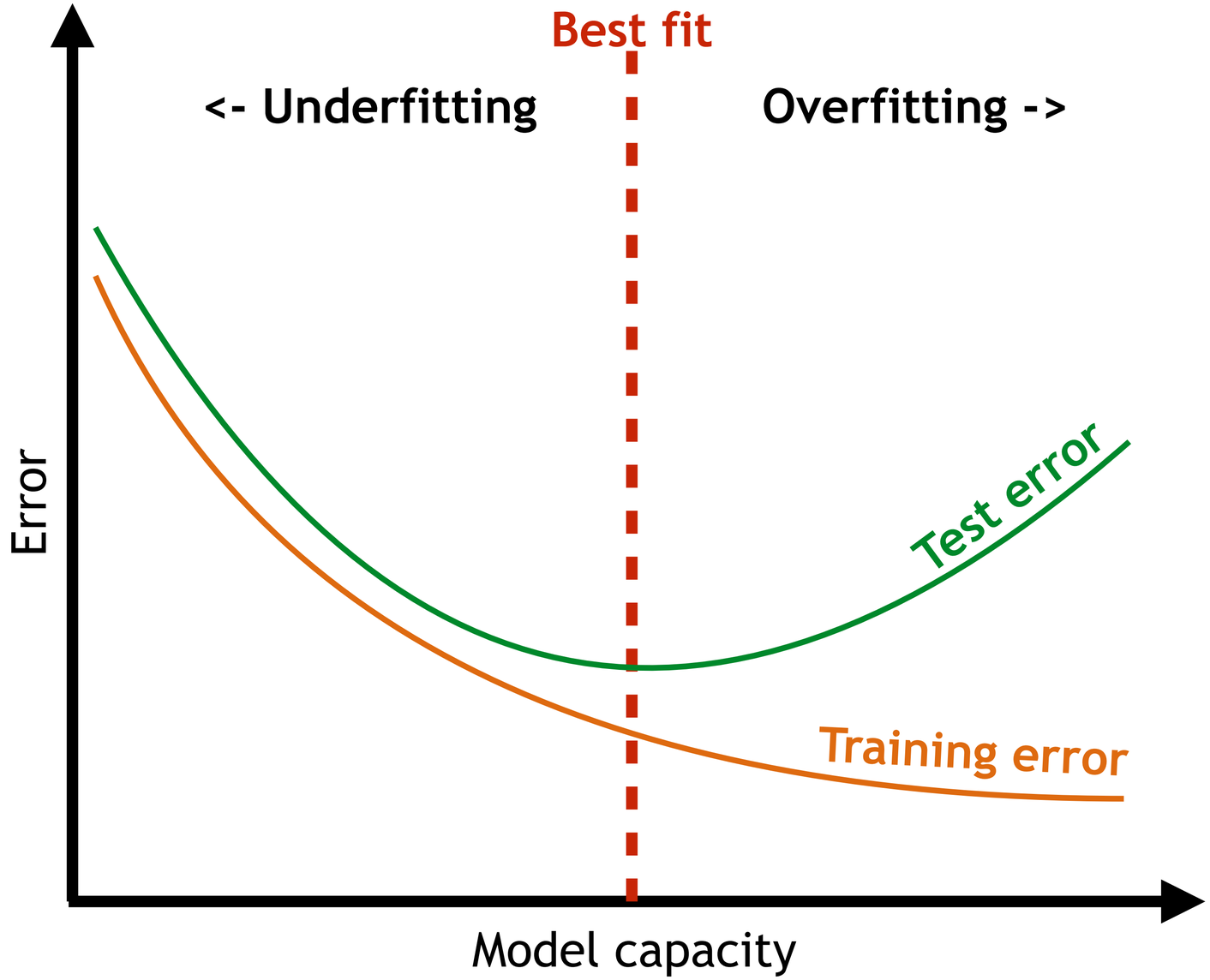
EMBEDDED MACHINE LEARNING
LECTURE 05 - REGULARIZATION

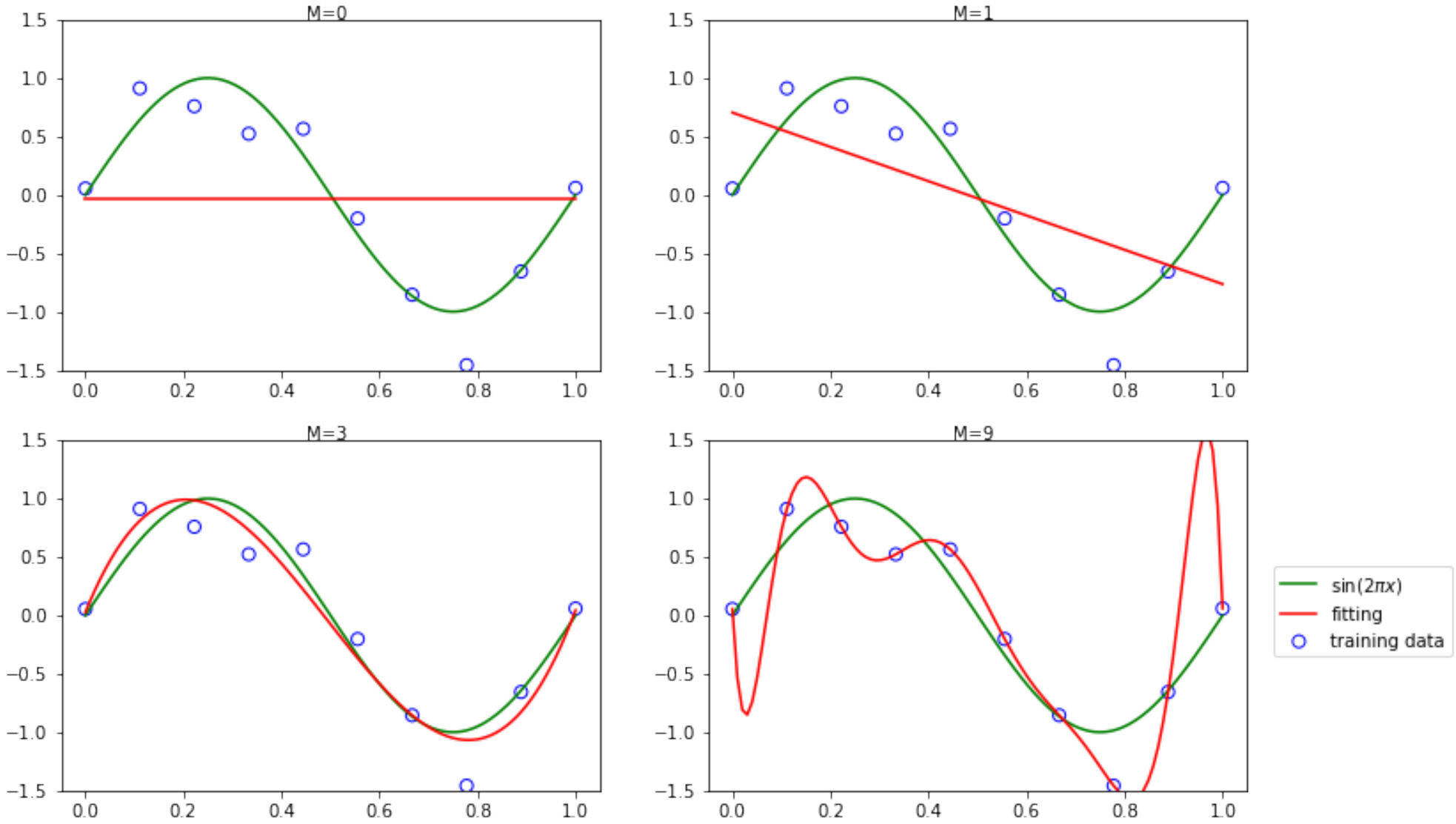
Holger Fröning
holger.froening@ziti.uni-heidelberg.de
HAWAII Group (formerly Computing Systems), Institute of Computer Engineering
Heidelberg University

With material from Roger Grosse (U. Toronto, CSC421/2516), Goodfellow et al. (Deep Learning)

WHY DO WE NEED REGULARIZATION?



RECAP: MODEL SELECTION



RECAP: REGULARIZATION

Regularization can control overfitting by adding a penalty term to the error function

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(h(x_n, \mathbf{w}) - t_n\right)^2 + \frac{\lambda}{2}\|\mathbf{w}\|$$

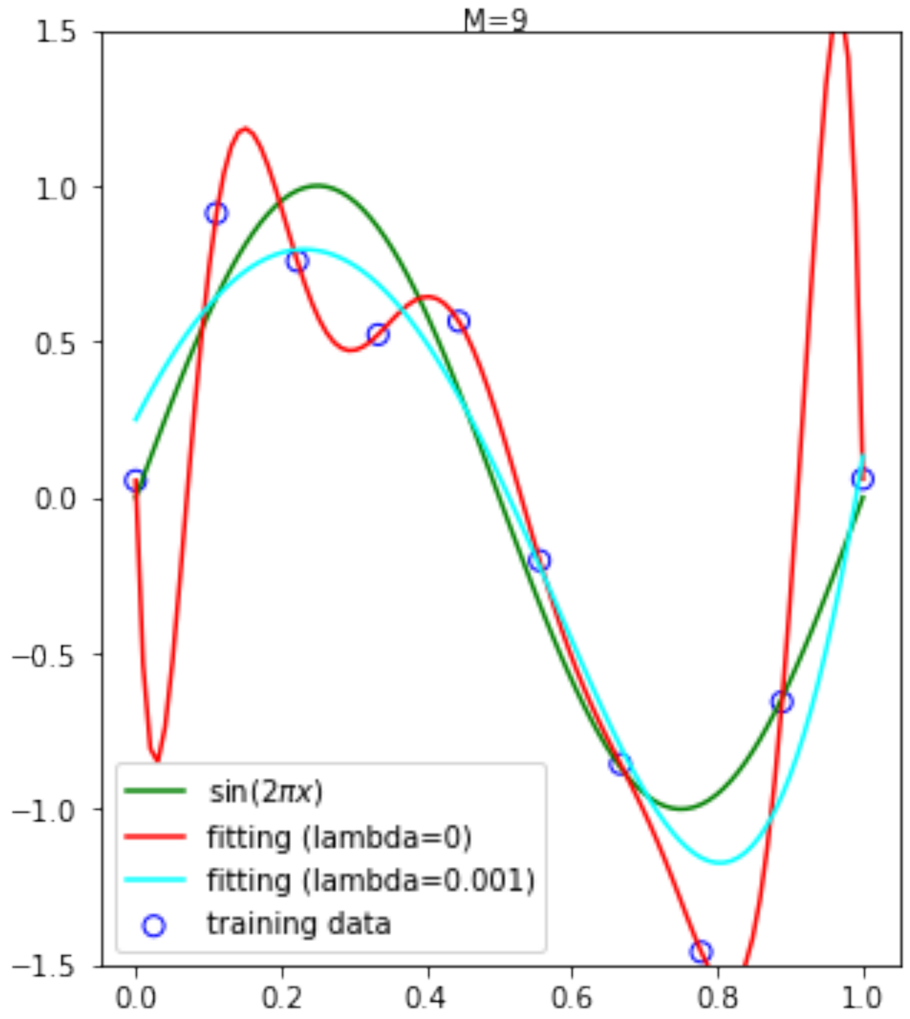
where $\|\mathbf{w}\| = \mathbf{w}^T\mathbf{w}$

λ governs the relative importance of the regularization term

Such shrinkage methods reduce the value of the coefficients

Quadratic regularizer: *ridge regression* or *weight decay* or *L2 regularization*

Validation set to optimize either M or λ



OPTIMALITY IS DIFFICULT IN ML

"No free lunch" theorem: there is no single best ML method

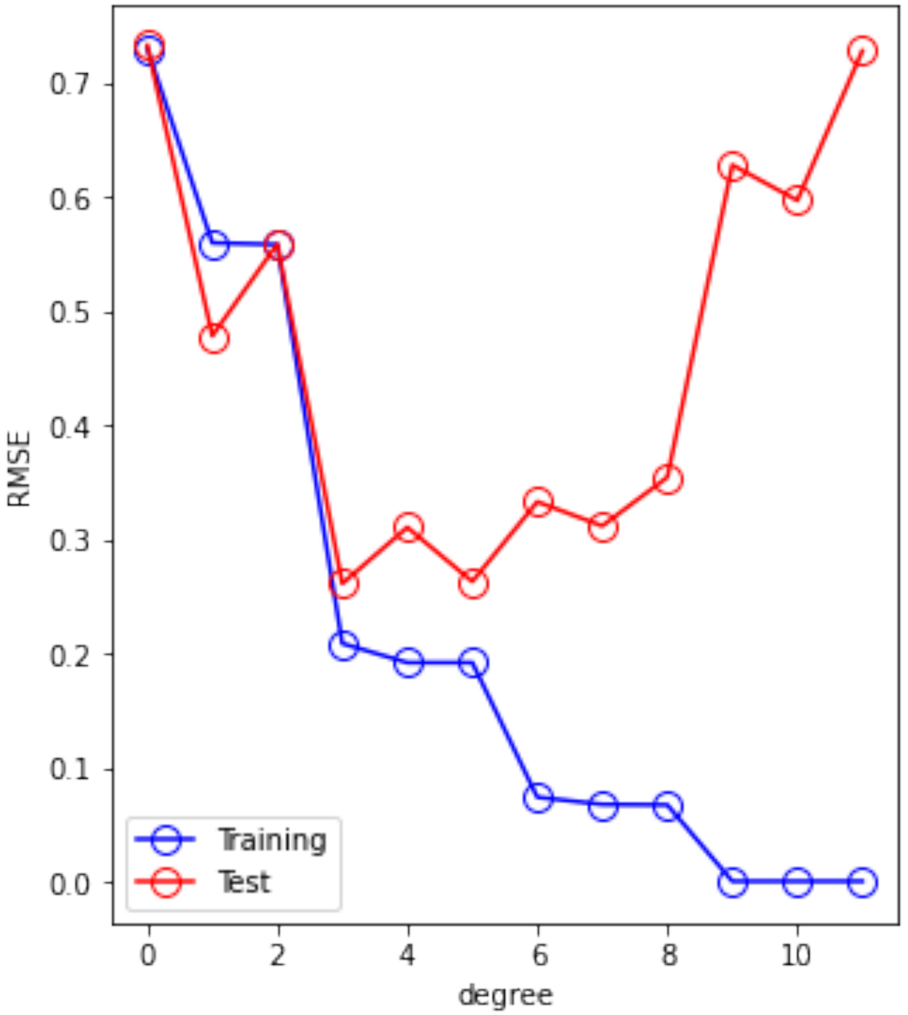
- Instead, ML methods are to be chosen to be most suited for a given task at hand
- Same applies for regularization

"Occam's Razor": always choose the simplest of multiple competing hypotheses, given they perform equally well

- Sufficiently complex as generalization error is a u-shaped curve
- See statistical learning theory for various means to quantify model complexity

Regularization is any modification of a learning method that reduces its generalization error but not the training error

- Machine learning is probabilistic, for exact reasoning see for instance logical reasoning



RANDOM VARIABLES

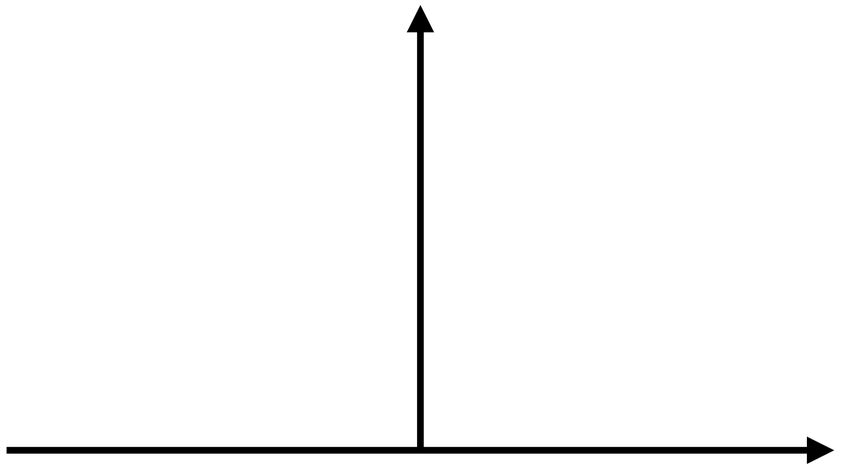
EXPECTATION AND VARIANCE

Given a random variable X, we often compute the expectation and variance

The expectation $\mathbb{E}$ describes the average value

For discrete random variables: $\mathbb{E}(X) = \sum_x x p_x(x)$

For continuous random variables: $\mathbb{E}(X) = \int_{-\infty}^{\infty} x f_x(x)$



Given the sum or integral exists

The variance Var describes the spread (amount of variability) around the expectation

$\text{Var}(X) = \mathbb{E}((X - \mathbb{E}(X))^2) = \mathbb{E}(X^2) - \mathbb{E}(X)^2$

The standard deviation of a random variable is then: $\text{std}(X) = \sqrt{\text{Var}(X)}$

If a random variable is deterministic: $\mathbb{E}(X) = x$, and $\text{Var}(X) = 0$

BIAS AND VARIANCE

The bias of an estimator is defined as: $\text{Bias}(\hat{y}_m) = \mathbb{E}(\hat{y}_m) - y$

The variance of an estimator is its variance $\text{Var}(\hat{y}_m)$

The mean squared error (MSE) of an estimate is defined as

$$\text{MSE} = \mathbb{E}\left[(\hat{y}_m - y)^2\right] = \text{Bias}(\hat{y}_m)^2 + \text{Var}(\hat{y}_m) + \text{Bayes Error}$$

Thus, we not only want small MSE but also a good trade among bias and variance

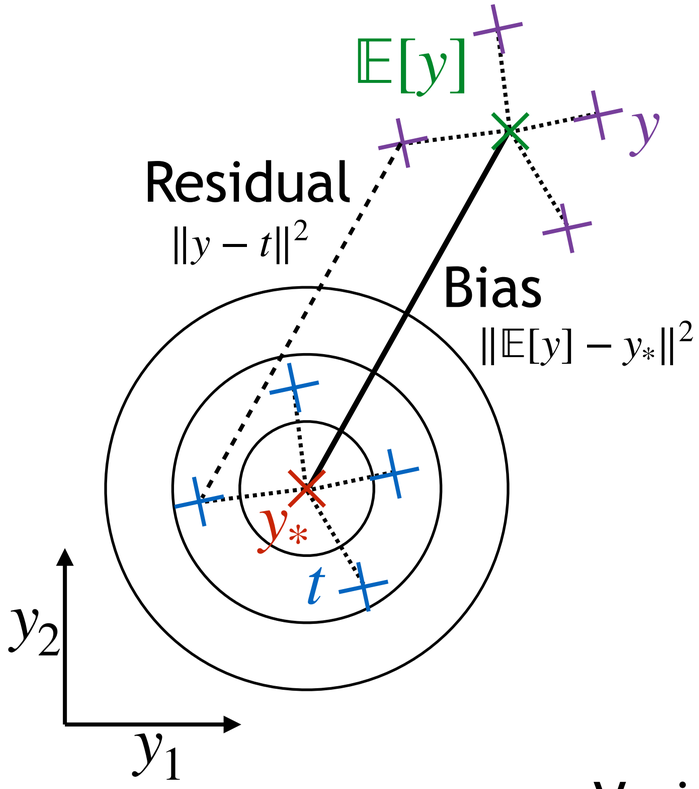
Most common solution is cross-validation (shuffling training and test set)

Still, fundamentally bias and variance will always be a trade-off

BIAS AND VARIANCE

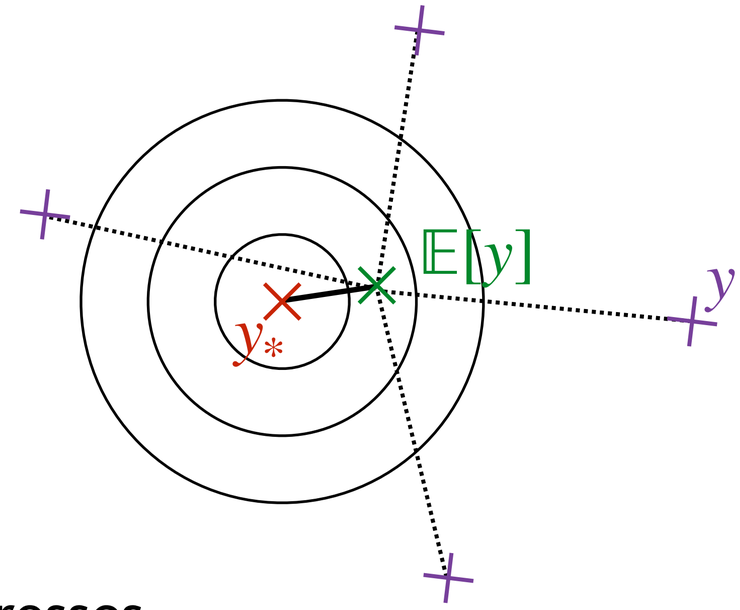
Underfitting

low variance, high bias



Overfitting

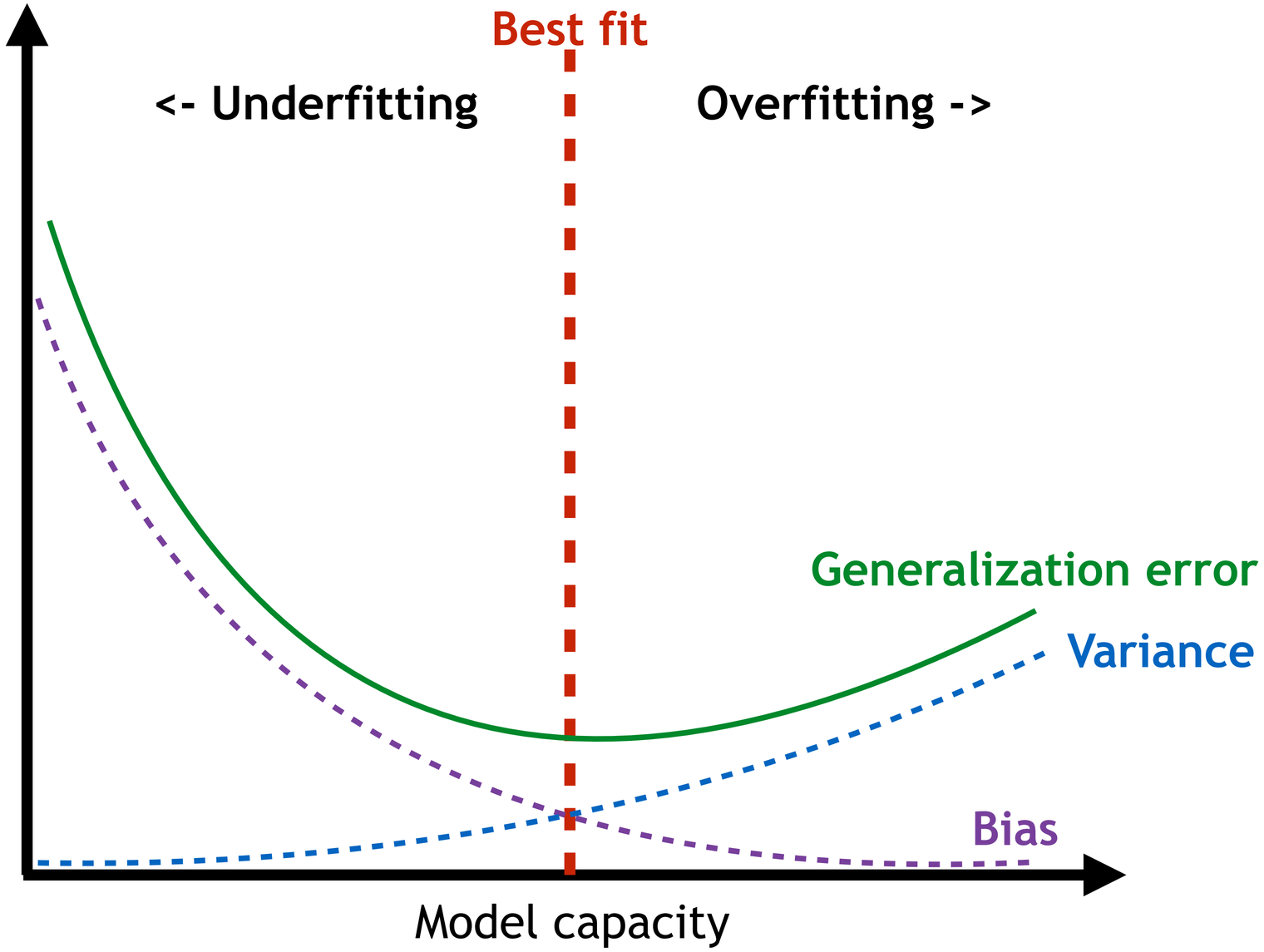
High variance, low bias



Contour of expected loss

Variance is the spread in the purple crosses

Bayes error is the spread in the blue crosses


Best fit
<- Underfitting
Overfitting ->
Generalization error
Variance
Bias
Model capacity

REGULARIZATION

Regularization refers to a set of different methods that lower the complexity of a neural network model during training to prevent overfitting

Many regularization approaches are based on limiting the capacity of models

- Neural networks, linear regression, polynomial regression, etc.
- A form of regression that shrinks (constrains, regularizes) the coefficient estimates (weights, not biases) towards zero
- Prevents the learning of complex models to avoid the risk of overfitting
- Penalize the flexibility of a model

Trading increased bias for reduced variance

- Profitable trade: reducing variance significantly while not overly increasing the bias

Regularizer examples: shrinkage methods (capacity reduction), early stopping, dropout, weight initialization techniques, and batch normalization

REGULARIZATION

EARLY STOPPING

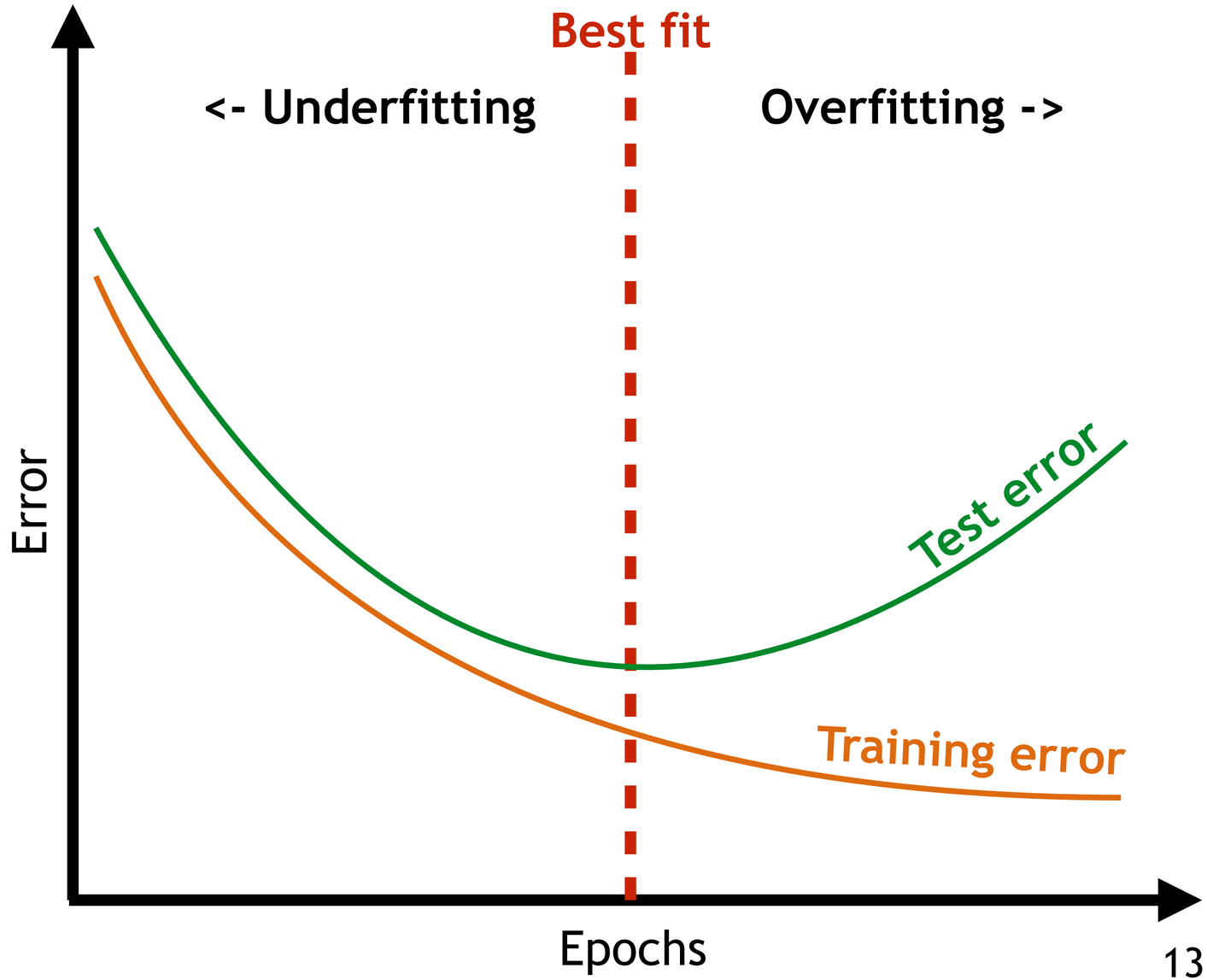
Monitoring performance on validation set (often: test set), stopping training when error starts increasing

Often difficult in practice due to error fluctuations

- Stochasticity of SGD

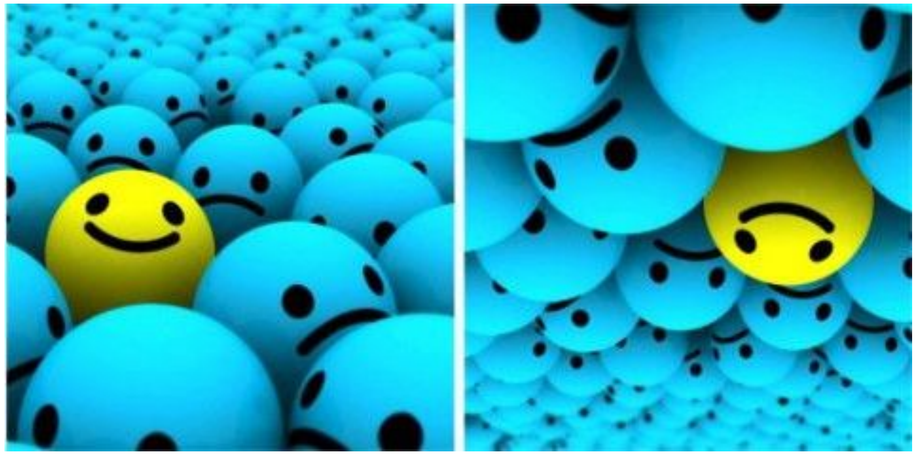
Intuition: weights start small, growing with number of epochs

- Stopping early has an effect similar to weight decay

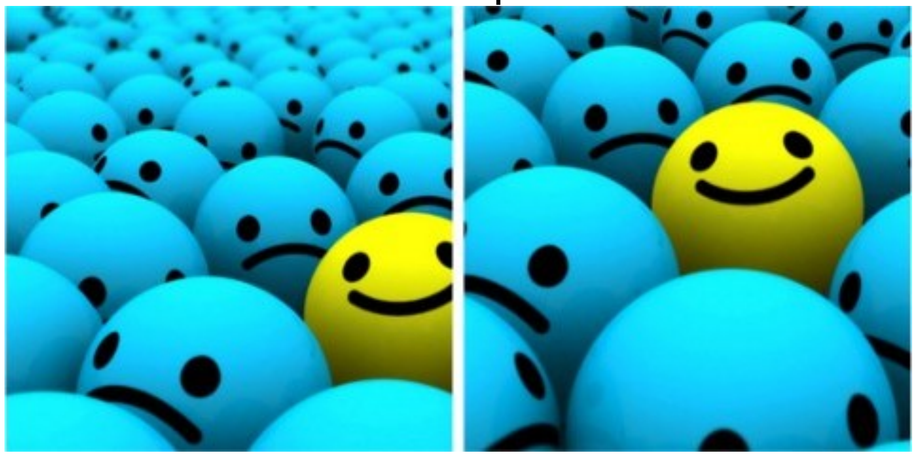


DATA AUGMENTATION

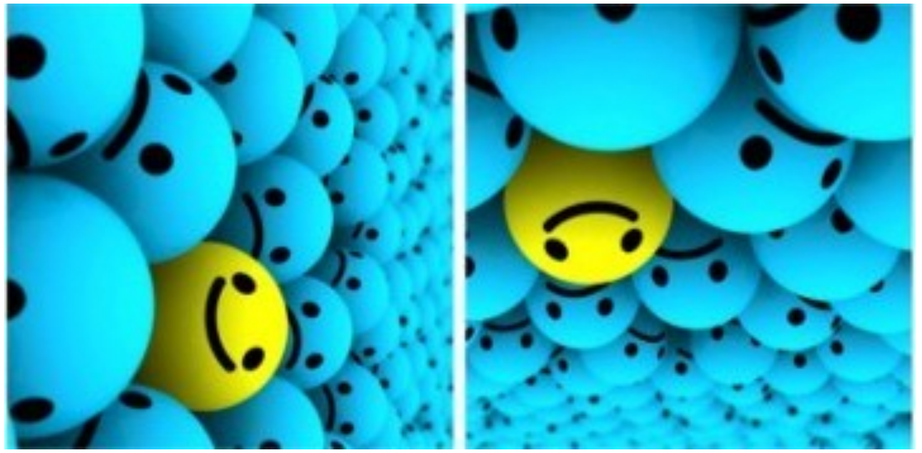
Flip



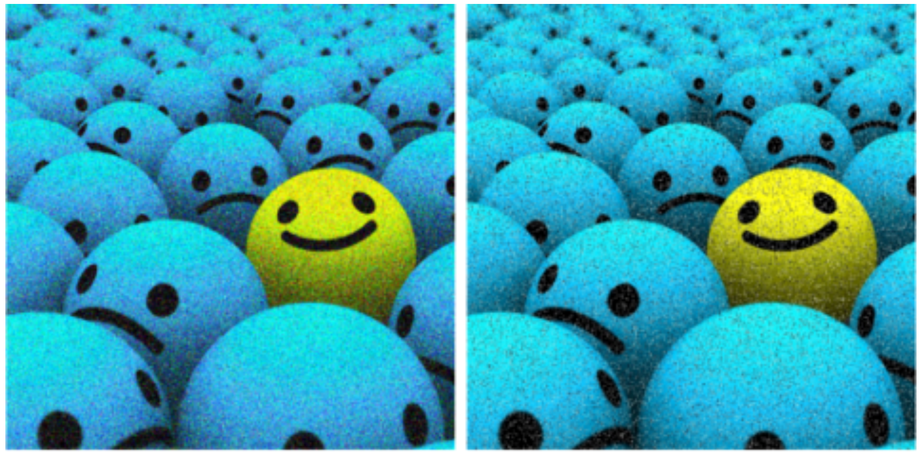
Crop



Rotate



Noise



DATA AUGMENTATION

Mixup

Cutout

CutMix

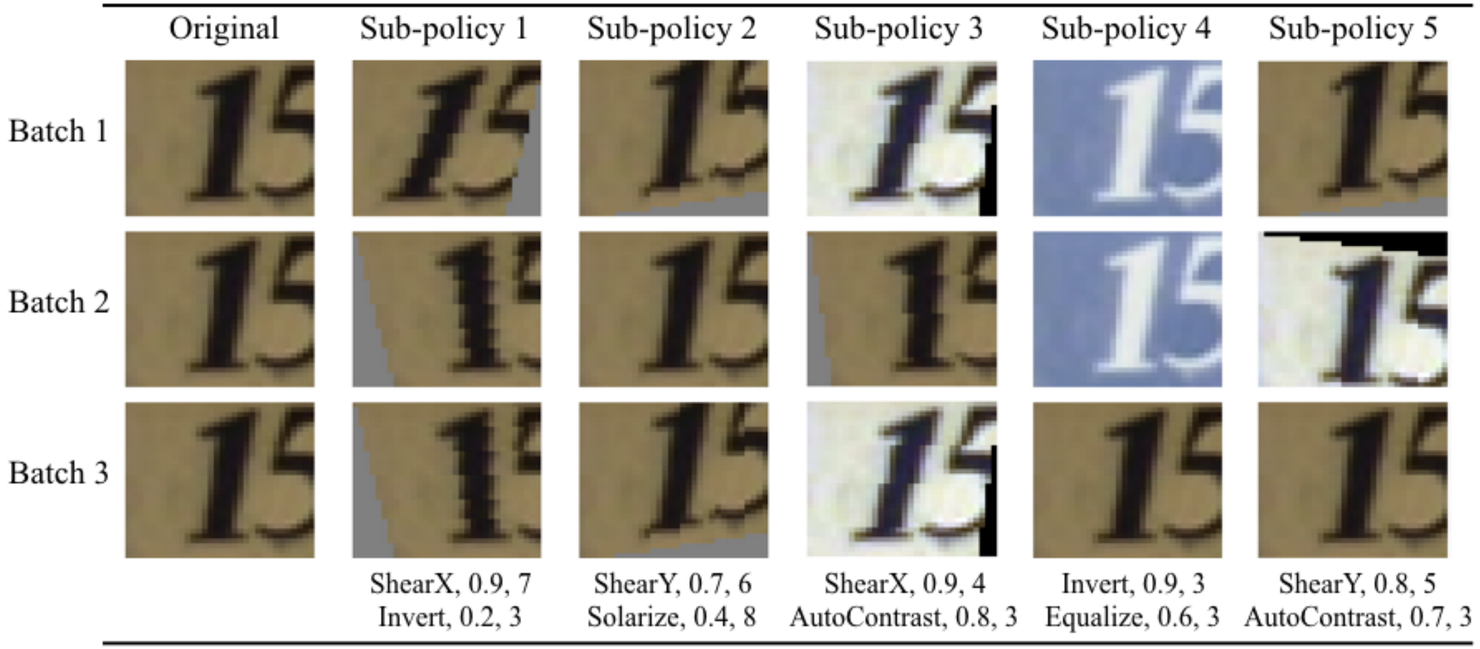
Ground truth labels are mixed proportionally to the area of the patches

DeVries et al., "Improved Regularization of CNNs with Cutout", Arxiv 2017

Zhang et al., "Mixup: Beyond Empirical Risk Minimization", ICLR 2018

Yun et al., "CutMix: Regularization Strategy to Train Strong Classifiers", ICCV 2019

DATA AUGMENTATION



~1000 GPU hours to train for SVHN (instead of a couple of hours)

	Operation 1	Operation 2
Sub-policy 0	(ShearX,0.9,4)	(Invert,0.2,3)
Sub-policy 1	(ShearY,0.9,8)	(Invert,0.7,5)
Sub-policy 2	(Equalize,0.6,5)	(Solarize,0.6,6)
Sub-policy 3	(Invert,0.9,3)	(Equalize,0.6,3)
Sub-policy 4	(Equalize,0.6,1)	(Rotate,0.9,3)
Sub-policy 5	(ShearX,0.9,4)	(AutoContrast,0.8,3)
Sub-policy 6	(ShearY,0.9,8)	(Invert,0.4,5)
Sub-policy 7	(ShearY,0.9,5)	(Solarize,0.2,6)
Sub-policy 8	(Invert,0.9,6)	(AutoContrast,0.8,1)
Sub-policy 9	(Equalize,0.6,3)	(Rotate,0.9,3)
Sub-policy 10	(ShearX,0.9,4)	(Solarize,0.3,3)
Sub-policy 11	(ShearY,0.8,8)	(Invert,0.7,4)
Sub-policy 12	(Equalize,0.9,5)	(TranslateY,0.6,6)
Sub-policy 13	(Invert,0.9,4)	(Equalize,0.6,7)
Sub-policy 14	(Contrast,0.3,3)	(Rotate,0.8,4)
Sub-policy 15	(Invert,0.8,5)	(TranslateY,0.0,2)
Sub-policy 16	(ShearY,0.7,6)	(Solarize,0.4,8)
Sub-policy 17	(Invert,0.6,4)	(Rotate,0.8,4)
Sub-policy 18	(ShearY,0.3,7)	(TranslateX,0.9,3)
Sub-policy 19	(ShearX,0.1,6)	(Invert,0.6,5)
Sub-policy 20	(Solarize,0.7,2)	(TranslateY,0.6,7)
Sub-policy 21	(ShearY,0.8,4)	(Invert,0.8,8)
Sub-policy 22	(ShearX,0.7,9)	(TranslateY,0.8,3)
Sub-policy 23	(ShearY,0.8,5)	(AutoContrast,0.7,3)
Sub-policy 24	(ShearX,0.7,2)	(Invert,0.1,5)

Cubuk et al., "AutoAugment: Learning Augmentation Strategies from Data", CVPR 2019

DATA AUGMENTATION IN PYTORCH

```
transforms_train = transforms.Compose([
      transforms.RandomCrop(32, padding=4),
      transforms.RandomRotation(3.8),
      transforms.RandomVerticalFlip(p=0.4),
      transforms.RandomHorizontalFlip(p=0.3),
      transforms.RandomAffine(translate=(0.8,0.9), shear=[0.2,0.4,0.7]),
      transform.ToTensor(),
      ...])

transforms_test = tranforms.Compose([
      transforms.ToTensor(),
      ...])
```

The choice of transformations highly depends on the task

E.g. horizontal flip for object recognition, but not for MNIST

Only augment the training set, not the test set

REDUCING THE NUMBER OF PARAMETERS

Number of parameters (respectively their precision) directly correlates with model capacity

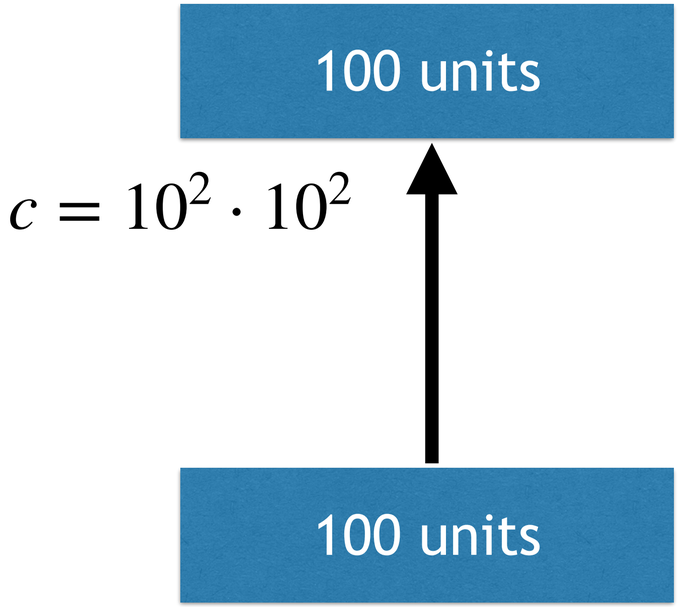
=> Linear bottleneck layers

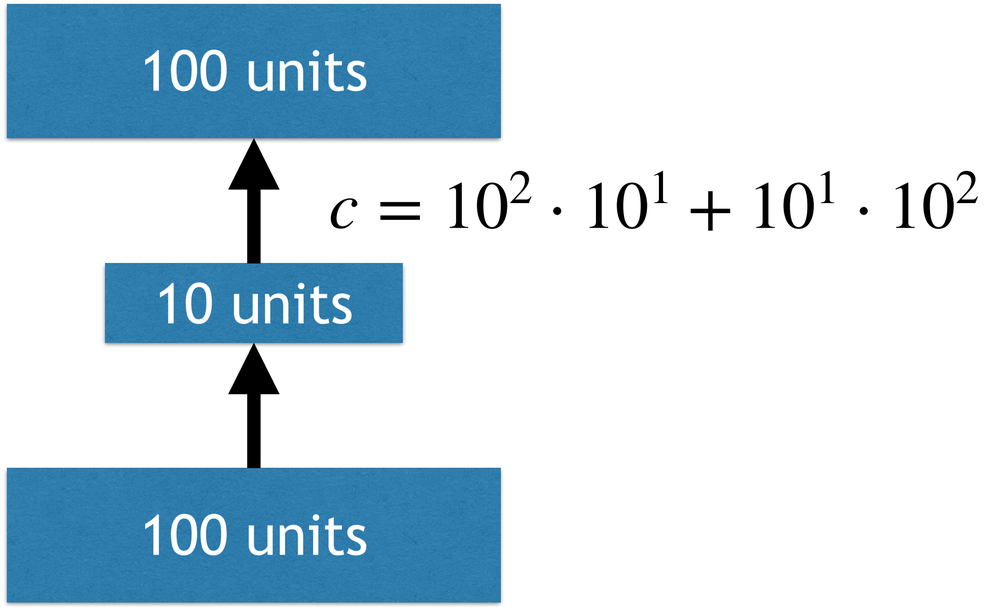
Remember that additional linear layers do not improve expressivity, but they might still improve generalization

Why is the left architecture more expressive than the right one?

Expressiveness == number of connections c

Note that such a static reduction can make the model architecture too simple to learn complex patterns => dynamic approaches more desirable





L2 NORM (AKA WEIGHT DECAY)

Regularization can control overfitting by adding a penalty term to the cost function

$$\mathcal{J}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}(y(\mathbf{w}, \mathbf{x}_n), t_n) + \lambda \mathcal{R}(\mathbf{w})$$

where for an L2 norm $\mathcal{R}(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2 = \frac{1}{2}\mathbf{w}^T\mathbf{w} = \frac{1}{2}\sum_j w_j^2$

and λ governs the relative importance of the regularization term (weight cost)

Update rule for SGD

$$w_j := w_j - \eta\frac{\partial \mathcal{J}}{\partial w_j} = w_j - \eta\left(\frac{\partial \mathcal{L}}{\partial w_j} + \lambda\frac{\partial \mathcal{R}}{\partial w_j}\right) = w_j - \eta\left(\frac{\partial \mathcal{L}}{\partial w_j} + \lambda w_j\right) = (1 - \eta\lambda)w_j - \eta\frac{\partial \mathcal{L}}{\partial w_j}$$

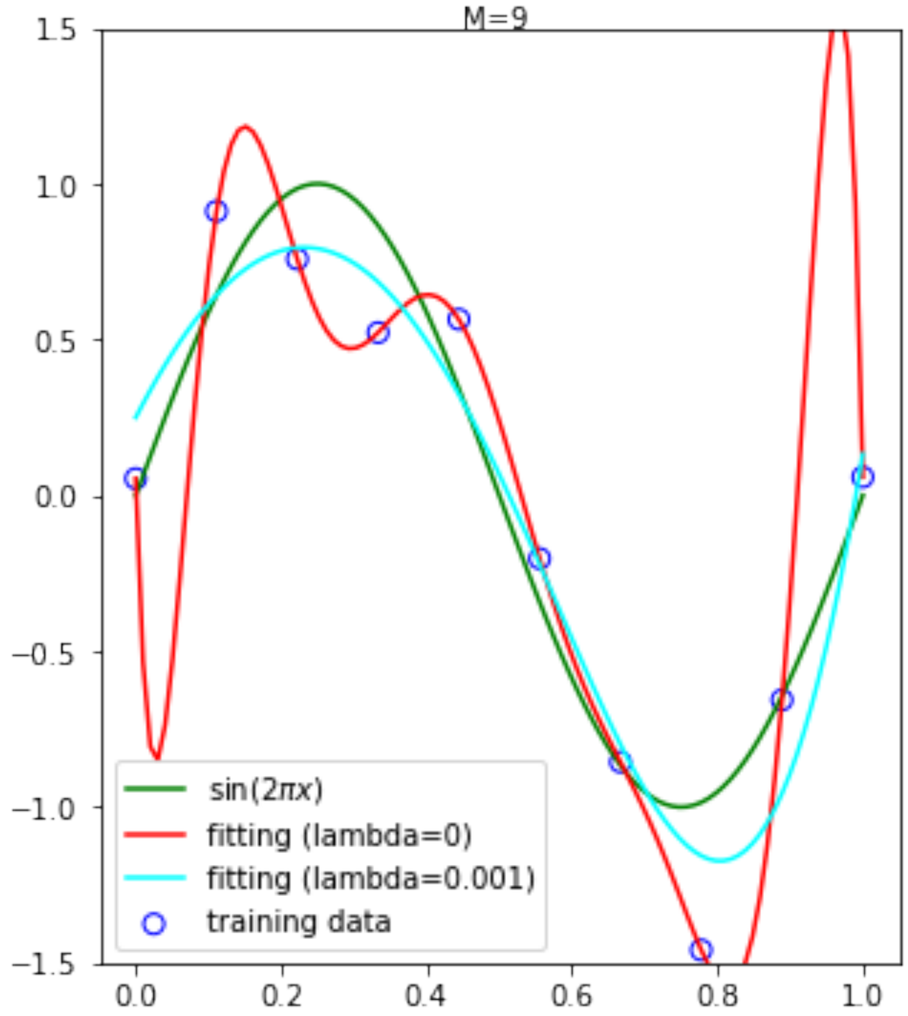
Each SGD step, we shrink weights by a factor of $(1 - \eta\lambda)$ => “weight decay”

Krogh et al., “A Simple Weight Decay Can Improve Generalization”, NIPS 1991

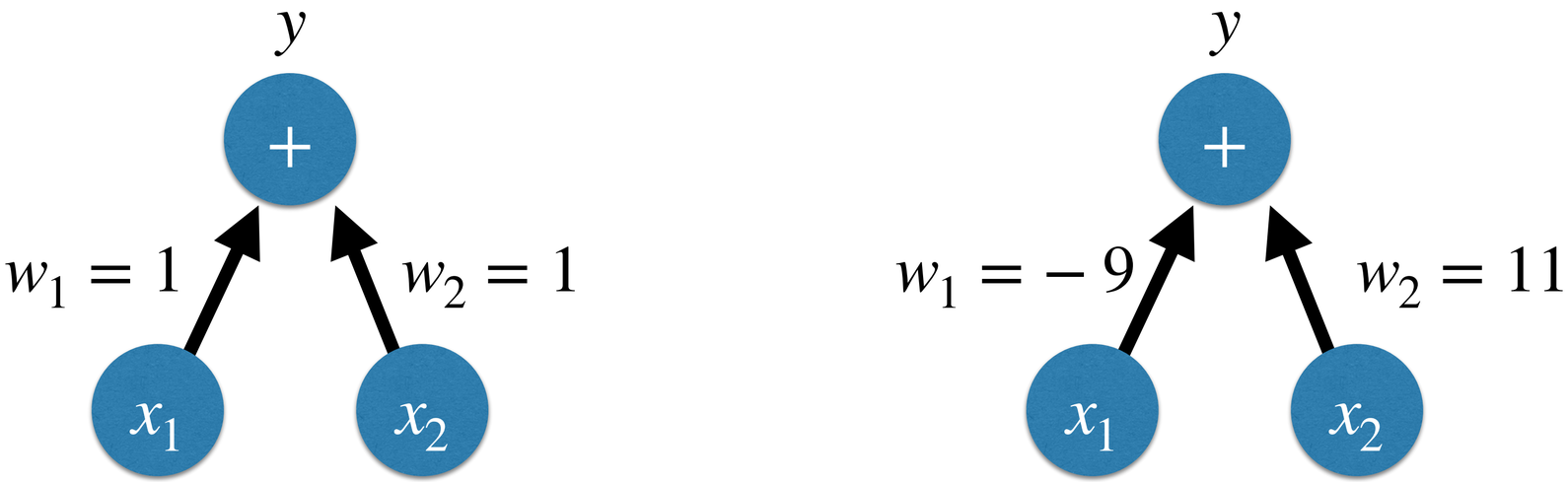
MODEL PARAMETER ANALYSIS

M	1	2	4	9	9
Reg.	no	no	no	no	L2
Weights	-3.6E-02	7.8E-01	1.1E-02	-3.2E+01	1.8E-01
		-1.6E+00	9.3E+00	5.5E+02	5.3E+00
			-2.7E+01	-2.7E+03	-1.0E+01
			1.7E+01	4.8E+03	-4.3E+00
				2.0E+03	1.8E+00
				-1.9E+04	4.5E+00
				2.8E+04	4.4E+00
				-1.8E+04	2.4E+00
				4.2E+03	-6.1E-01
					-4.2E+00

Overfit (often?) correlates with large weights



AN INTUITION

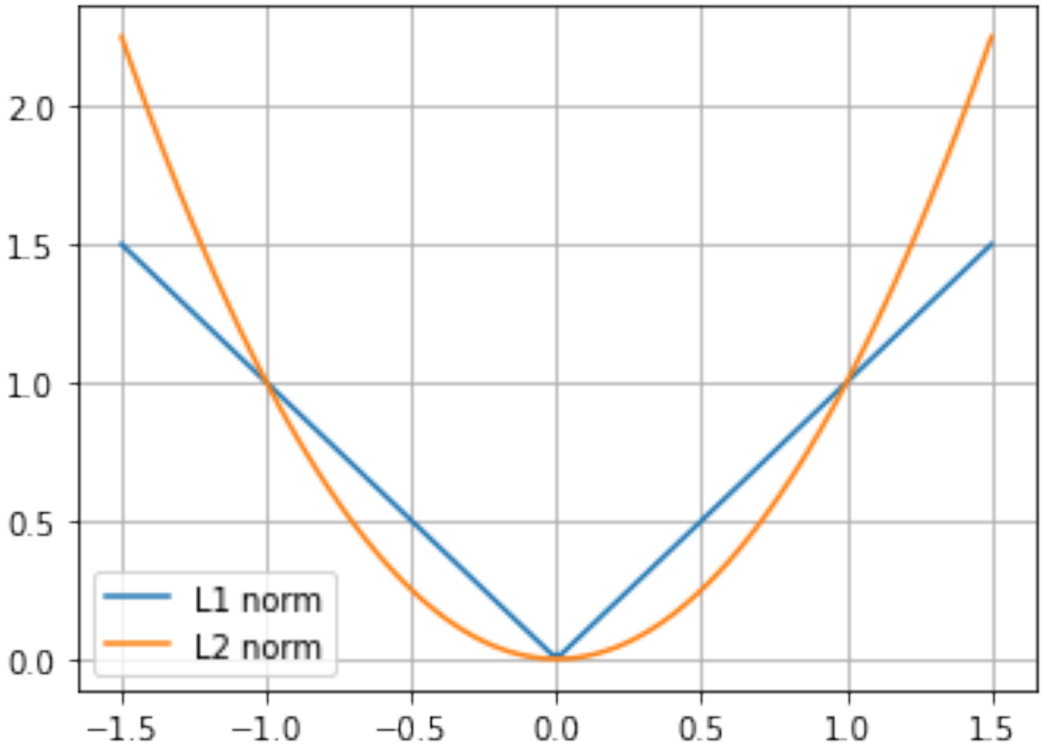


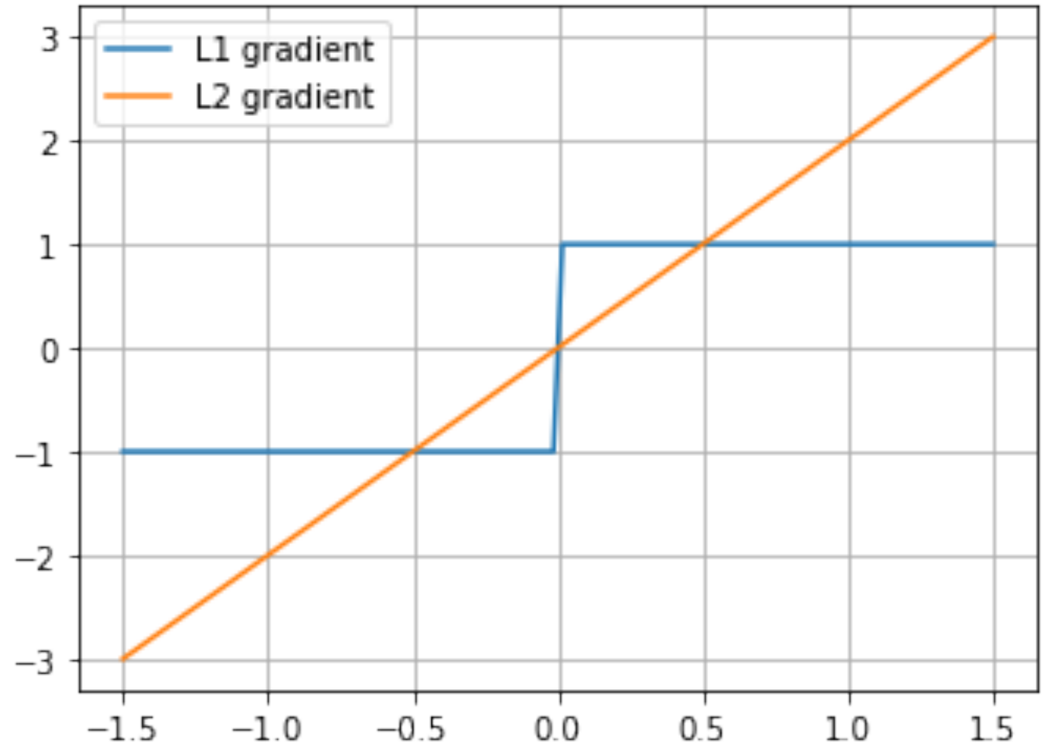
$$\frac{\partial y}{\partial x_1} = w_1, \frac{\partial y}{\partial x_2} = w_2$$

Assume x_1 and x_2 are equal

Assume either one slightly changes

L1 NORM (AKA LASSO)





$$\mathscr{R}_{L1}(\mathbf{w}) = \|\mathbf{w}\|_1 = \sum_j |w_j|$$

$$\mathscr{R}_{L2}(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2 = \frac{1}{2}\sum_j w_j^2$$

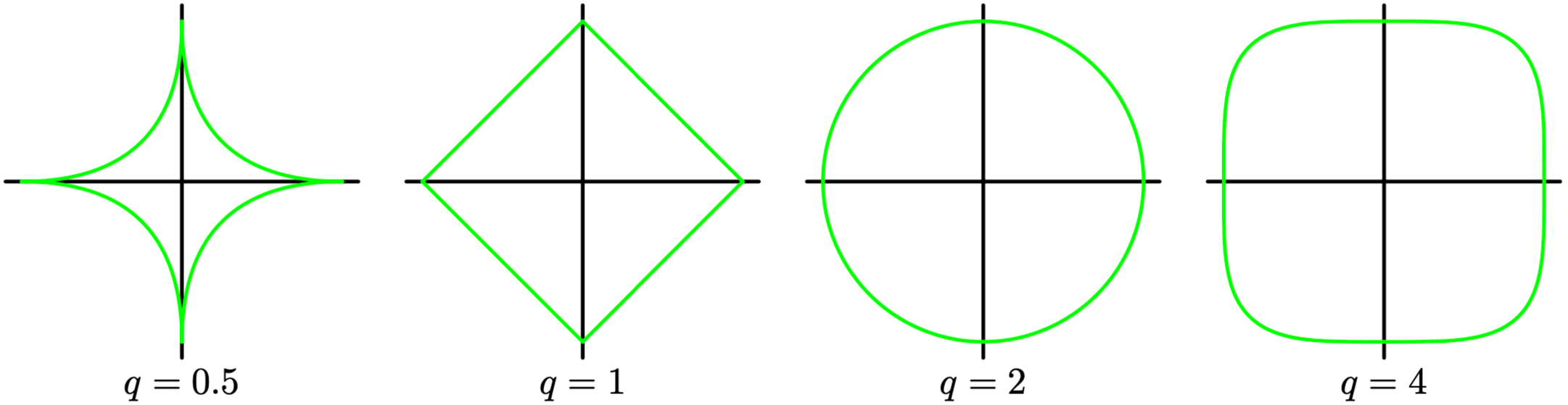
GEOMETRIC INTERPRETATION

Think of the norms as of solving an equation, where the sum of squares resp. sum of absolute values equals a constant s

E.g., for two weights: L2: $w_1^2 + w_2^2 = s$ and L1: $|w_1| + |w_2| = s$

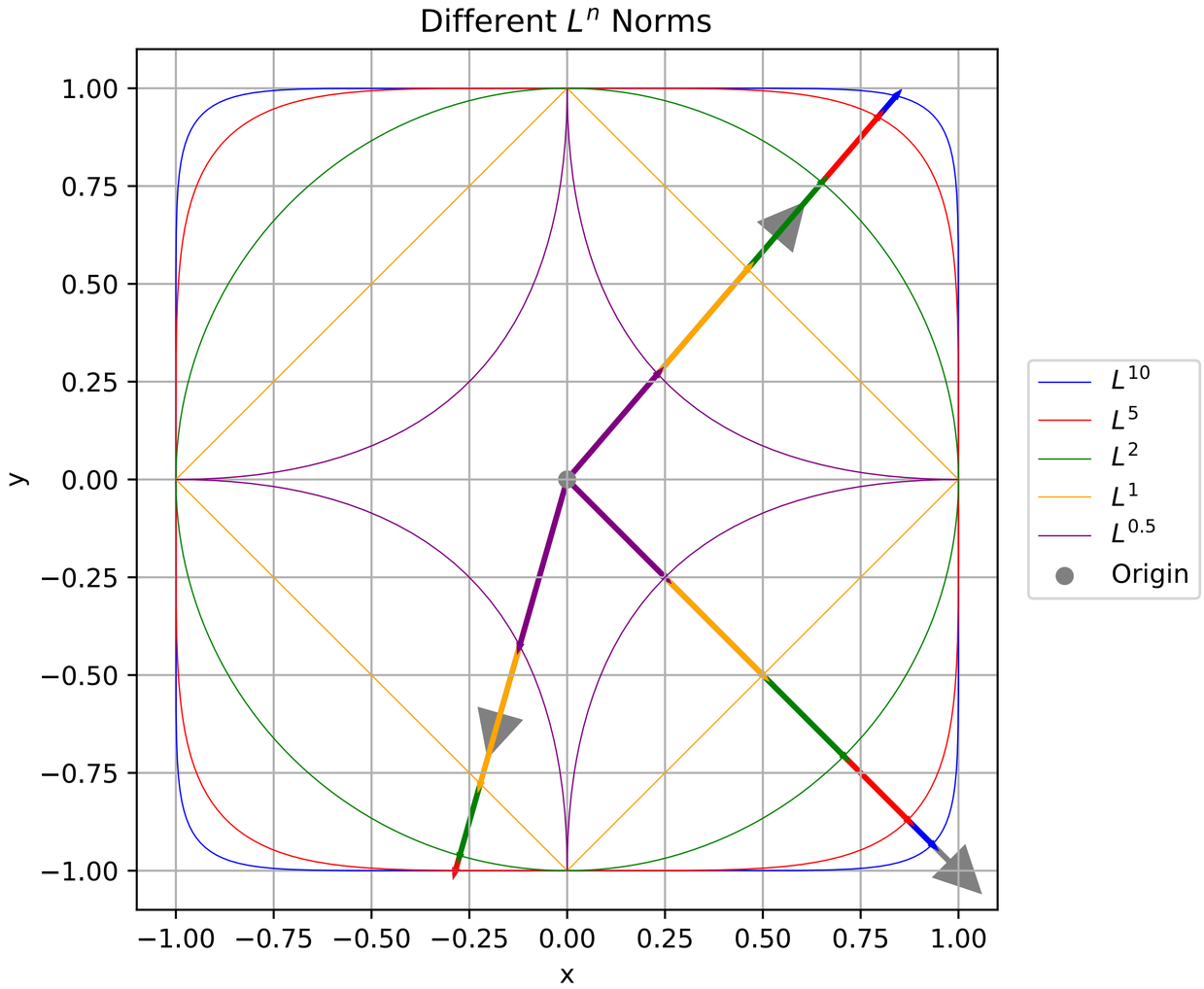
What geometric form would be described by these equations?

$$\frac{1}{q}\|\mathbf{w}\|_q = \frac{1}{q}\sum_j |w_j|^q$$



Bishop - Pattern Recognition And Machine Learning, Springer, 2006

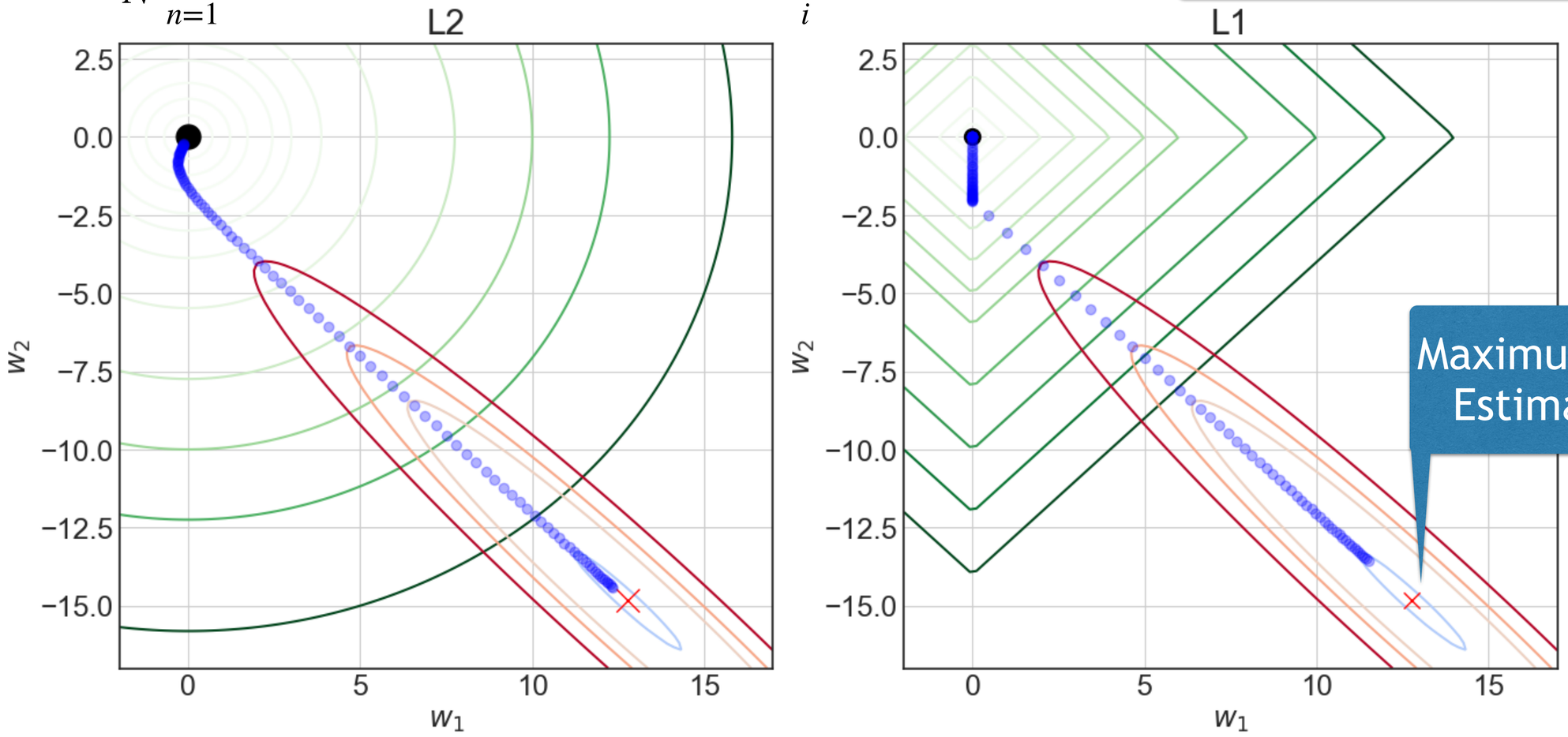
BEHAVIOR OF DIFFERENT L-NORMS ON EXAMPLE VECTORS



RIDGE AND LASSO: VISUALIZING THE OPTIMAL SOLUTIONS

$$\mathcal{J}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(y(\mathbf{w}, \mathbf{x}_n), t_n) + \lambda\mathcal{R}(\mathbf{w}); \mathcal{L} = \sum_{i}(y_i - t_i)^2$$

OLS == an ellipse contour plot centered around MLE



Maximum Likelihood Estimation (MLE)

https://xavierbourretsicotte.github.io/ridge_lasso_visual.html

ENSEMBLES

Ensemble: average the predictions of multiple networks trained independently on separate training sets

- Reduce the variance of the predictions => lower loss

Or simulate the effects of independent training sets by other means (training procedure)

- Bagging: train on random subsets of the full training data
- Different variants of a model architecture
- Entirely different models or learning algorithms

Better prediction performance can be proven for convex loss functions

- Lots of loss functions are convex wrt y, such as squared error or cross entropy
- Doesn't hold for non-convex functions, but still works out surprisingly well in practice

STOCHASTICITY

Use stochasticity to improve generalization

- Up to now, all model architectures have been deterministic
- Introducing (some) stochasticity can improve overfitting

Dropout: drop out each unit by setting its output activation to zero with some probability ρ

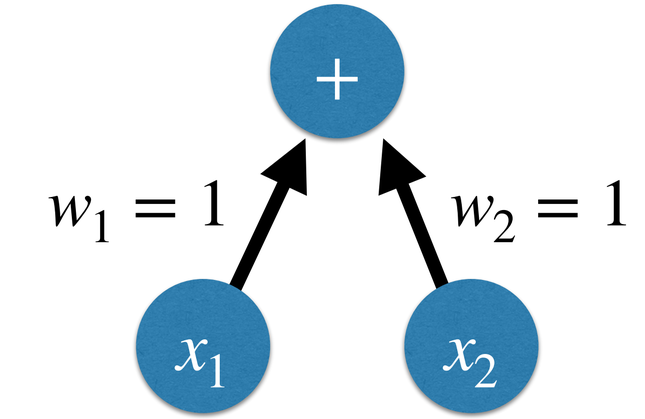
- Represent this by a mask m_i that randomly is set to $\{0,1\}$

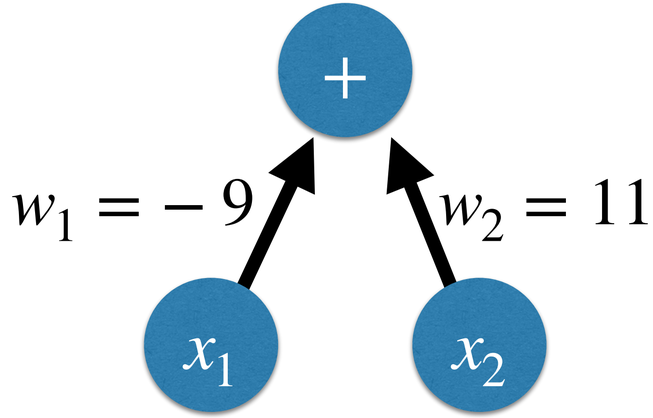
$$h_i = m_i \cdot \sigma(z^{(i)}), \text{ with } \overline{z^{(i)}} = \overline{h_i}\frac{dh_i}{dz^{(i)}} = \overline{h_i} \cdot m_i \cdot \sigma'(z^{(i)})$$

- Intuition: Large weights in combination with dropout will cause predictions to vary widely => higher error => same effect as L2 reg.

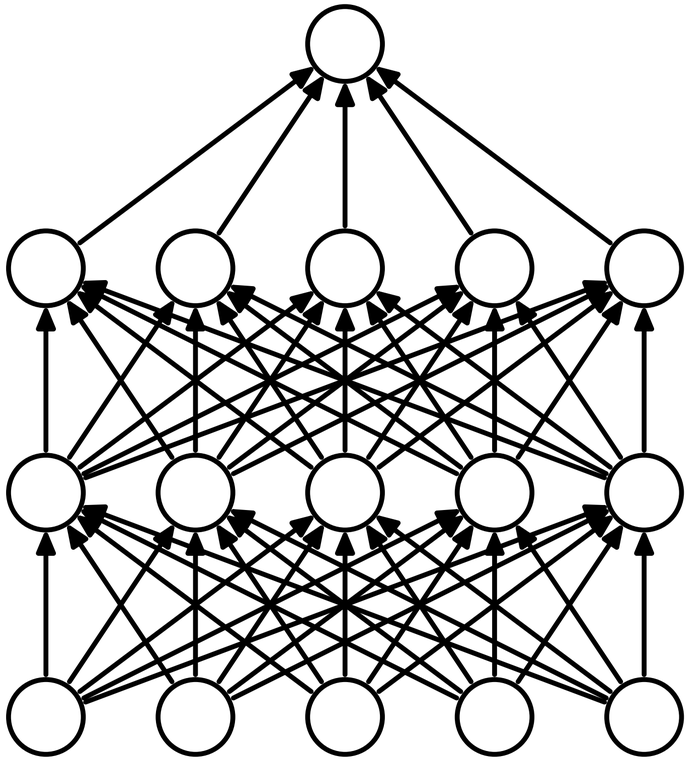
Training time only, not test time

- Compensate at test time by multiplying weights with $(1-\rho)$

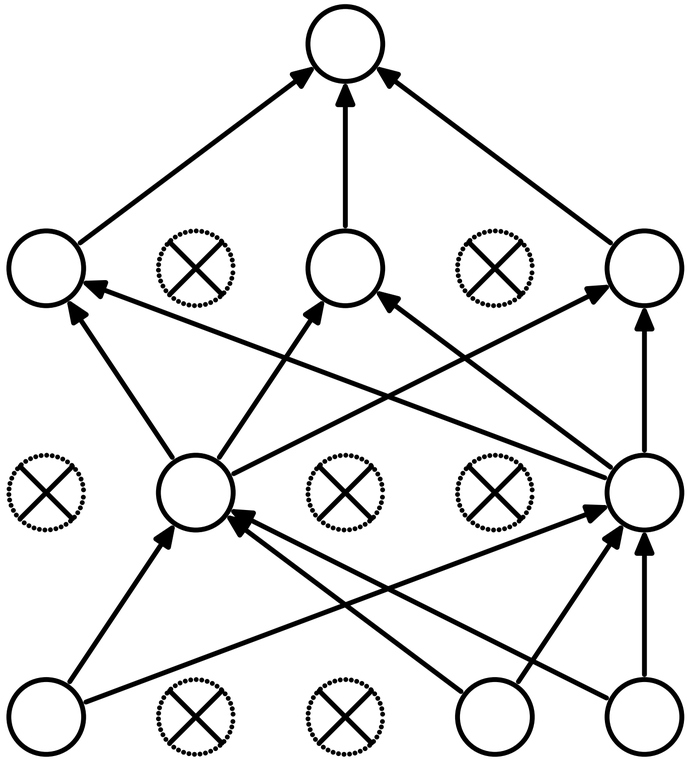




STOCHASTIC REGULARIZATION: DROPOUT



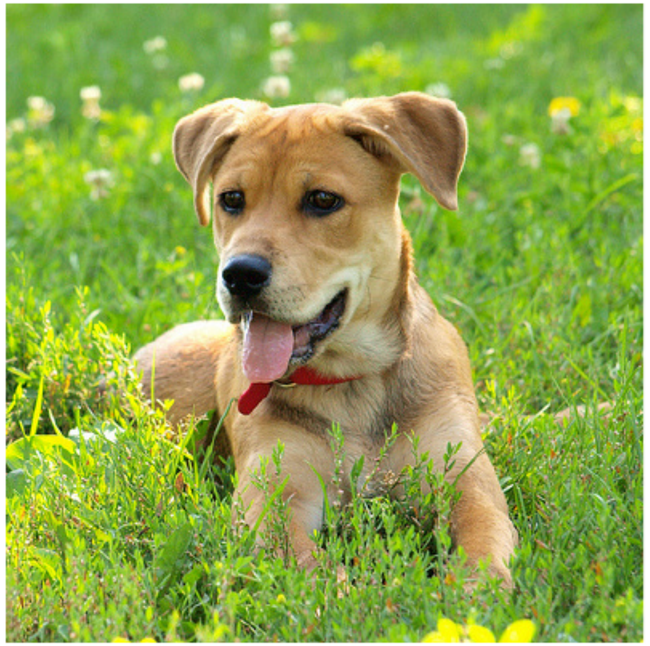
(a) Standard Neural Net



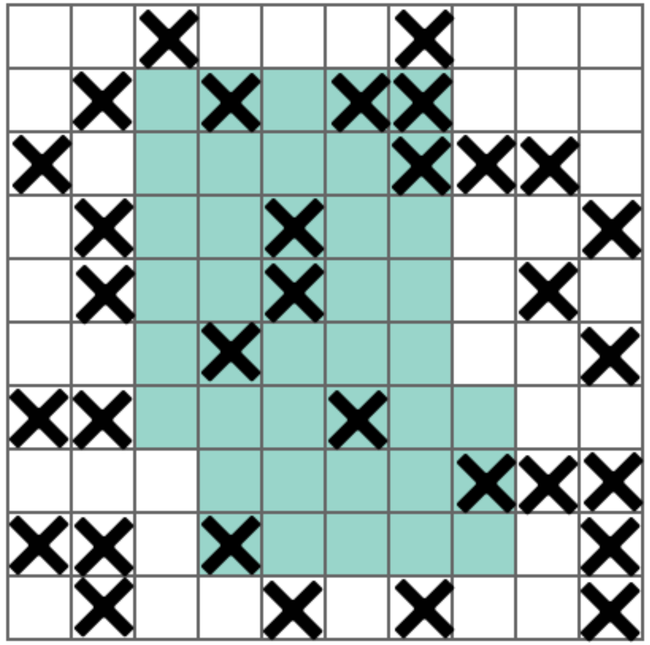
(b) After applying dropout.

Srivastava et al., "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", JMLR 2014

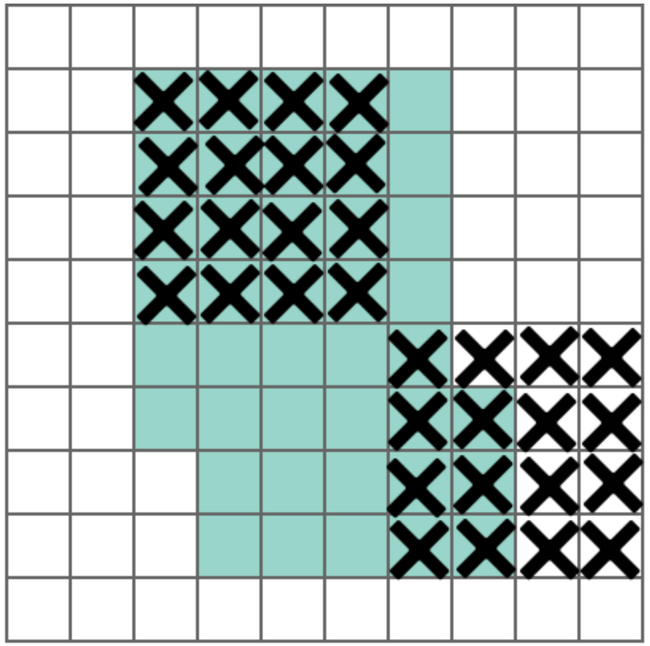
DROPBLOCK



Dropout



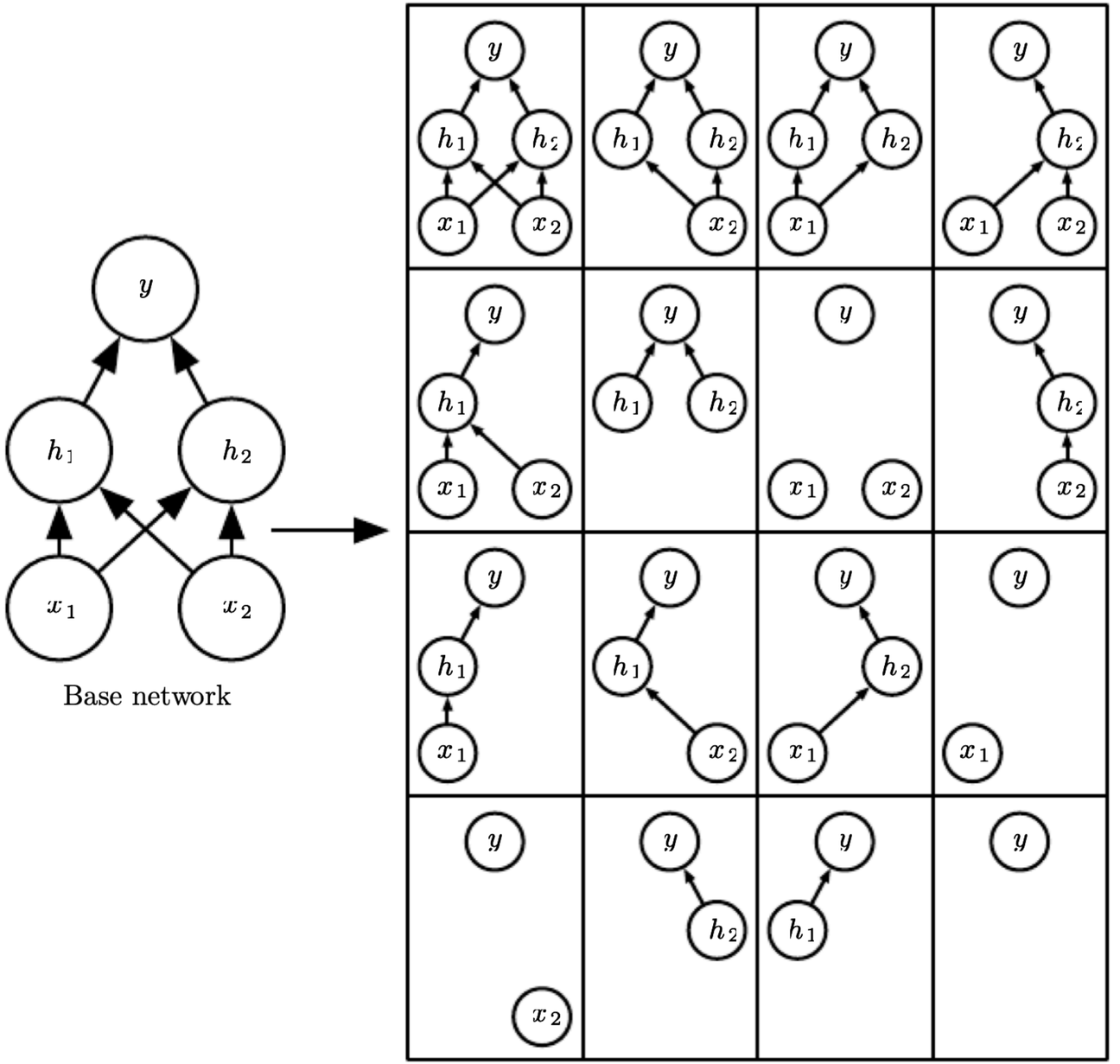
DropBlock



Dropblock extends dropout for convolutional networks, for whose dropout was shown to be less effective (spatial correlation can allow activations to flow through in spite of being dropped out somewhere)

Ghiasi et al., "DropBlock: A Regularization Method for Convolutional Networks", NIPS 2018

Interpret dropout as transforming one single network into an ensemble



Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016

WRAPPING UP

SUMMARY

Regularization is of prime concern in ML (only rivaled by optimization)

- In general: any modification of a learning method that reduces its generalization error but not the training error

Different regularization methods show a surprising amount of similar interpretations

- Ensemble <=> dropout <=> data augmentation, early stopping <=> shrinking <=> bottleneck

Stochastic regularization is a very powerful technique, but poorly understood

- See also SGD
- See also Batch Normalization (next lecture)

Sneak preview: often model compression (pruning, quantization) introduces stochasticity, which in turn improves generalization

- Models get smaller and even more accurate :)