_i - y_i)^2 + \frac{\lambda}{2}\|\theta\|^2$
 - Solution: $\hat{\theta} = (X^\top X + \lambda I)^{-1} X^\top y$
 - Assuming standardized data ($\sum_i x_{i,k} = 0$, $\sum_i x_{i,k}^2 = 1$) equivalent to *multiplicative noise* on inputs: $\min_{\theta} \mathbb{E}_{\xi \sim \mathcal{N}(1, \frac{\lambda}{2}I)} \sum_i (\langle \theta, \xi \odot x_i \rangle - y_i)^2$
 - Even small λ helps with underdetermined problems
- ◆ **Linear Classifiers:**
 - SVMs maximize margin $\frac{1}{\|w\|}$ by minimizing $\|w\|^2$ under constraints
 - Generalization guarantees improve with larger margin
- ◆ **Sigmoid NNs:**
 - Small $\|\theta\| \Rightarrow$ sigmoid outputs are close to linear
 \Rightarrow smoother classification boundary

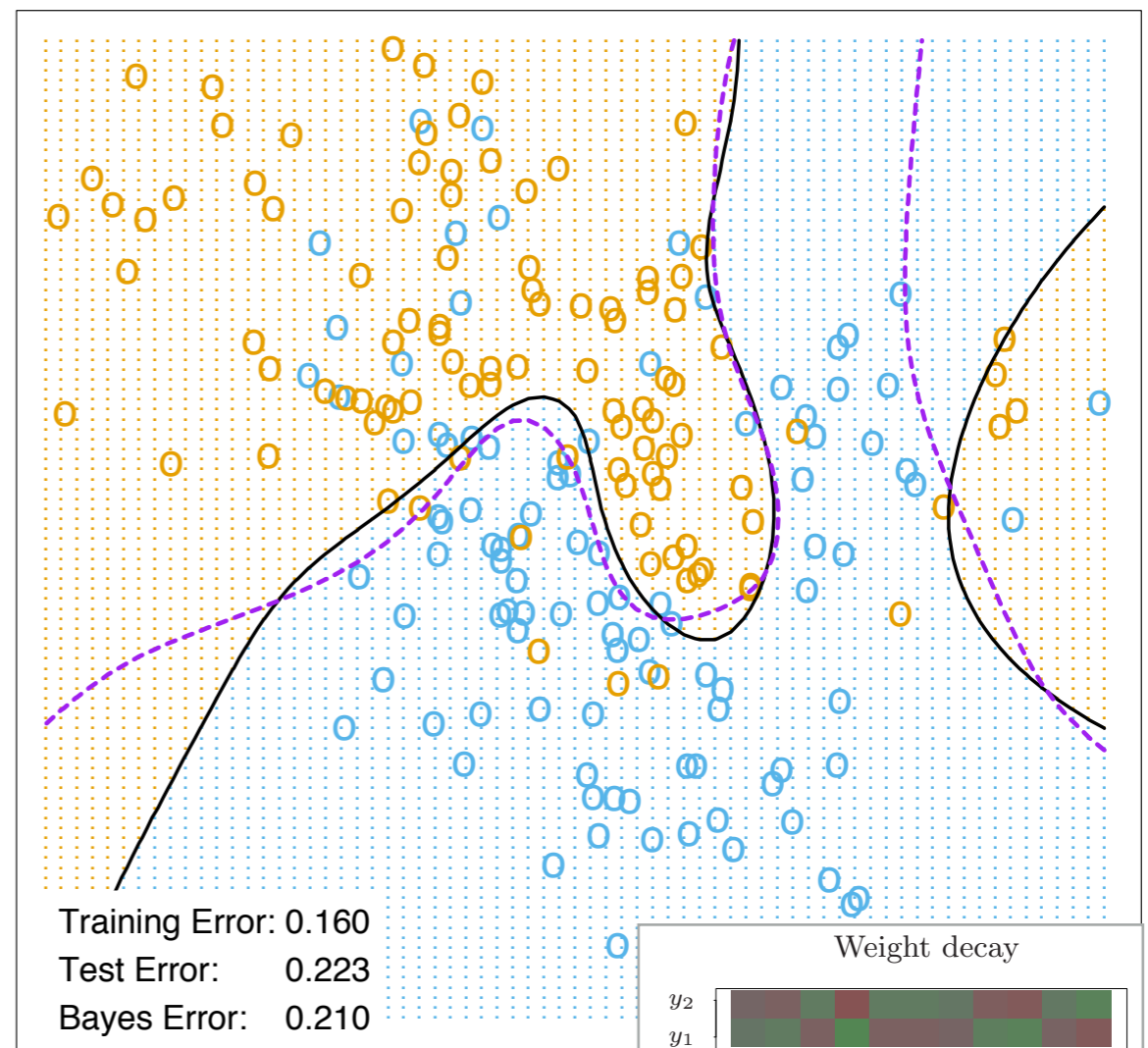
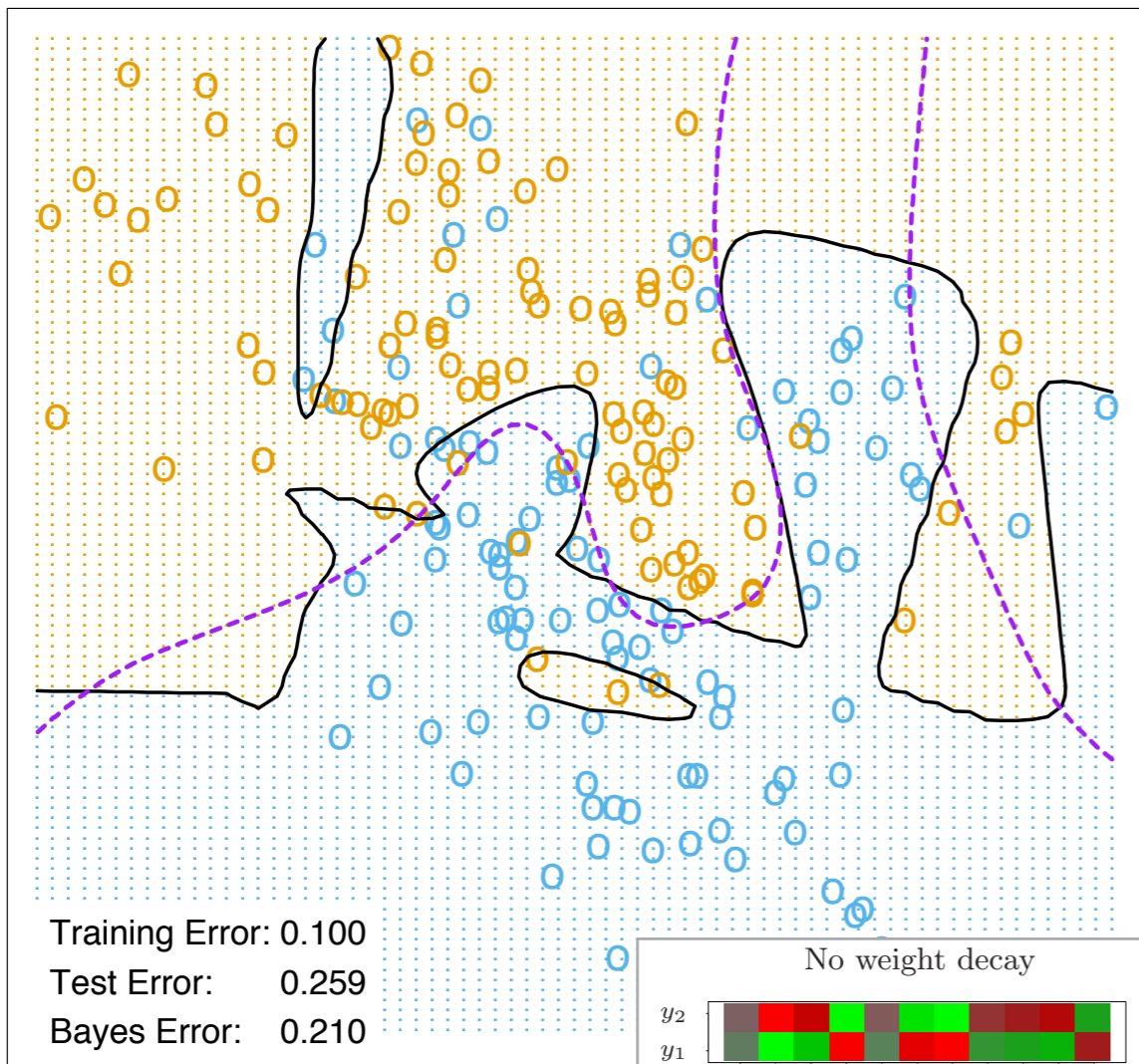


L₂ Regularization: Simulated Data Example



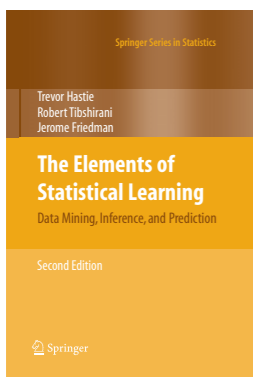
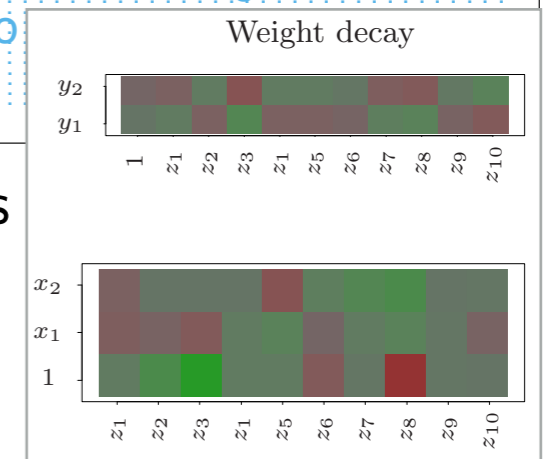
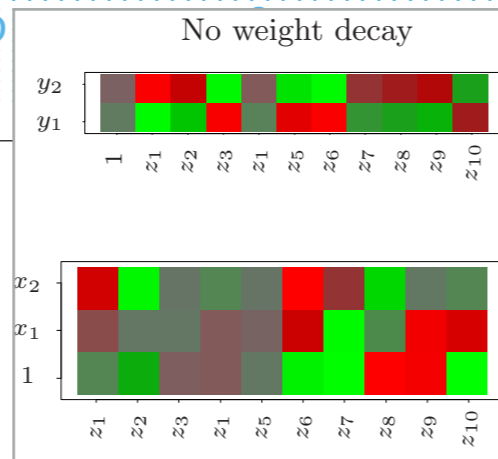
Neural Network - 10 Units, No Weight Decay

Neural Network - 10 Units, Weight Decay=0.02



weights

weights



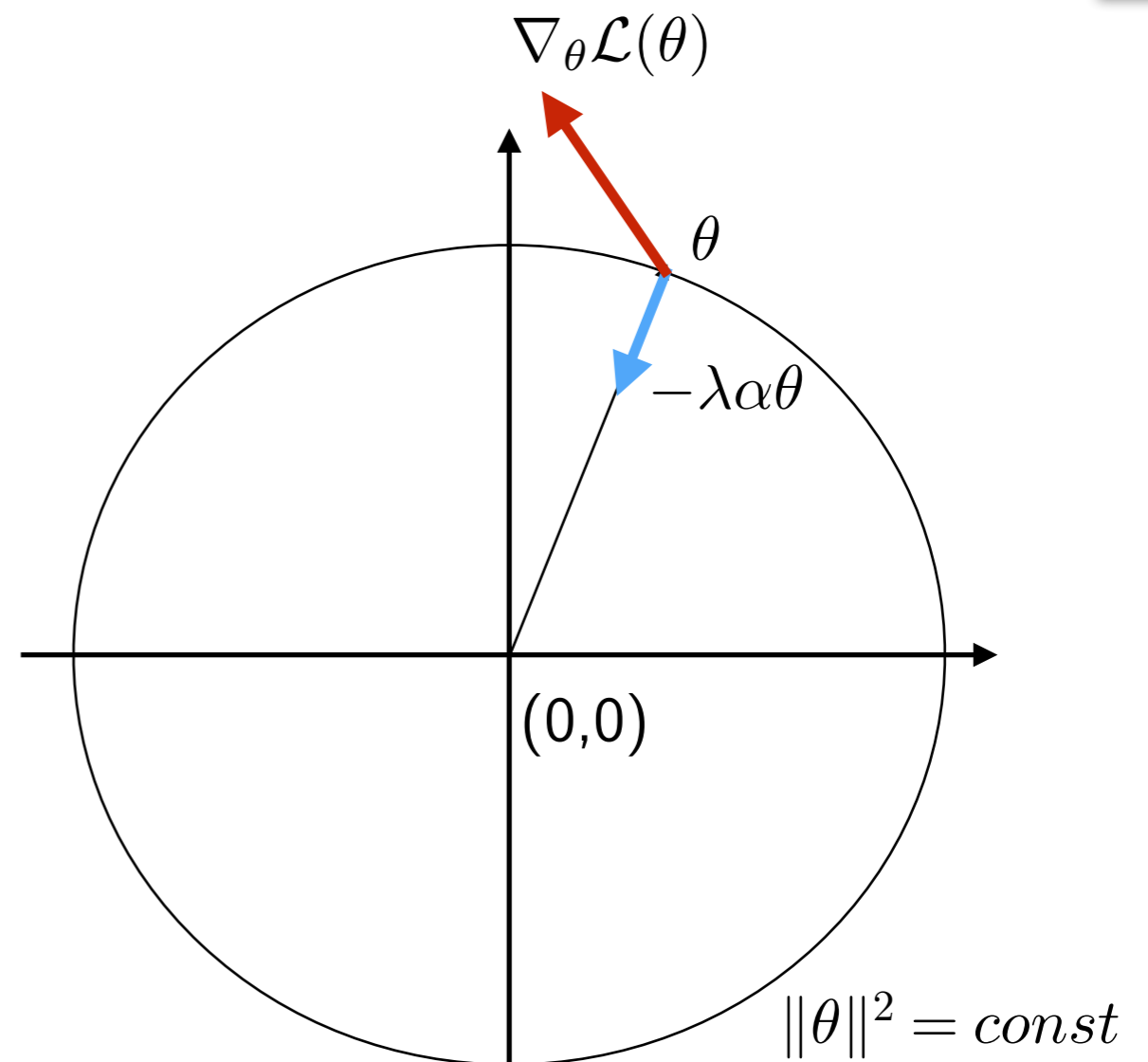
Hastie, Tibshirani and Friedman: The Elements of Statistical Learning

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

L₂ Regularization: Weight Decay

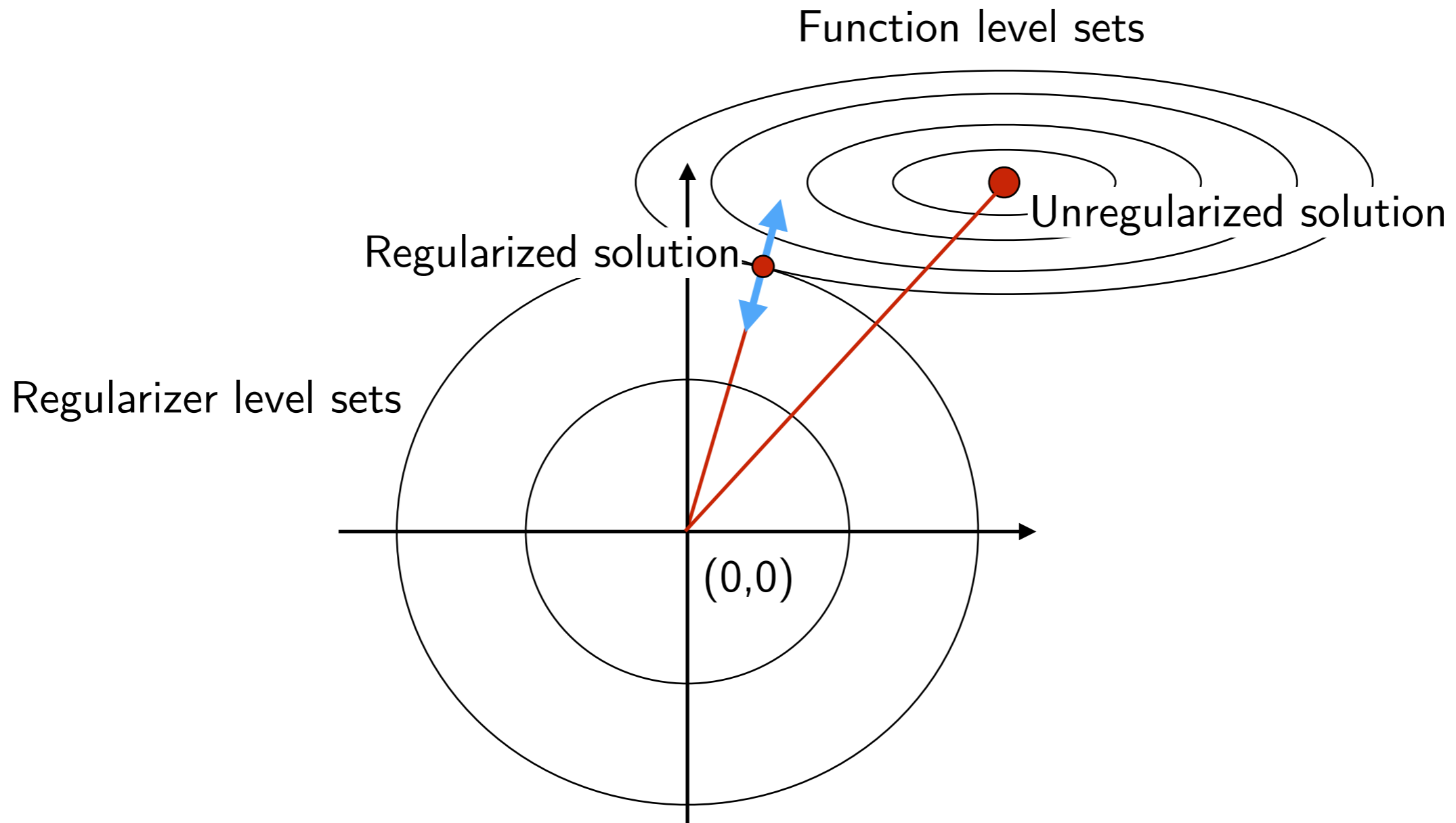


- ◆ $\min_{\theta} \mathcal{L}(\theta) + \frac{\lambda}{2} \|\theta\|^2$
- ◆ Gradient descent:
 - $g^t := \nabla_{\theta} \mathcal{L}(\theta) + \lambda \theta$
 - $\theta^{t+1} = \theta^t - \alpha g^t$
 - $\theta^{t+1} = \underbrace{(1 - \alpha \lambda)}_{\text{decay}} \theta^t - \alpha \nabla_{\theta}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



- ◆ In neural networks:
 - There is typically a manifold of optimal solutions, small regularization of order 10^{-5} may have effect

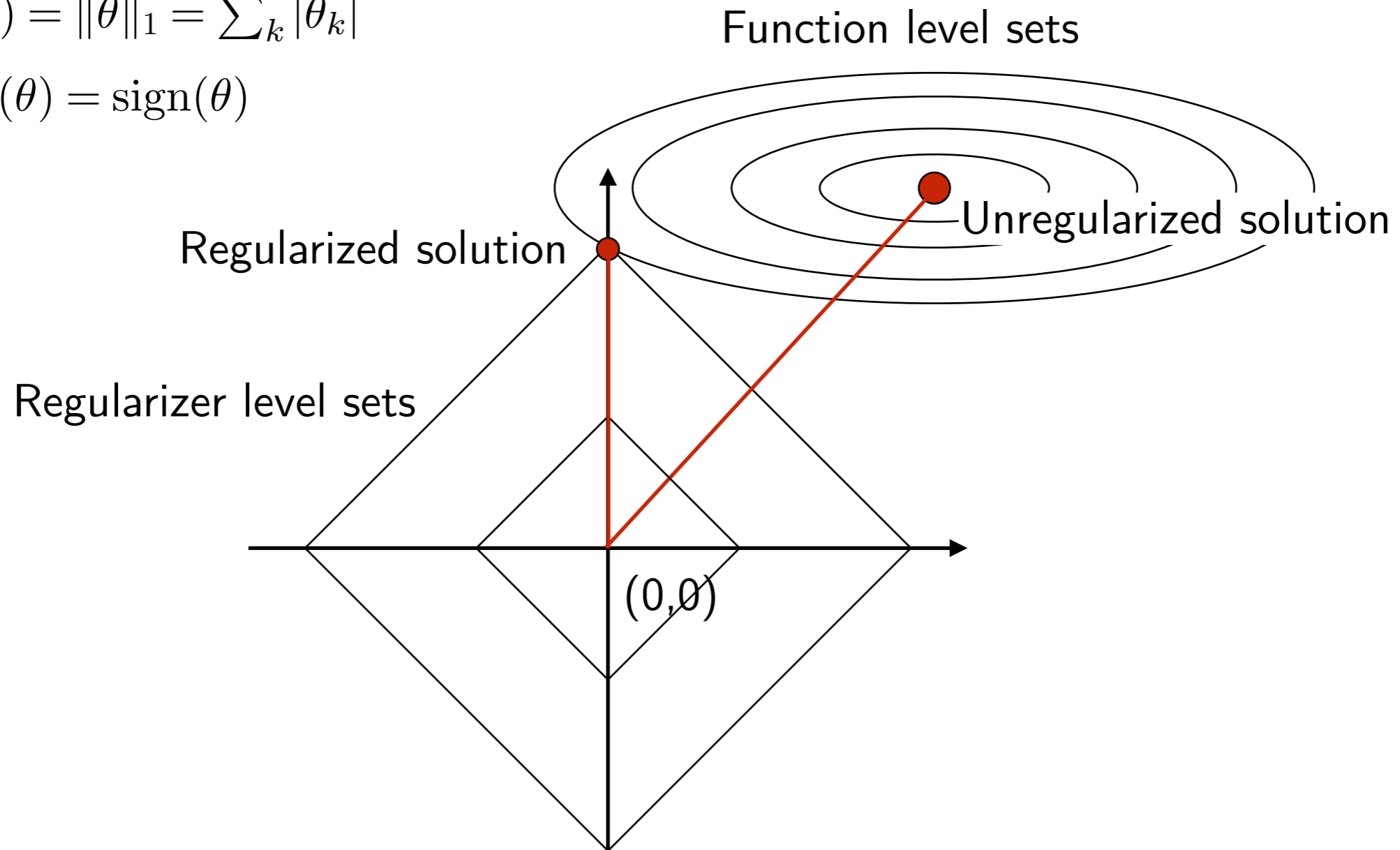
L2 Regularization



- ◆ 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)

◆ Binary classification

- Training set $\{x_i, y_i\}_{i=1}^N$ with $y_i \in \{1, -1\}$.

◆ Linear predictor

- Score: $s(x) = w^\top \phi(x)$,
- Decision: $y = \text{sign}(s)$,
- Regularized loss solution $w^*(\lambda) = \arg \min_w \sum_i l(y_i s(x_i)) + \lambda \|w\|_p^p$.

◆ Theorem [Rosset et al. 2003]:

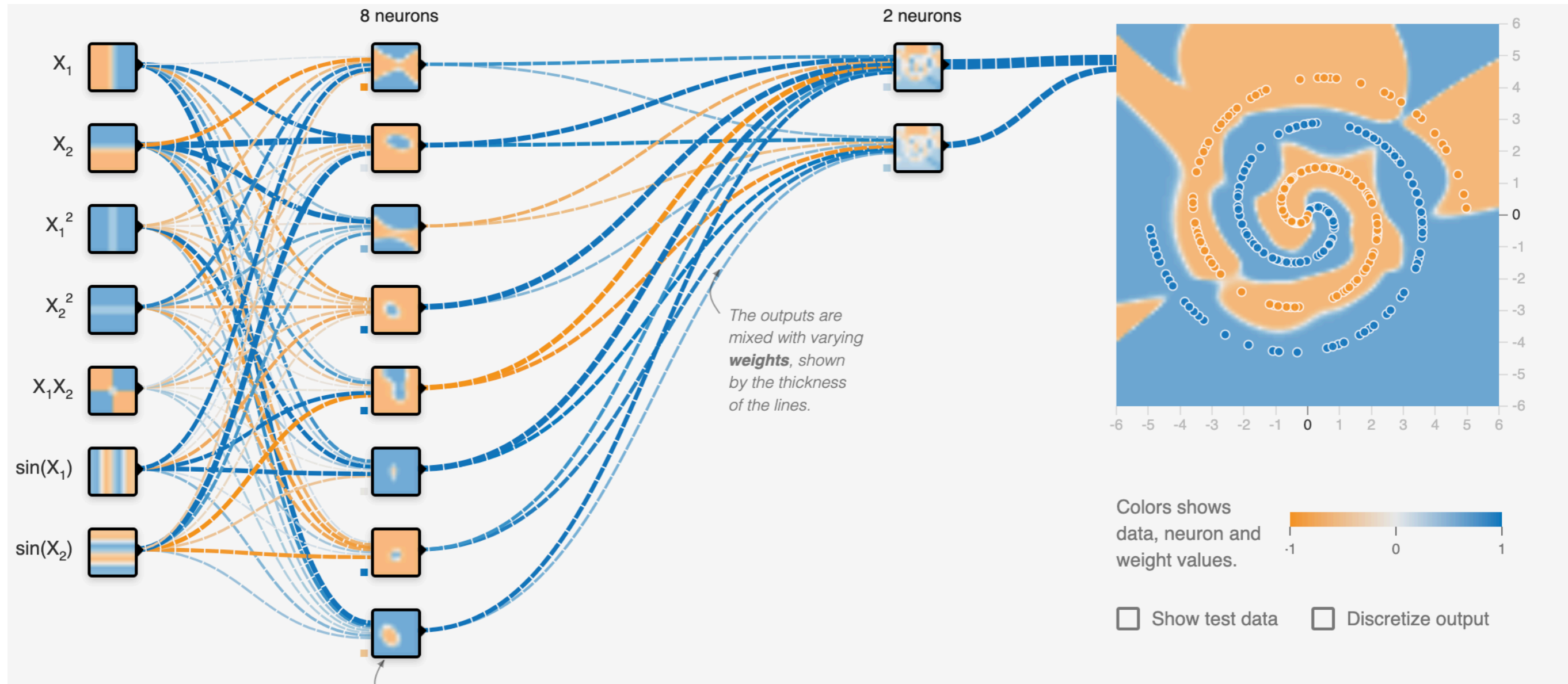
- Assume that data is linear separable and l is monotone and decreasing "quickly" (includes SVM hinge loss, logistic regression loss, AdaBoost exponential loss).
- Then

$$\lim_{\lambda \rightarrow 0} \|w^*(\lambda)\|_p = \infty,$$

$$\lim_{\lambda \rightarrow 0} \frac{w^*(\lambda)}{\|w^*(\lambda)\|_p} = w^M,$$

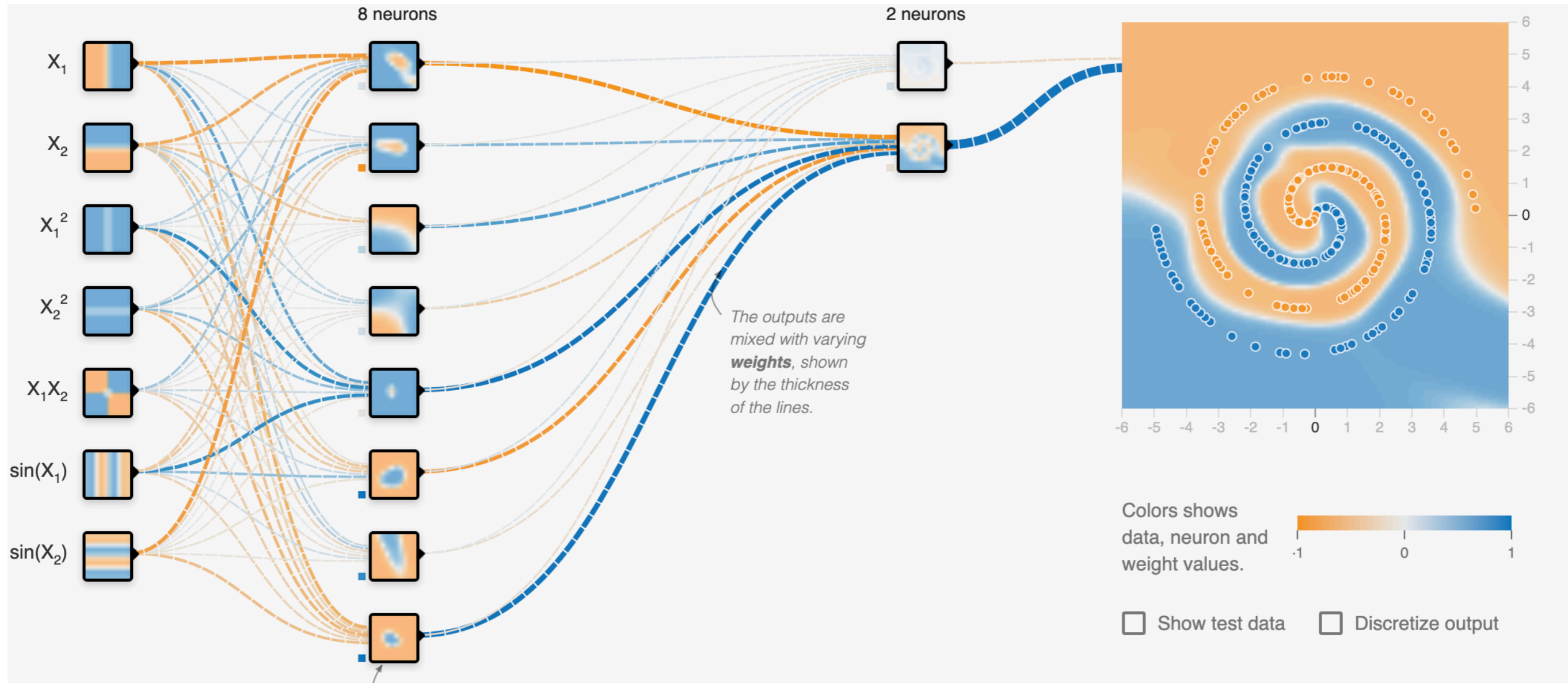
where $w^M = \arg \max_{\|w\|_p=1} \min_i y_i w^\top \phi(x)$ is the max-margin separating hyperplane.

Example: No regularization

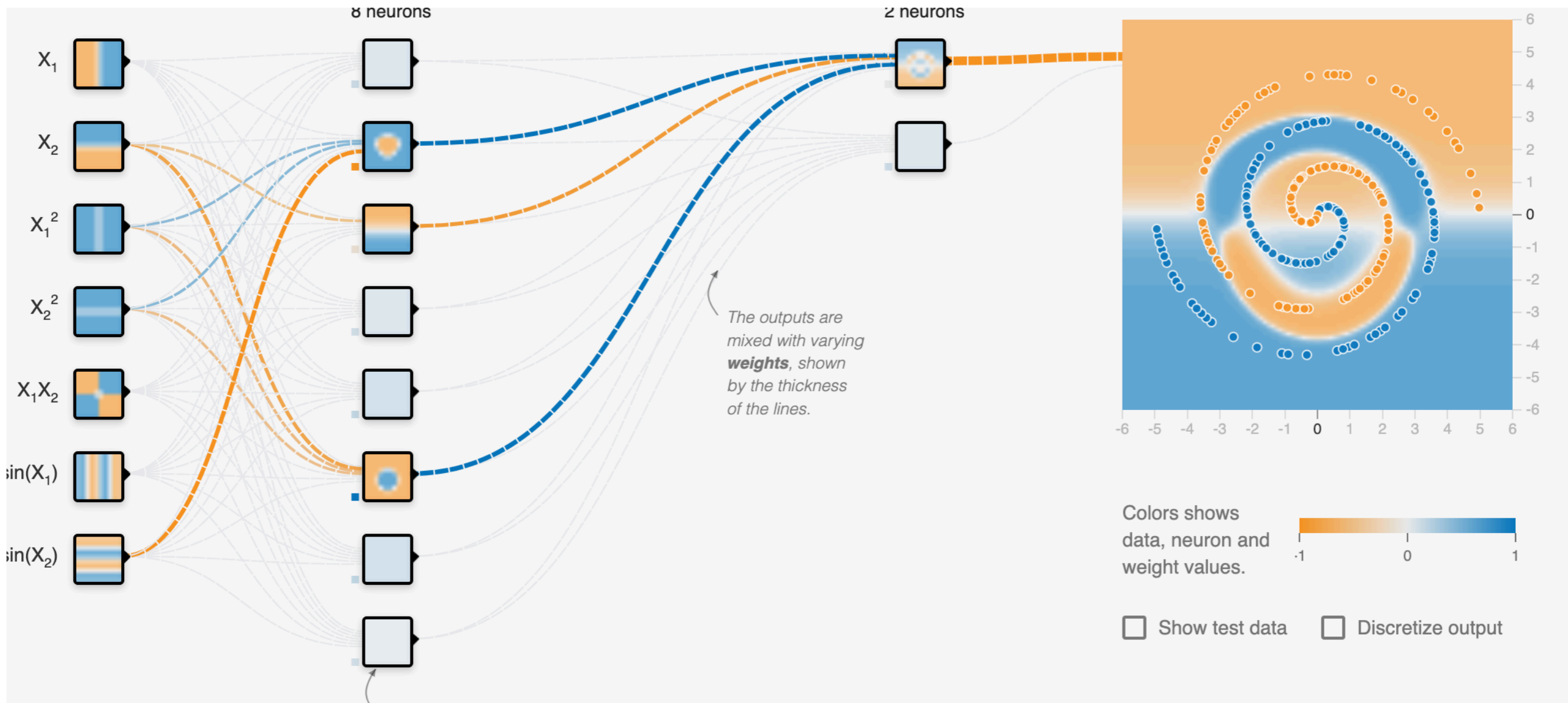


[Tensorflow playground]

Example: L₂ Regularization



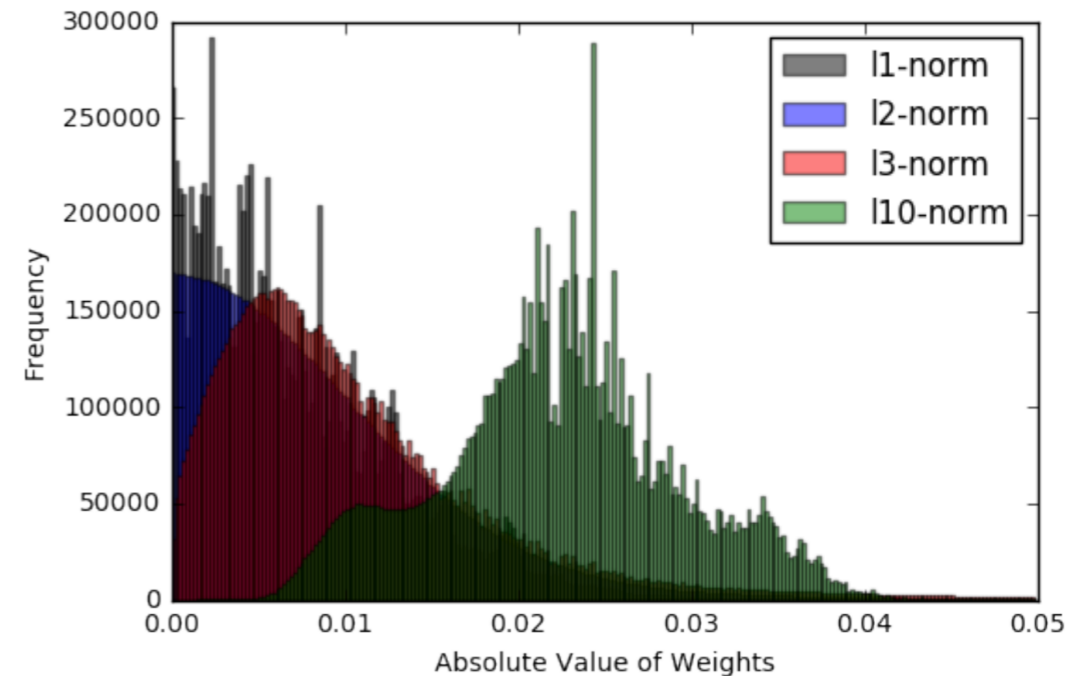
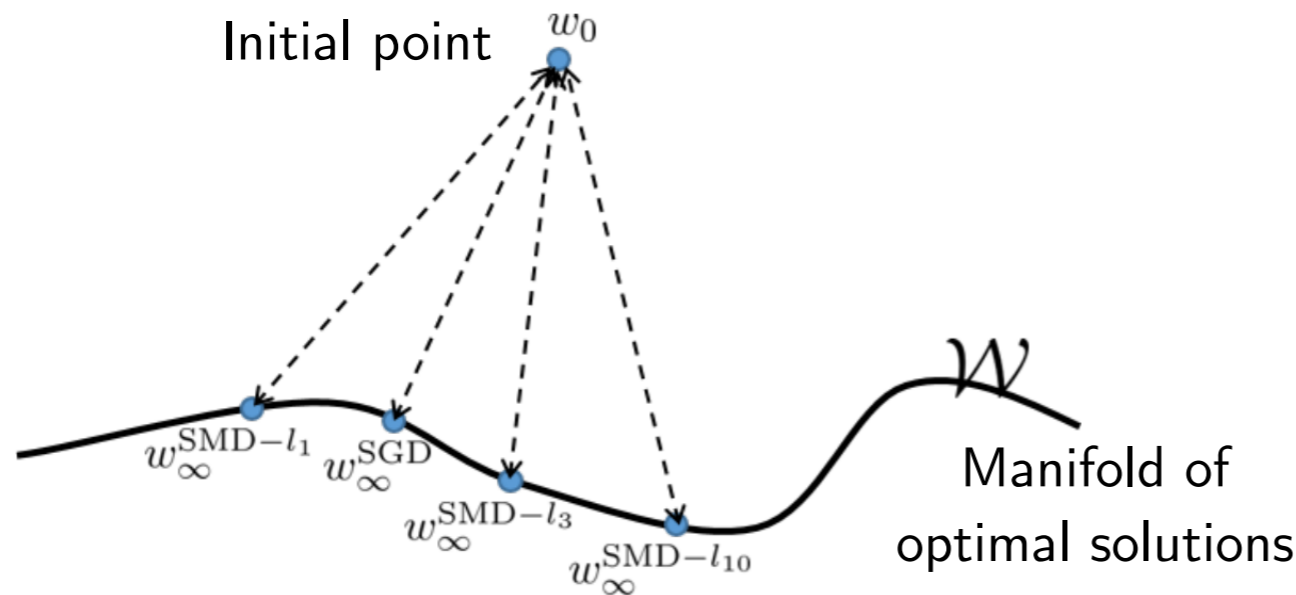
Example: L₁ Regularization



Implicit Regularization by p -Norm-Stepest SGD



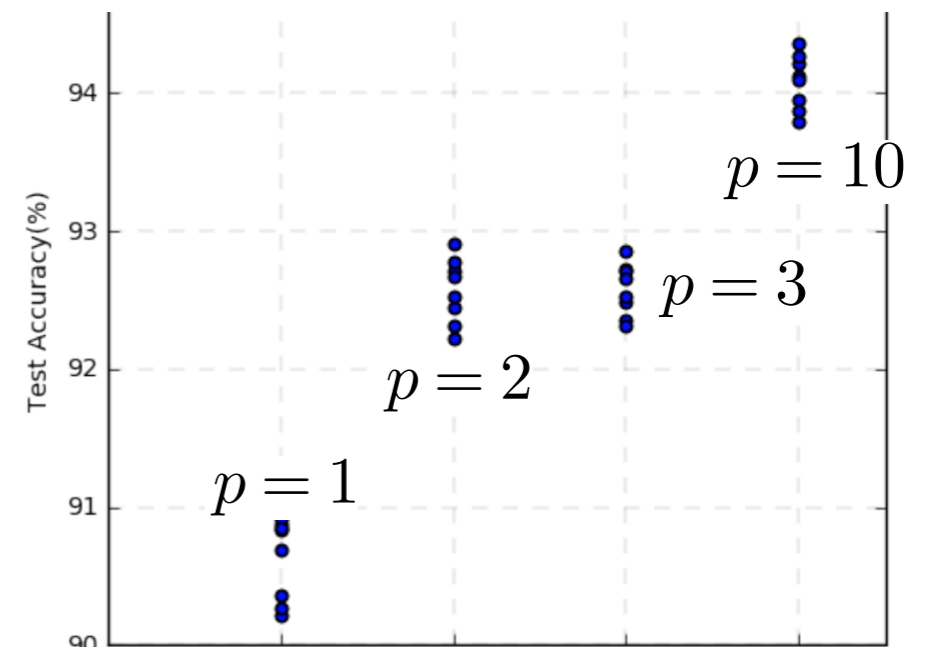
- ◆ Consider step proximal problem: $\min_x \langle \nabla f(x_0), x - x_0 \rangle + \lambda \|x - x_0\|_p^p$
 - i.e., p -norm SGD (mirror descent)
- ◆ Using different p leads to solutions with different properties



- Iterates tend to $\operatorname{argmin}_{w \in \mathcal{W}} \|w - w_0\|_p^p$, the closest point in the respective norm

	SMD 1-norm	SMD 2-norm (SGD)	SMD 3-norm	SMD 10-norm
1-norm BD	141	9.19×10^3	4.1×10^4	2.34×10^5
2-norm BD	3.15×10^3	562	1.24×10^3	6.89×10^3
3-norm BD	4.31×10^4	107	53.5	1.85×10^2
10-norm BD	6.83×10^{13}	972	7.91×10^{-5}	2.72×10^{-8}

- Different sparsity and generalization



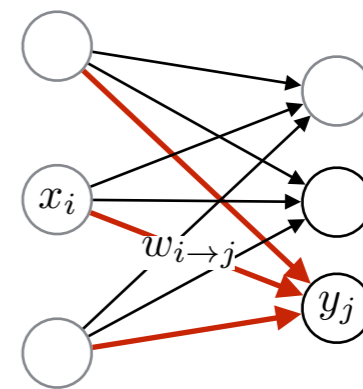
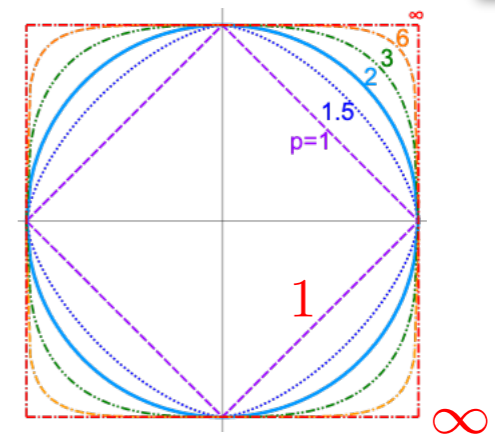
[Azizan et al. (2022):
Stochastic Mirror Descent on Overparameterized Nonlinear Models]

★ Group Norm, Generalization Bounds



◆ More General Norms:

- L_p norm: $\|\theta\|_p = \left(\sum_i |\theta_i|^p\right)^{\frac{1}{p}}$
- $p < 1$ is closer to counting non-zero weights, *i.e.* sparsity
- $p = \infty$ results in $R(\theta) = \max_i \theta_i$
- Hidden layer: $y_j = \sum_i w_{i \rightarrow j} x_i$
- $L_{p,q}$ norm: $\|W\|_{p,q} = \left(\sum_j \left(\sum_i |w_{i \rightarrow j}|^p\right)^{\frac{q}{p}}\right)^{\frac{1}{q}}$



◆ Generalization Bounds:

- Feed-forward NN with L layers, ReLU activations and weights W^k , $k = 1 \dots L$.
- Class of predictors \mathcal{F} defined by $\prod_{k=1}^L \|W^k\|_{p,q} \leq \psi$ for $q, p > 1$.
- **Theorem** [Neyshabur 2017] For any $\delta > 0$ with probability $1 - \delta$ over draws of training set \mathcal{T} of size N :

0-1 Risk Training risk on \mathcal{T} with margin γ Radamacher complexity of linear class $\propto \sqrt{\frac{1}{N}}$

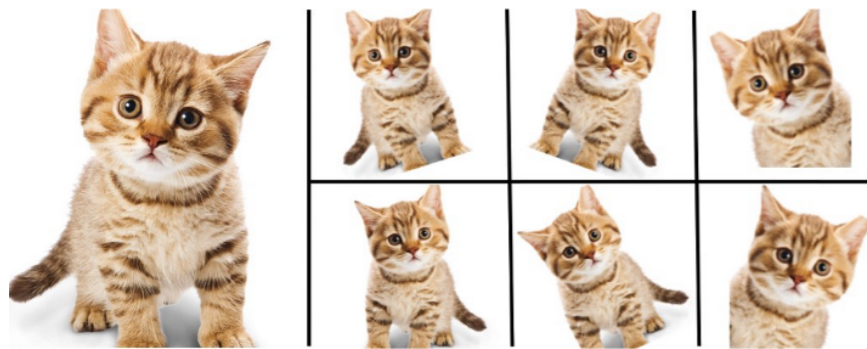
$$R \leq \hat{R}_{\mathcal{T}, \gamma} + \psi \frac{1}{\gamma} 2^L \left(H^{(L-1) \max(1 - \frac{1}{p} - \frac{1}{q}, 0)} C_{N, p, n_{in}}^{\text{linear}} + \sqrt{\frac{8 \ln(2/\delta)}{N}} \right)$$

Number of neurons Vanishes when $q \leq \frac{p}{p-1}$

Data Augmentation

◆ 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)
- increases training time

Data Augmentation (Image Classification)

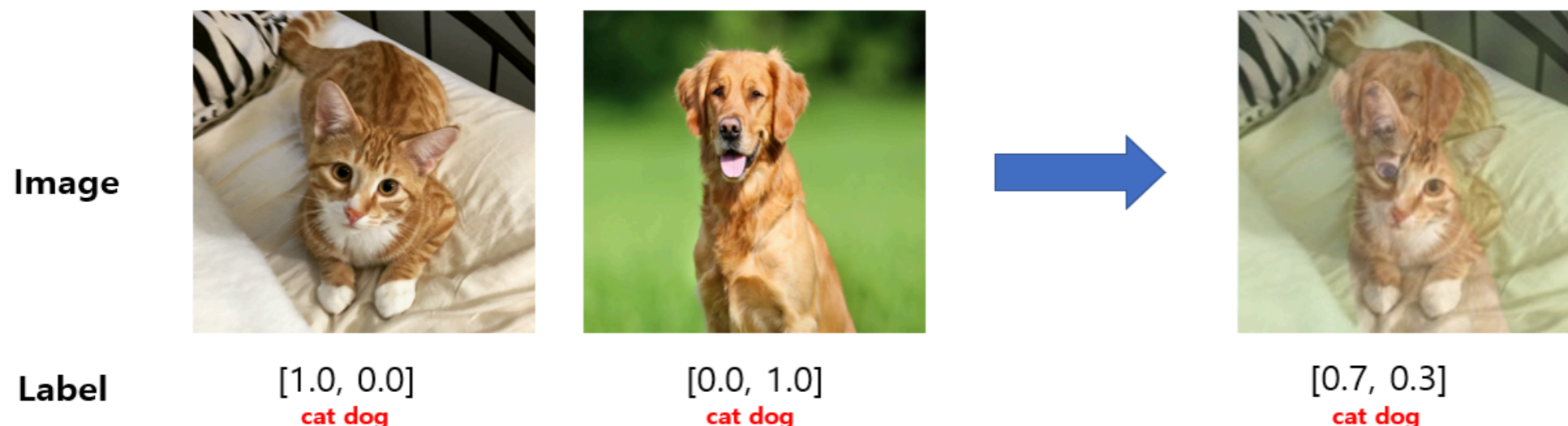
◆ **Mixup** (Zhang et al .2018):

$$\tilde{x} = \lambda x_i + (1 - \lambda)x_j,$$

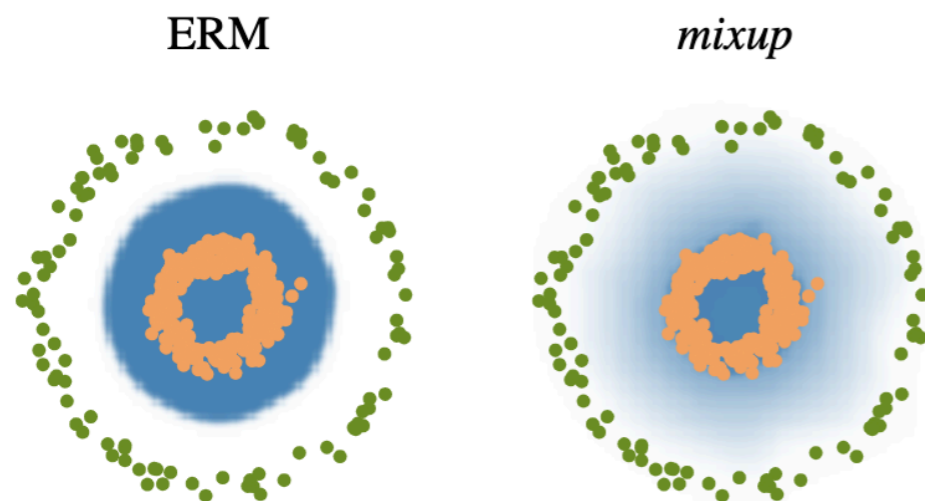
where x_i, x_j are raw input vectors

$$\tilde{y} = \lambda y_i + (1 - \lambda)y_j,$$

where y_i, y_j are one-hot label encodings



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



Dataset	Model	ERM	<i>mixup</i>
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)

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

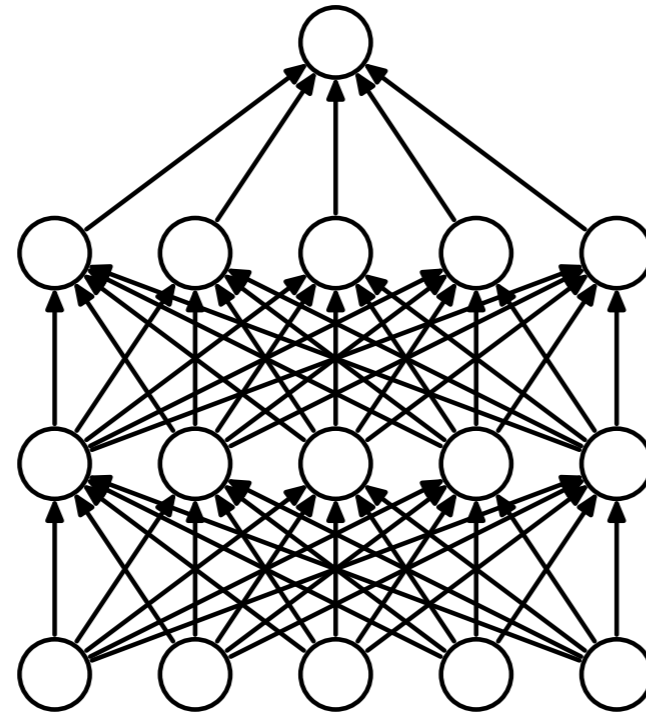
Injected Noises / Dropout

◆ Injected Noises:

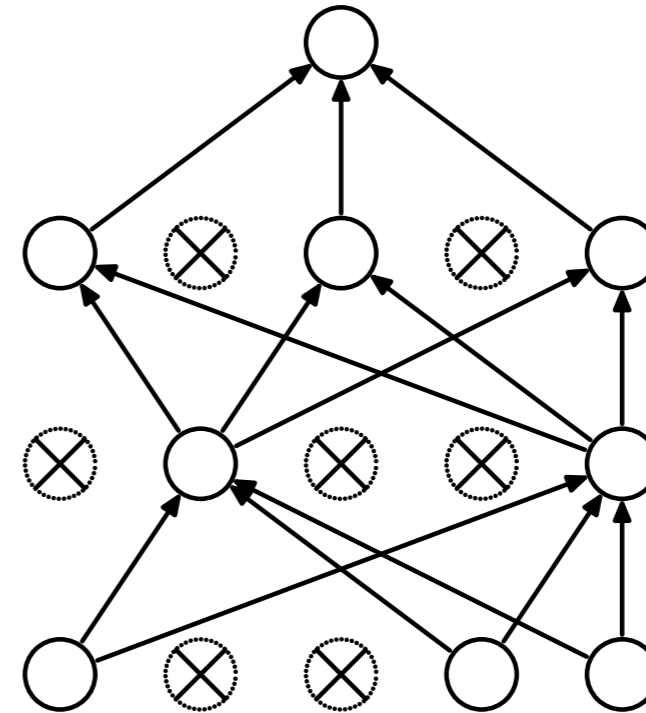
- input
- deep features
- parameters
- gradient updates

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

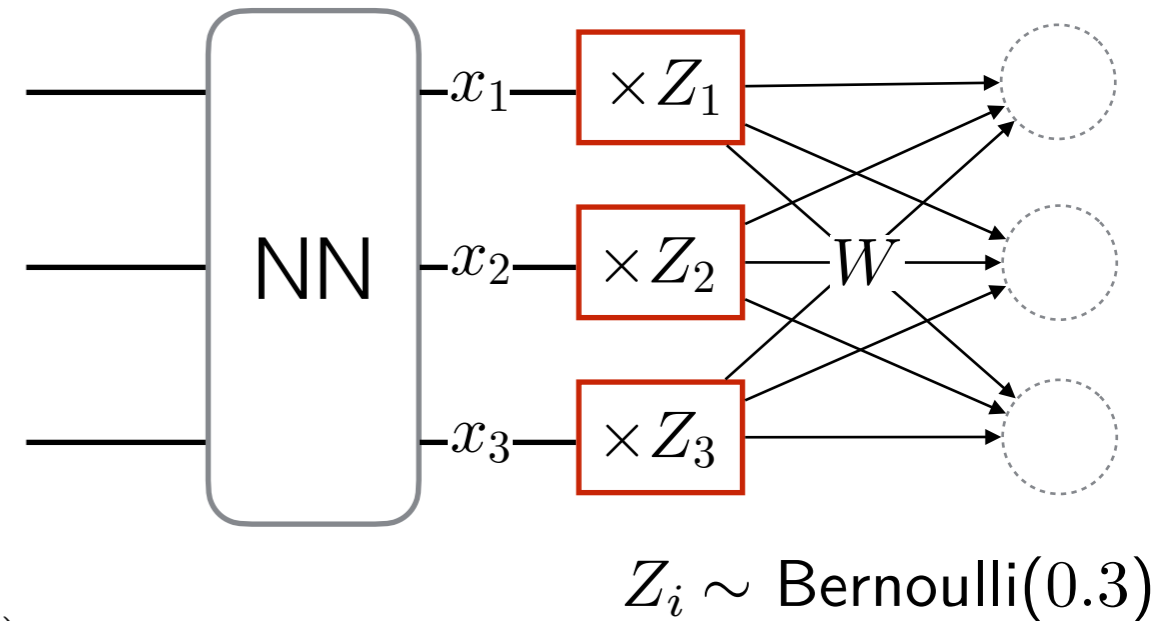
◆ During testing:

- Use all units -- approximates ensemble of all random subnetworks

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: $a = W(x \odot Z)$



◆ 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)

◆ Expected loss of randomized predictor:

- Double expectation in noises and data: $\mathbb{E}_Z \left[\mathbb{E}_{(x,y) \sim \text{data}} \left[l(y, f(x, Z; \theta)) \right] \right]$
- Same as: $\mathbb{E}_{Z \sim \text{Bernoulli}(q), (x,y) \sim \text{data}} \left[l(y, f(x, Z; \theta)) \right]$

◆ 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):

- $\mathbb{E}_Z [f(x, Z; \theta)] \approx f(x, \mathbb{E}_Z [Z]; \theta)$

- 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

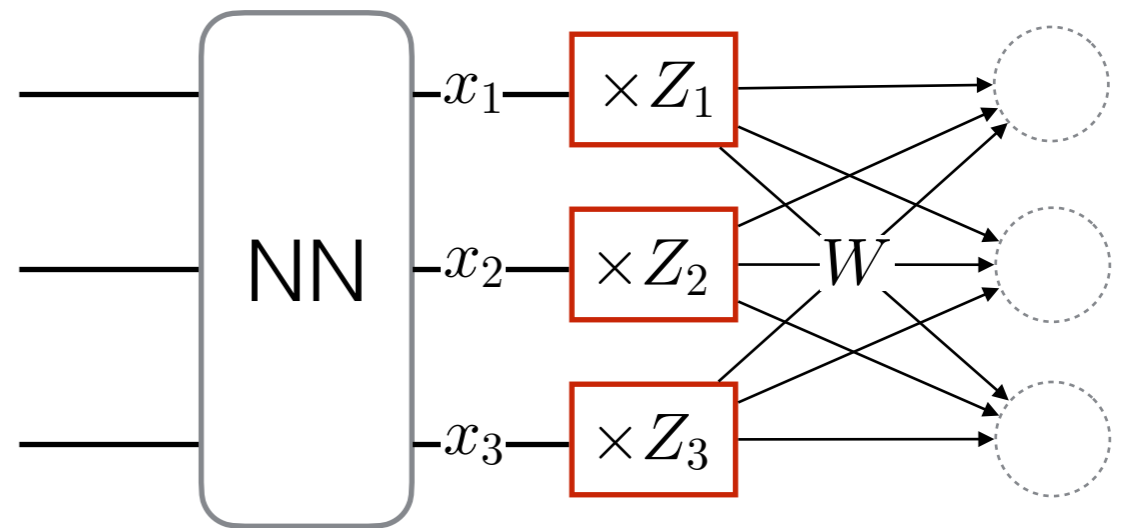
◆ Use sampling:

- $\mathbb{E}_Z [f(x, Z; \theta)] \approx \frac{1}{M} \sum_{i=1}^M f(x_i, z_i; \theta)$

- Generalizes slightly better than the above

- Can be used to also estimate model uncertainty

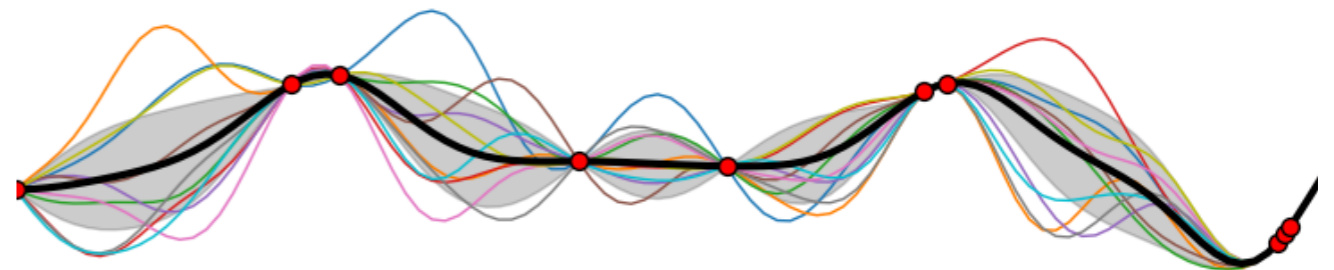
◆ Both variants achieve a "committee" or "ensembling" effect



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

$$E[Z] = p$$

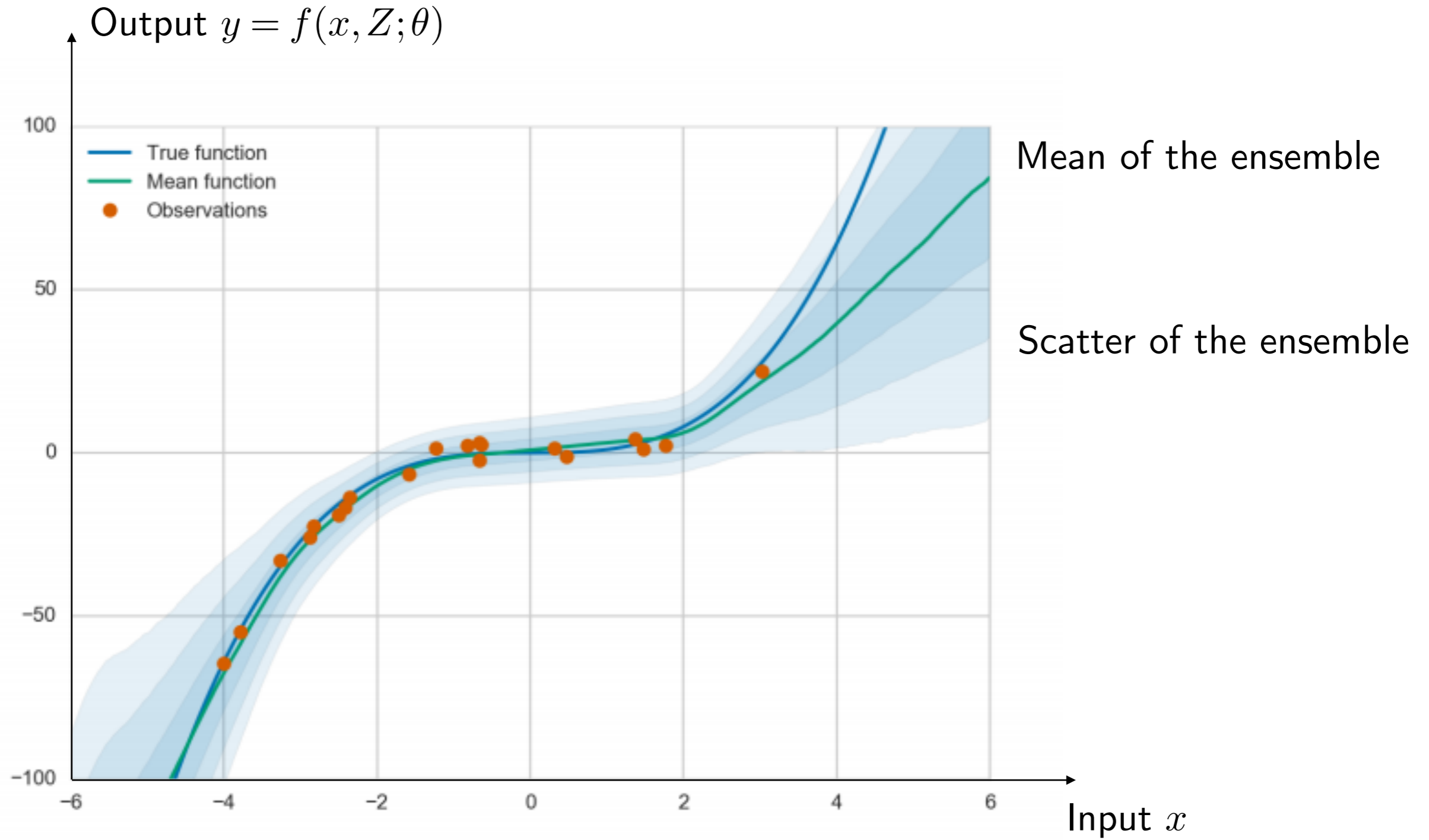
averaging of many well fitting models:



◆ More accurate analytic approximations than the first option are possible

Model Uncertainty with Dropout

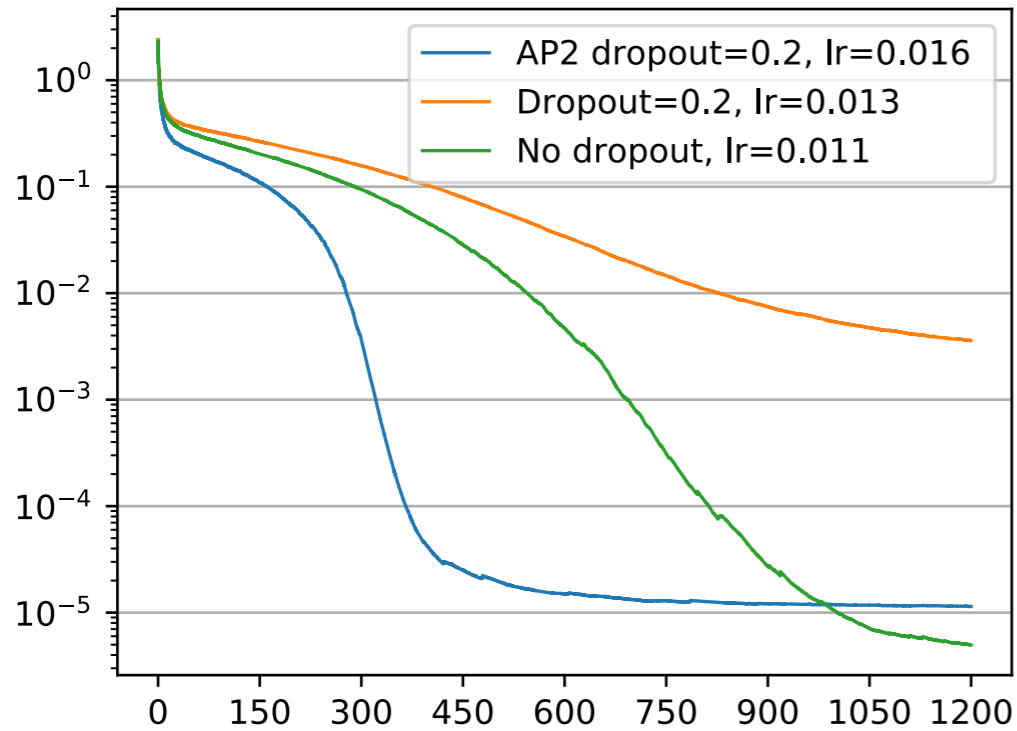
[Louizos and Welling 2017]



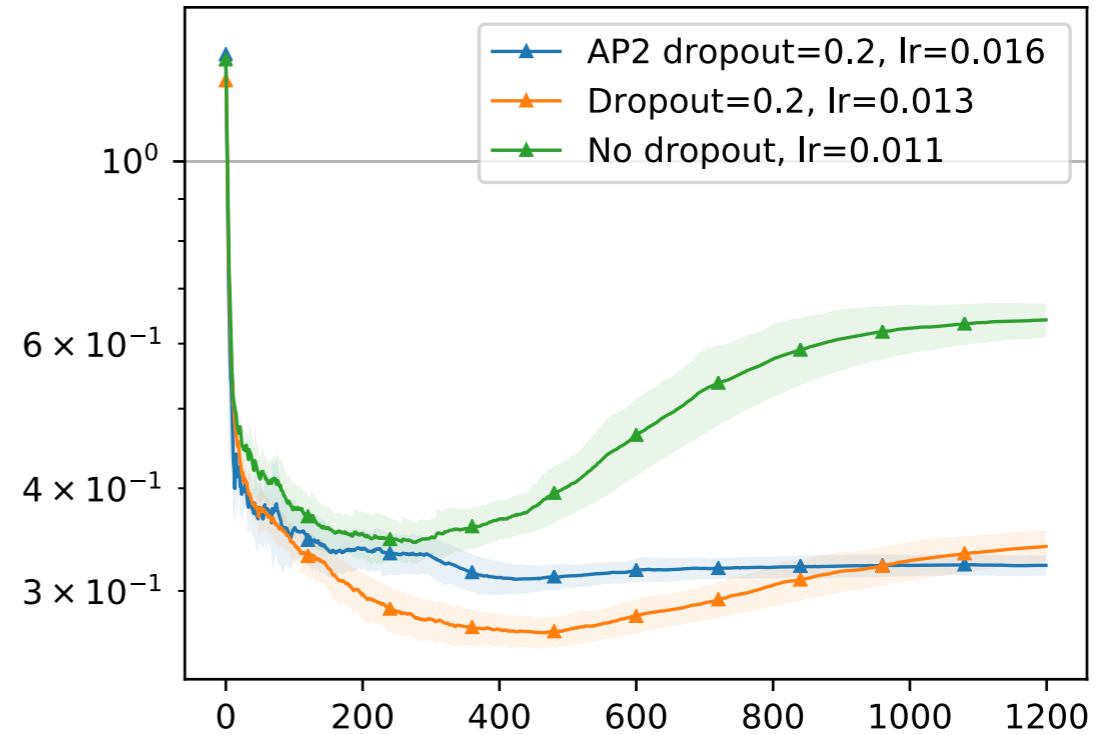
CIFAR10 Example: Dropout



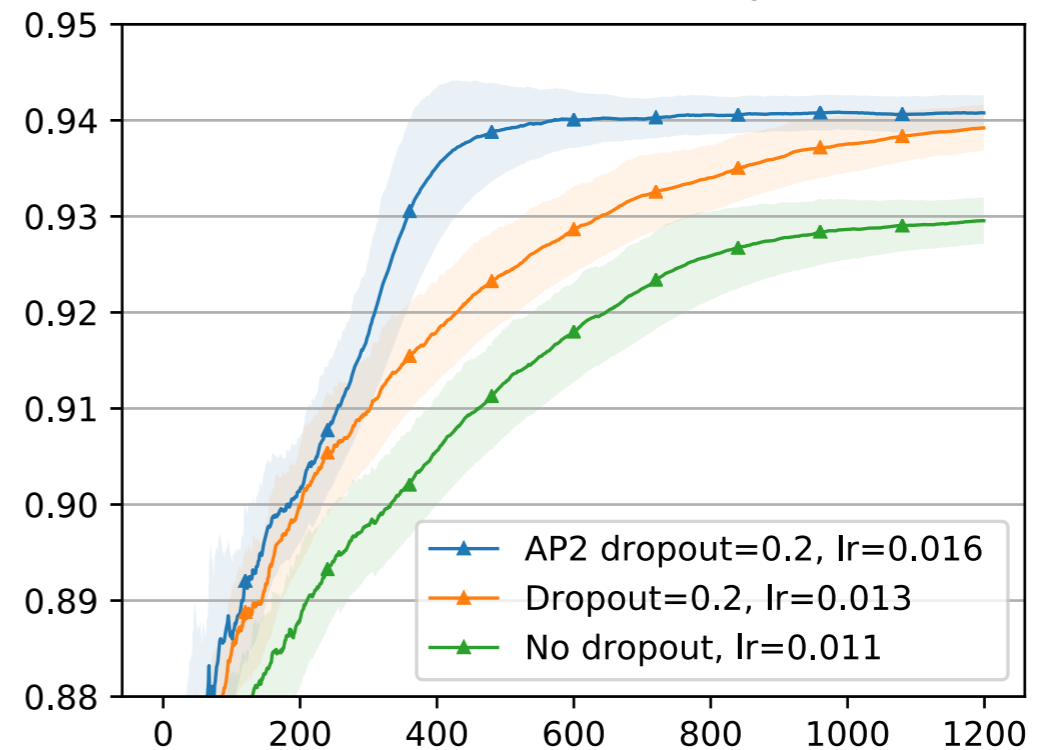
Training Loss



Validation Loss



Validation Accuracy

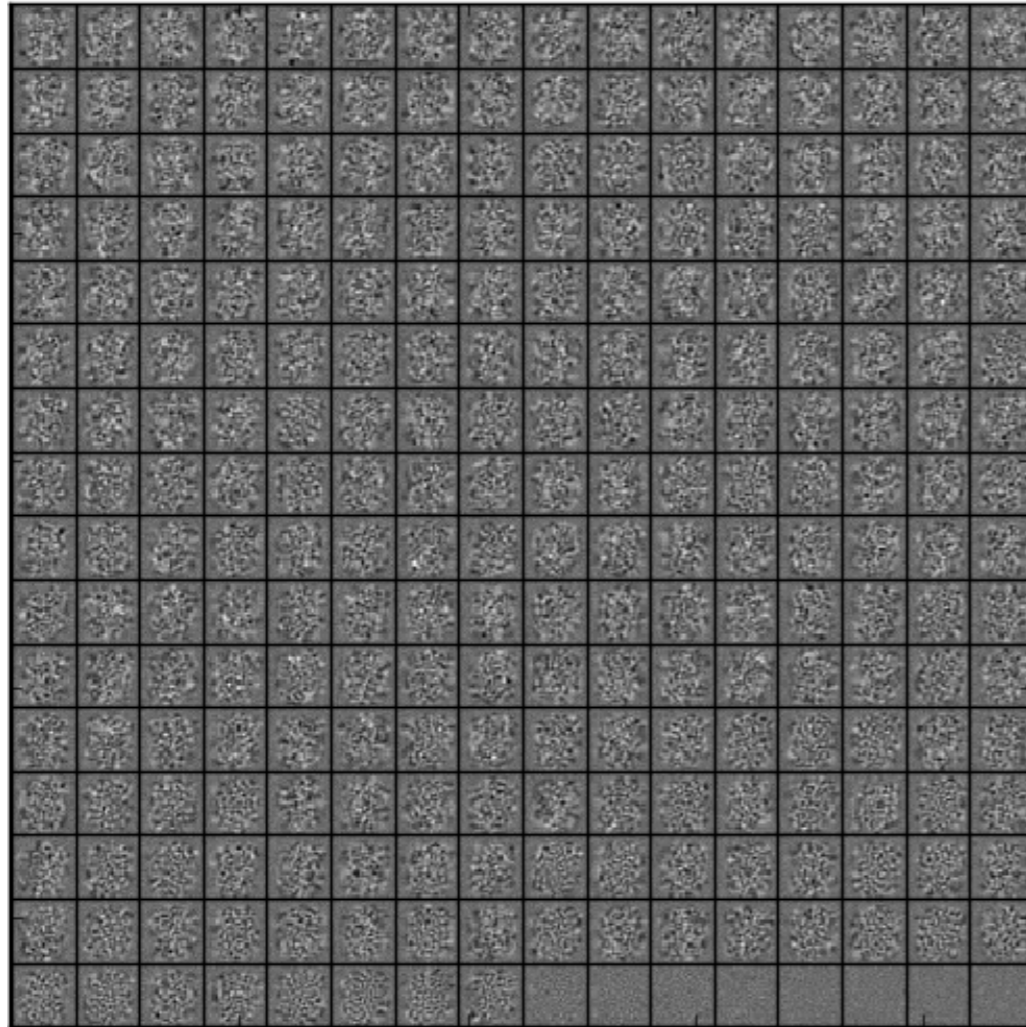


- ◆ Typically need to train longer due to higher gradient variance
- There are techniques to approximate the effect analytically:
Fast Dropout, Analytic Dropout (AP2)

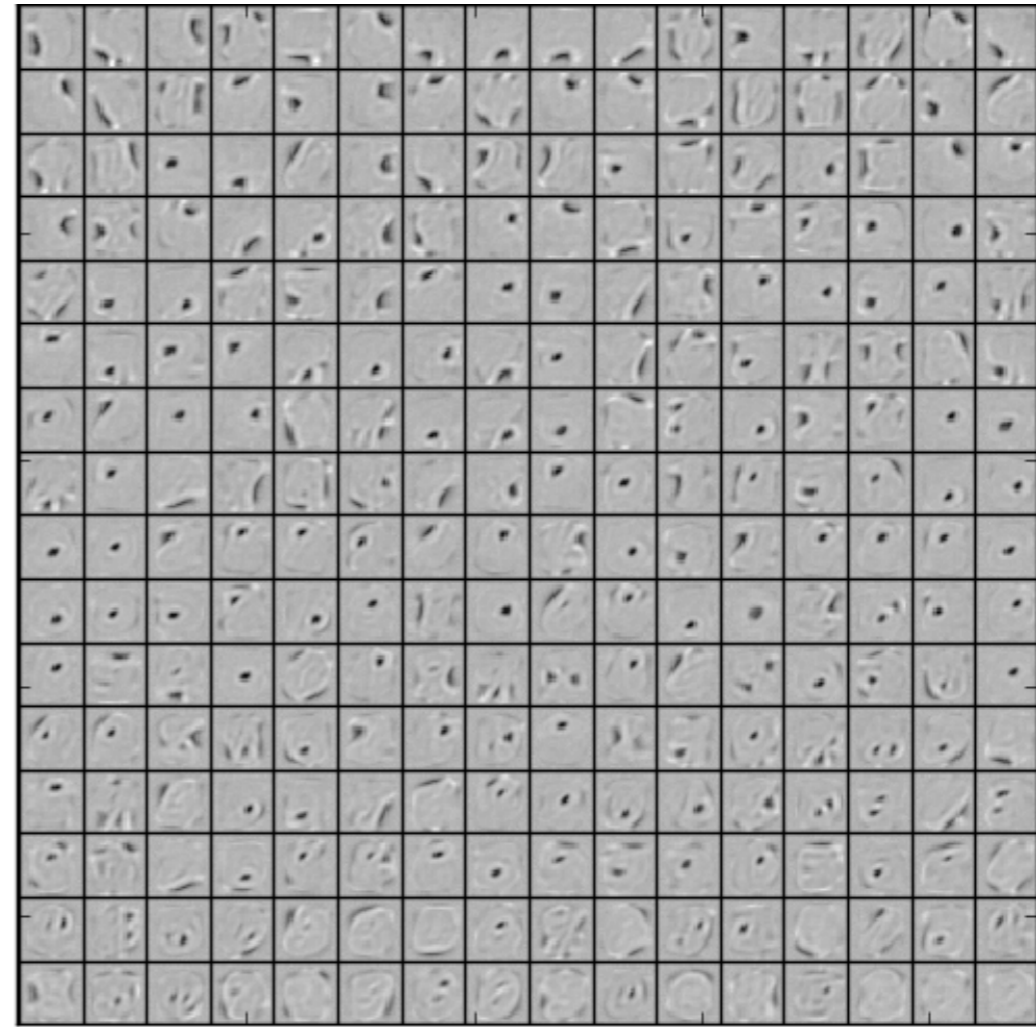
Effect on Features

✦ 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)]

- ✦ Hypothesis: dropout prevents co-adaptation (learns simpler and more robust features)
- ✦ Further interesting studies in the paper: effect on activation sparsity, connection to ridge regression, etc.