Deep Learning

Lecture 6 – Bias, variance and regularization

UPPSALA
UNIVERSITET

Niklas Wahlström
Division of Systems and Control
Department of Information Technology
Uppsala University

Email: niklas.wahlstrom@it.uu.se

Summary Lecture 5 (I/III) - Lenet

1998

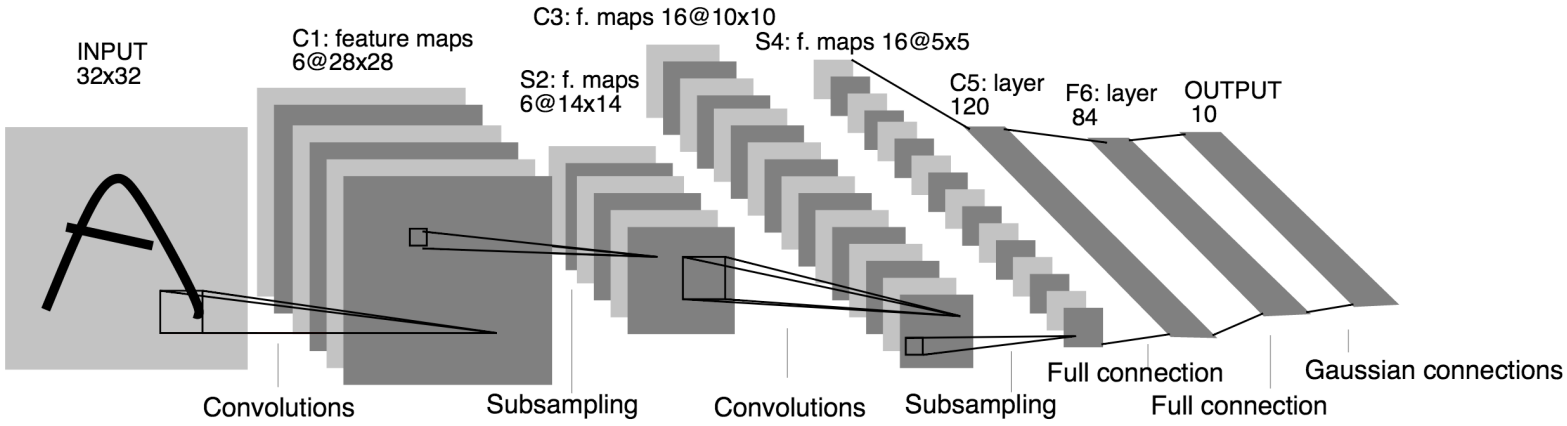


Figure: Src. Yann LeCun, et al, Gradient-based learning applied to document recognition, 1998

UPPSALA UNIVERSITET

Summary Lecture 5 (II/III) - Googlenet

2015

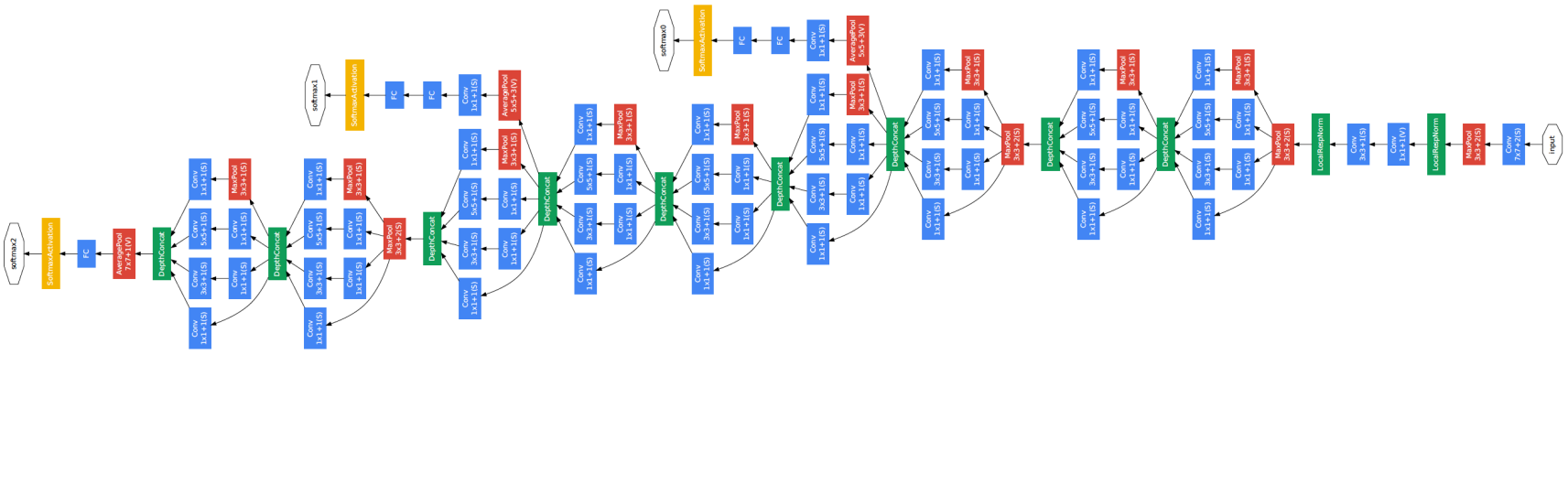
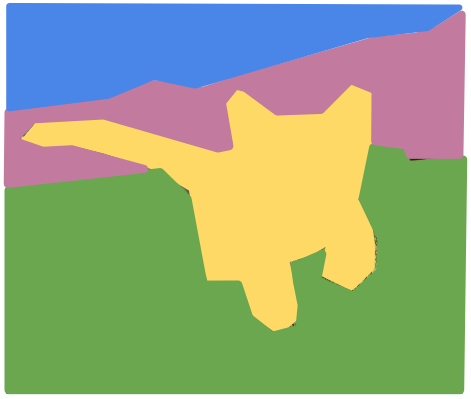


Figure: Src. Going deeper with convolutions, Szegedy et al., 2015

Other Computer Vision Tasks

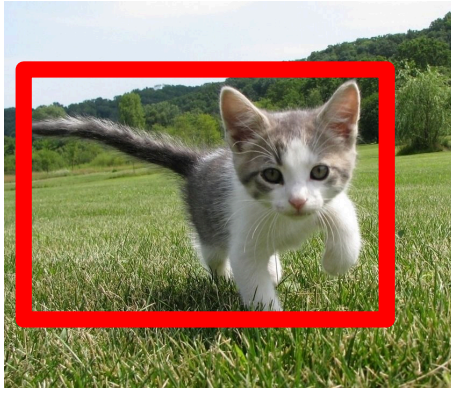
Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

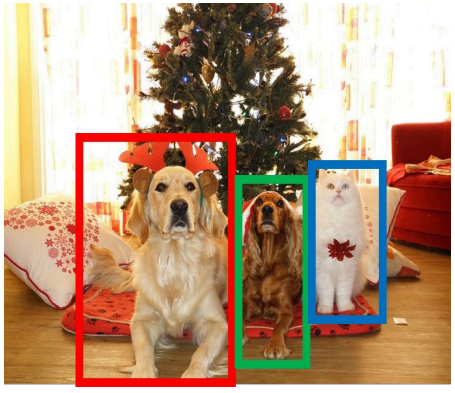
Classification + Localization



CAT

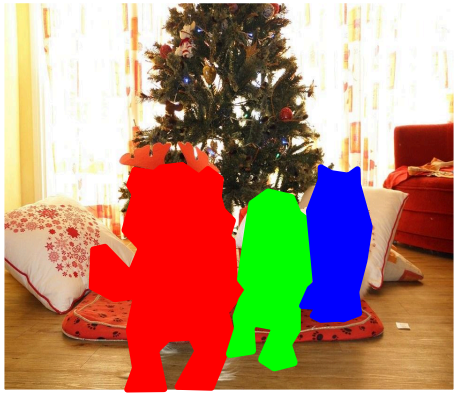
Single Object

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

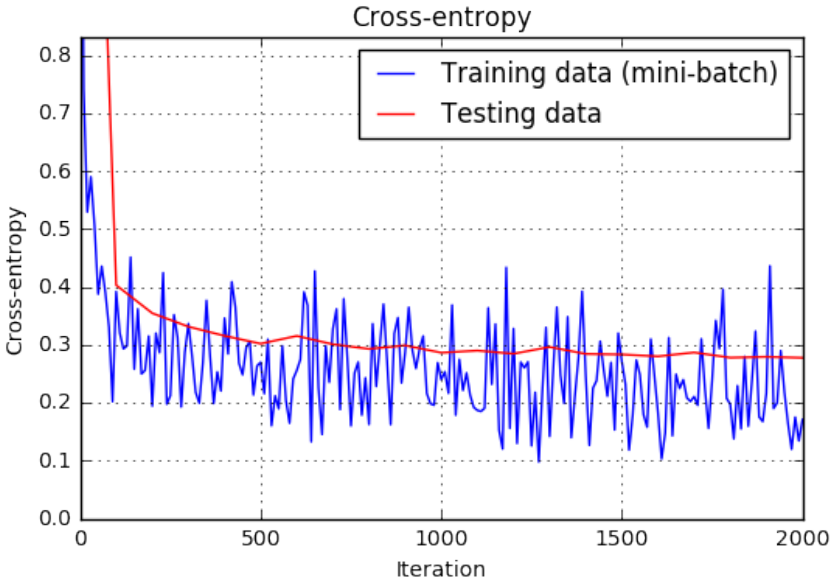
Multiple Object

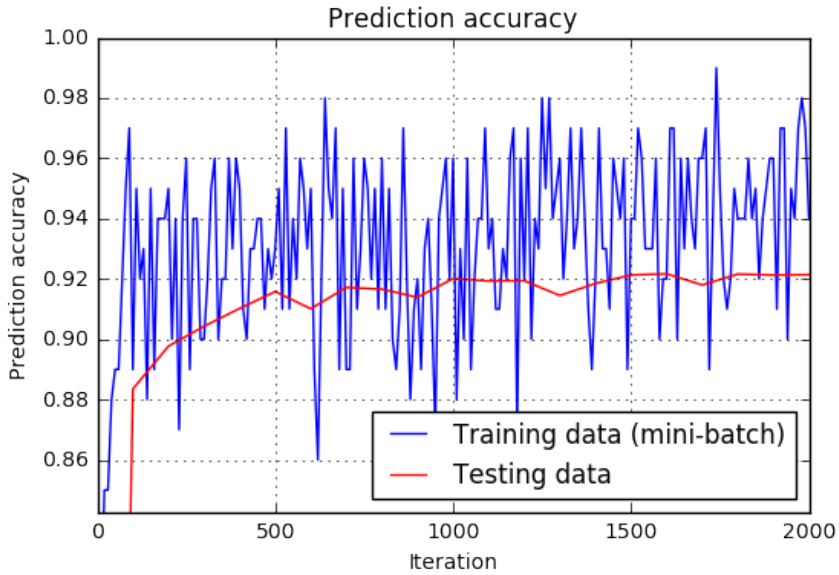
Some reflections on Hand-in assignments 1

One layer neural network (softmax regression)

Trained for 10'000 iterations. SGD with learning rate: $\gamma = 0.5$

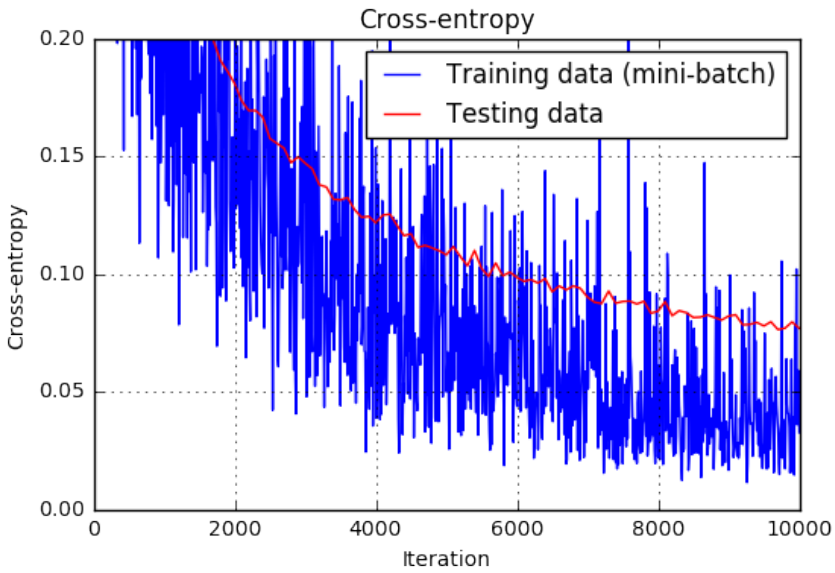
Mini-batch size: 100

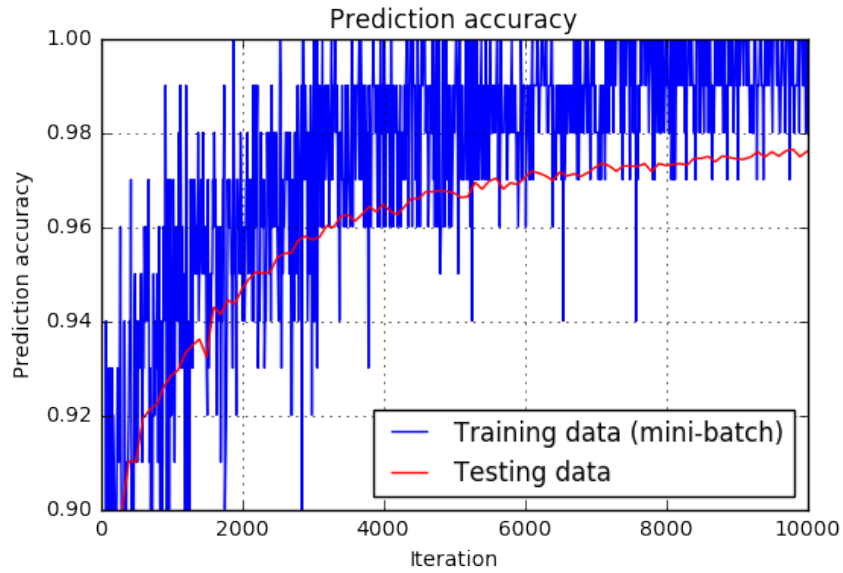




Some reflections on Hand-in assignments 1

Two layer neural network with sigmoid activation function.
Trained for 10'000 iterations.
SGD with learning rate: $\gamma = 0.5$

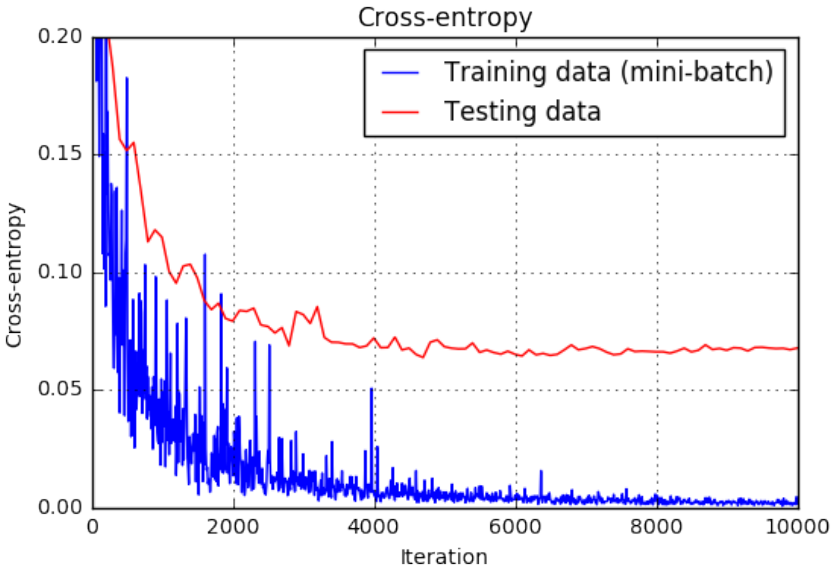


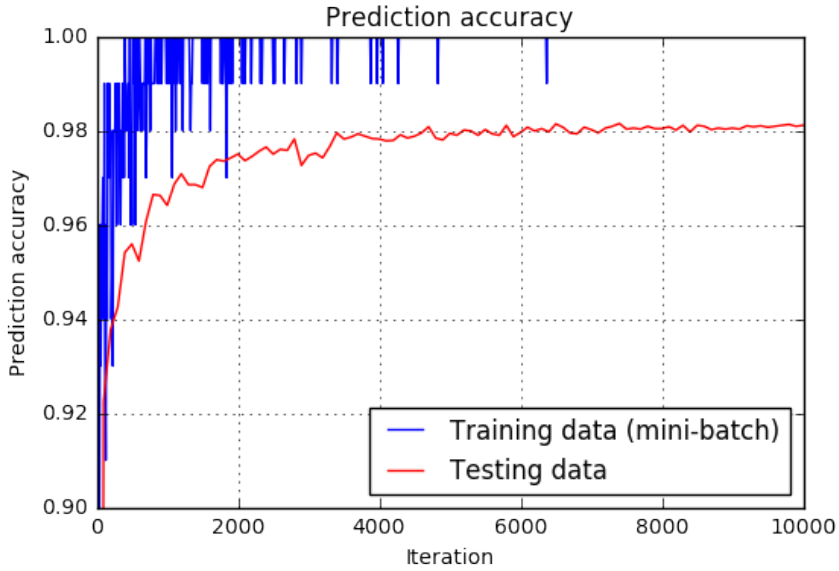


Convergence slow, not yet converged.

Some reflections on Hand-in assignments 1

Two layer neural network with relu activation function.
Trained for 10'000 iterations.
SGD with learning rate: $\gamma = 0.5$

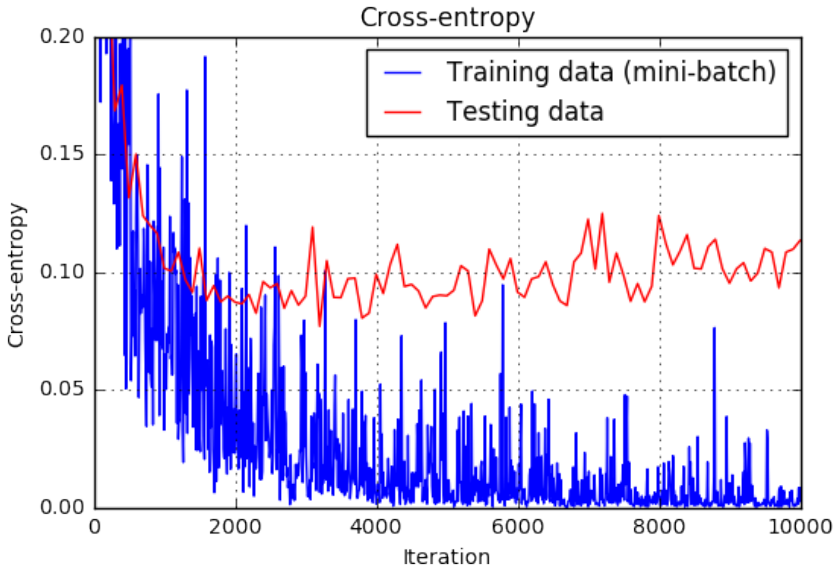


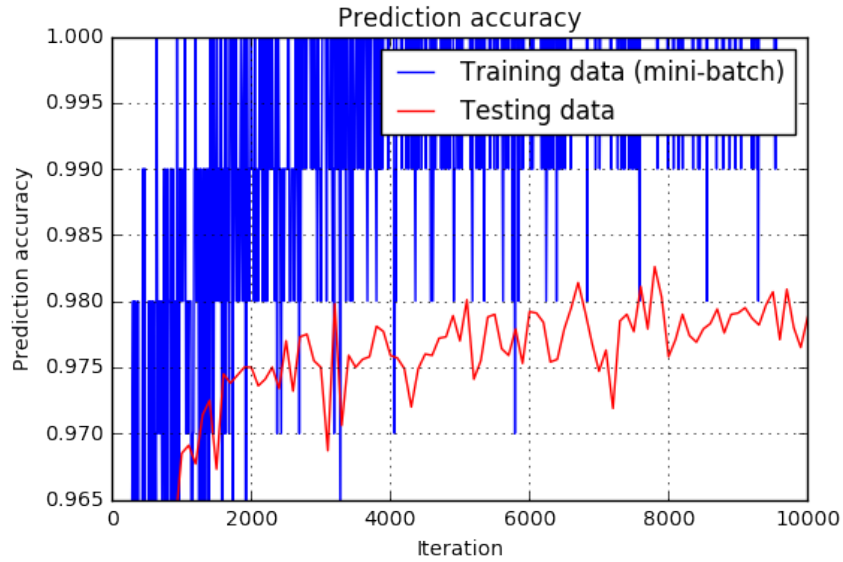


It trains much faster!!

Some reflections on Hand-in assignments 1

Five layer neural network with constant learning rate.
Trained for 10'000 iterations.
Adam with learning rate: $\gamma = 0.002$

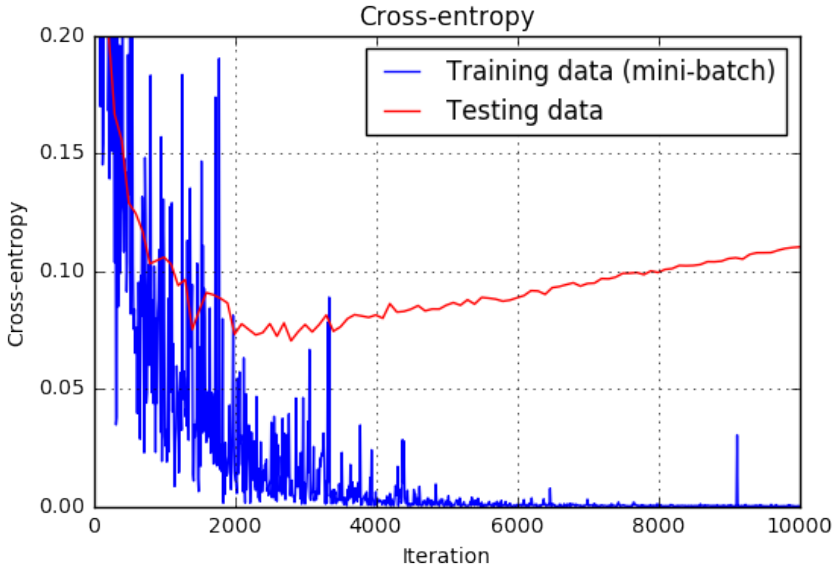


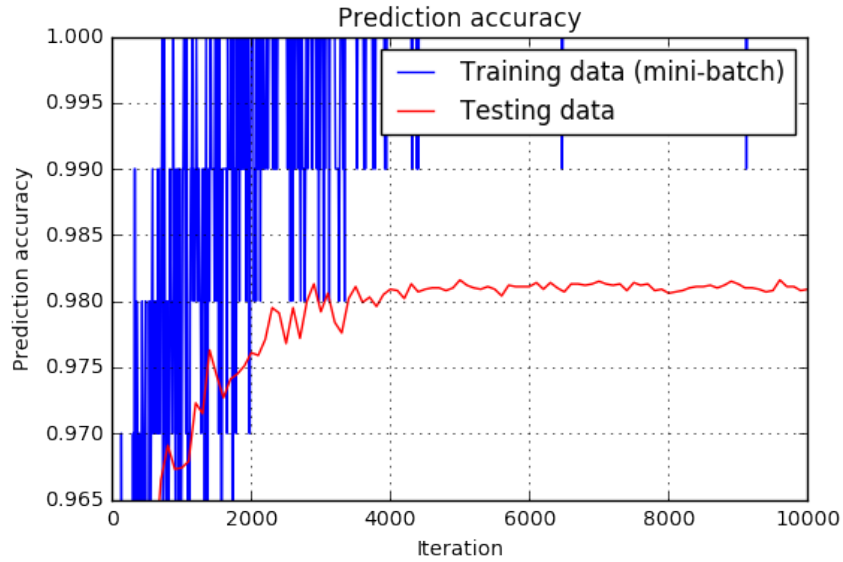


Cost function jumps back and forth in the end of training.
Decreasing learning rate would be an option but training would take longer (more iterations).

Some reflections on Hand-in assignments 1

Five layer neural network with adaptive learning rate.
Trained for 10'000 iterations.
Adam with learning rate: $\gamma_t = 0.003$ to $\gamma_t = 0.0001$

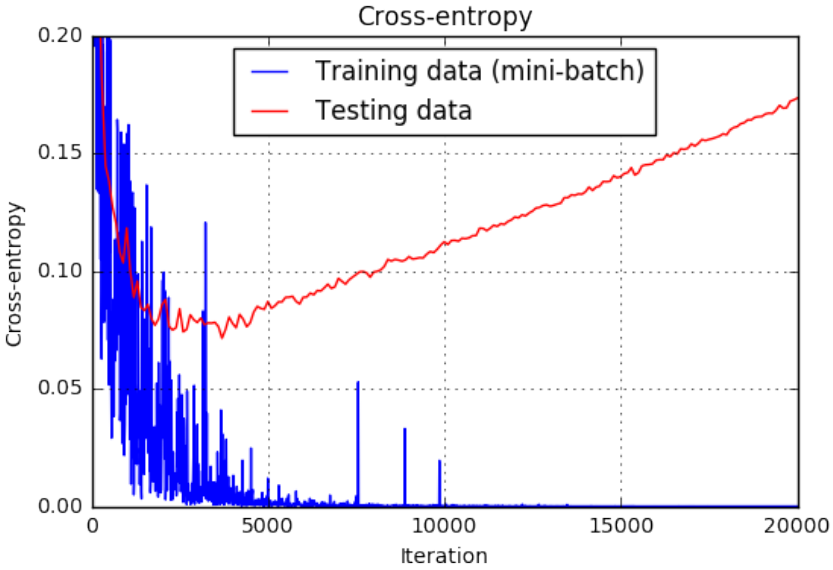


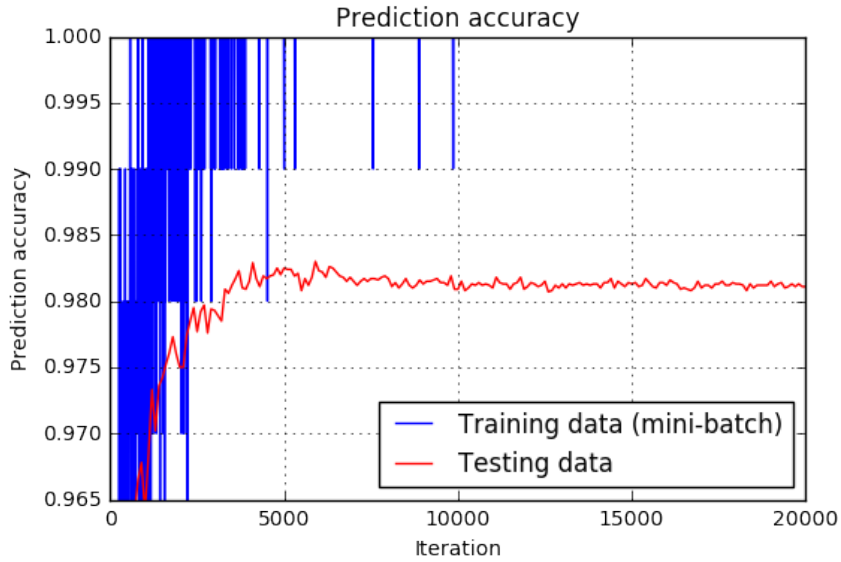


Starting fast, slowing down closer when we optimum. It indeed improves prediction accuracy!!
Why is cross-entropy on test data increasing??? **Overfitting!!**

Some reflections on Hand-in assignments 1

Five layer neural network with adaptive learning rate.
Trained for 20'000 iterations.
Adam with learning rate: $\gamma_t = 0.003$ to $\gamma_t = 0.0001$





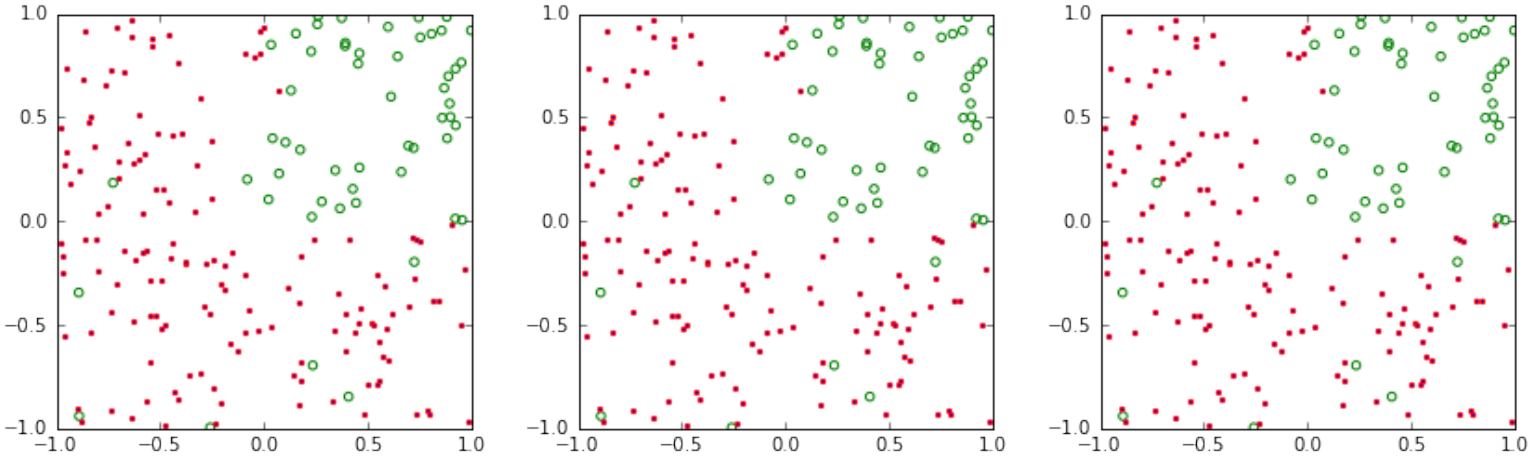
Starting fast, slowing down closer when we optimum. It indeed improves prediction accuracy!!
Why is cross-entropy on test data increasing??? **Overfitting!!**

Outline - Lecture 6

Outline:

1. Bias and variance, overfitting and underfitting
2. Regularization techniques for deep learning
    a. Weight decay
    b. Ensemble methods and bagging
    c. Dropout
    d. Data augmentation
    e. Early stopping

Bias and variance - classification problem

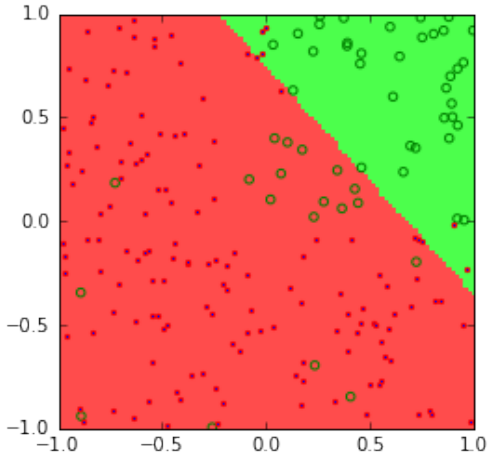


UPPSALA UNIVERSITET

Bias and variance - classification problem

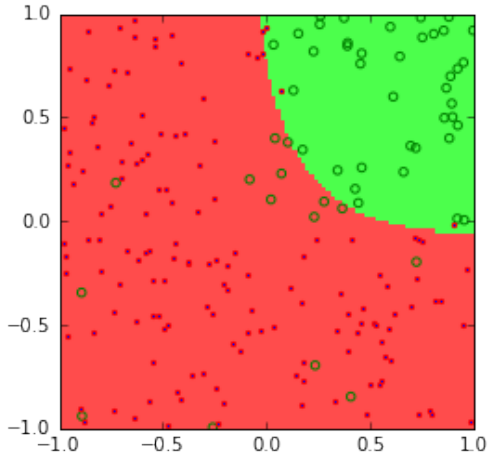
Assume that we can collect a new training data from the same distribution

Training dataset 1

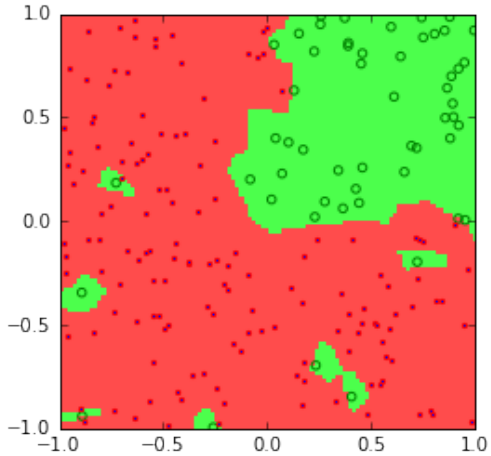


Model 1

Low model complexity



Model 2



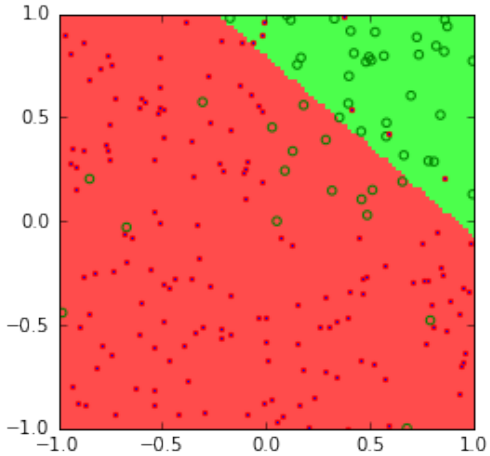
Model 3

High model complexity

Bias and variance - classification problem

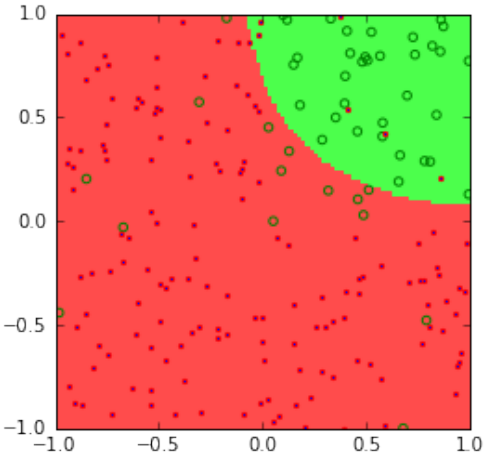
Assume that we can collect a new training data from the same distribution

Training dataset 2

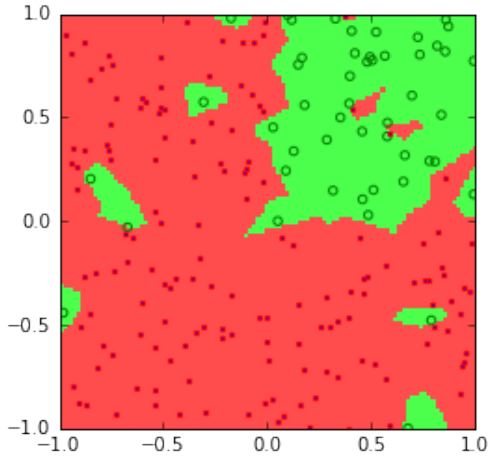


Model 1

Low model complexity



Model 2



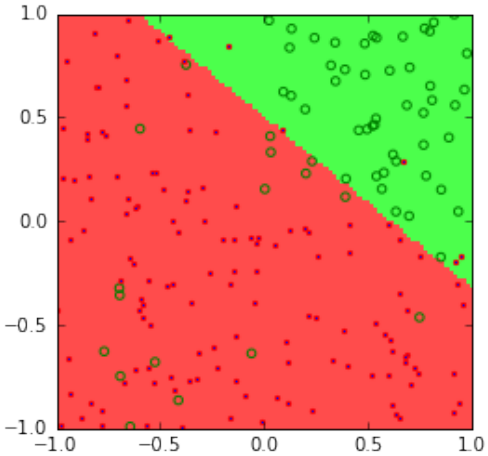
Model 3

High model complexity

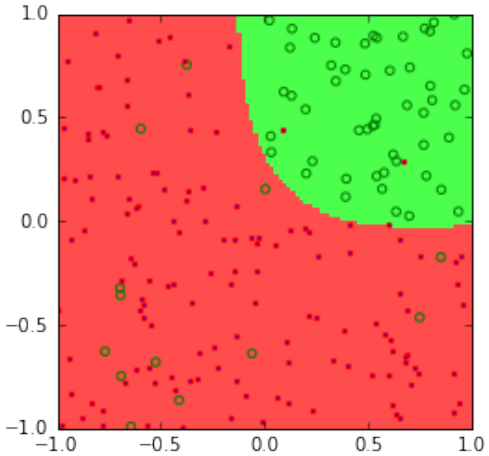
Bias and variance - classification problem

Assume that we can collect a new training data from the same distribution

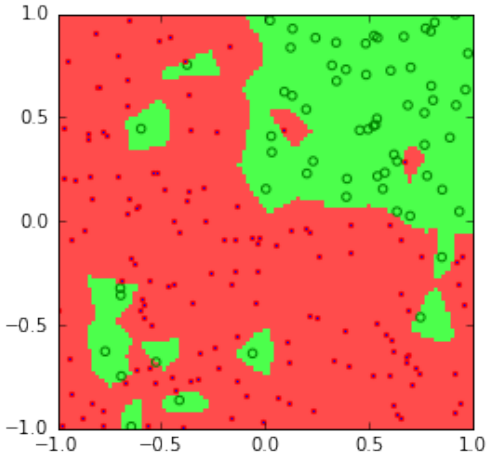
Training dataset 3



Model 1
Low model complexity
High bias
Low variance
Underfitting



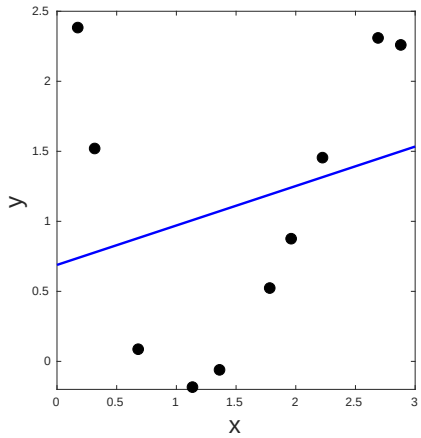
Model 2
"Just right"



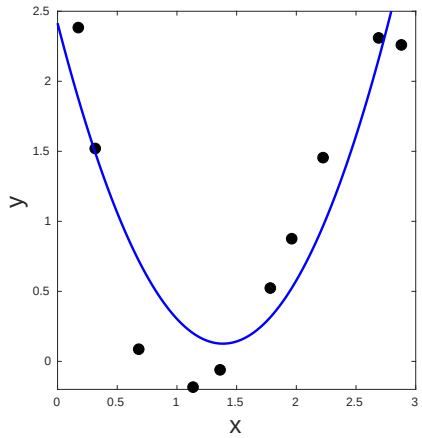
Model 3
High model complexity
High variance
Low bias
Overfitting

Bias and variance - regression problem

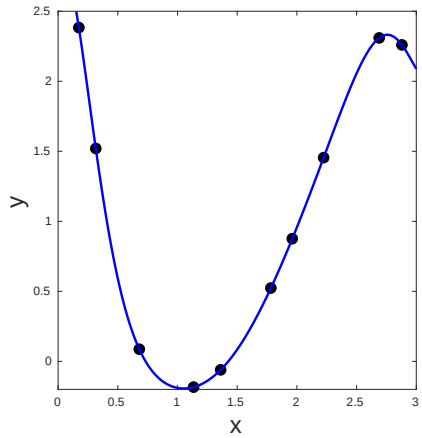
Training dataset 1



Model 1
Low model complexity



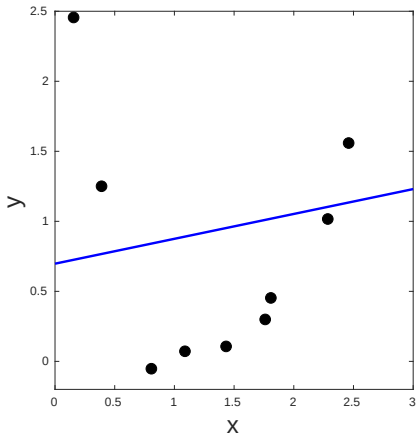
Model 2



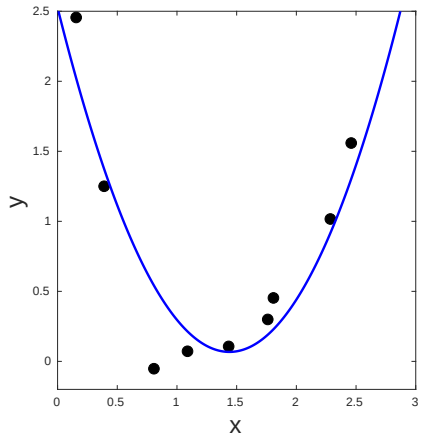
Model 3
High model complexity

Bias and variance - regression problem

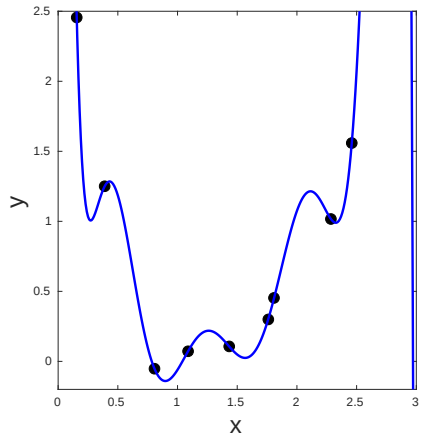
Training dataset 2



Model 1
Low model complexity



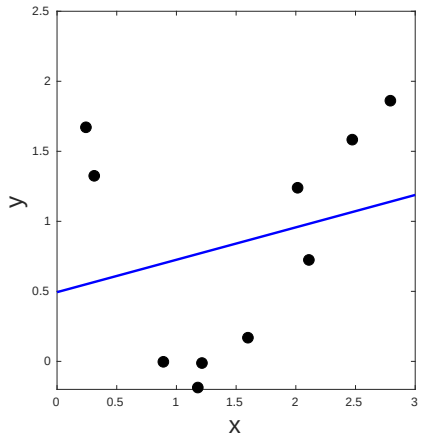
Model 2



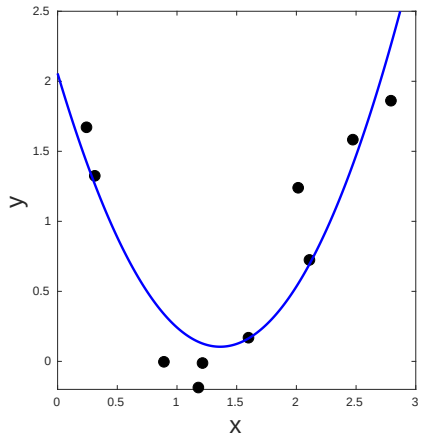
Model 3
High model complexity

Bias and variance - regression problem

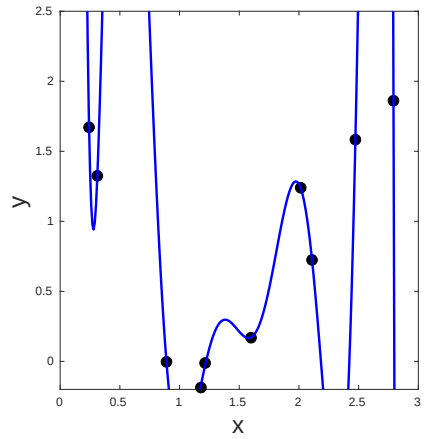
Training dataset 3



Model 1
Low model complexity



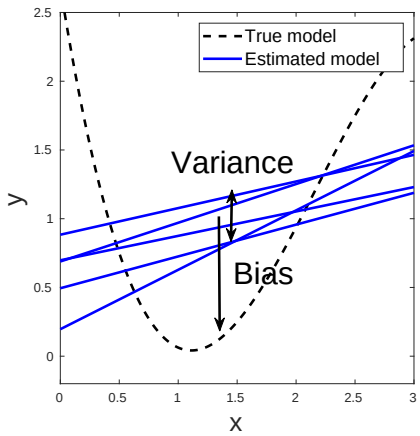
Model 2

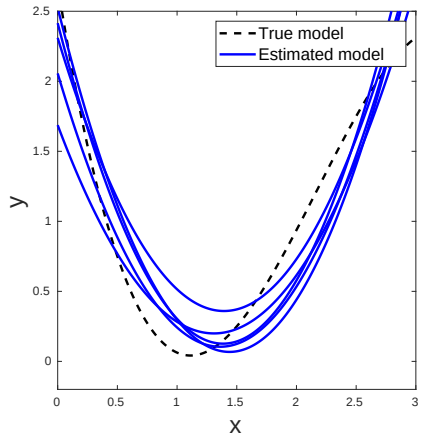


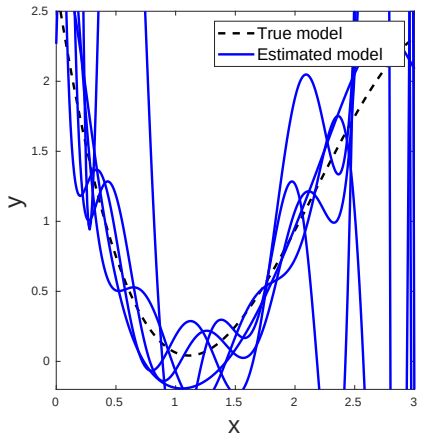
Model 3
High model complexity

Bias and variance - regression problem

Models from all training datasets







Model 1	Model 2	Model 3
Low model complexity		High model complexity
High bias		High variance
Low variance	"Just right"	Low bias
Underfitting		Overfitting

Let's generalize using math

- Training data $\mathcal{T} \triangleq \{\mathbf{x}_i, y_i\}_{i=1}^n$
- Prediction $\widehat{y}(\mathbf{x}; \mathcal{T})$, output from a method learned using $\mathcal{T}$
- The distribution of inputs and outputs in the real world $p(\mathbf{x}, y)$
- True model: $y = f(\mathbf{x}) + \epsilon$, where ϵ has mean 0 and variance σ^2.

Bias-variance decomposition I/III

Consider a test input $\mathbf{x}_\star$ and denote $\widehat{y} = \widehat{y}(\mathbf{x}_\star; \mathcal{T})$, $f = f(\mathbf{x}_\star)$, and $y = f + \epsilon$. The expected mean square error of $\widehat{y}$ can be decomposed as

$$\begin{aligned}
\mathbb{E}_{\mathcal{T}}\left[(\widehat{y} - y)^2\right] &= \mathbb{E}_{\mathcal{T}}\left[(\widehat{y} - f - \epsilon)^2\right] \\
&= \mathbb{E}_{\mathcal{T}}\left[\left((\widehat{y} - \mathbb{E}_{\mathcal{T}}\left[\widehat{y}\right]) + (\mathbb{E}_{\mathcal{T}}\left[\widehat{y}\right]) - f\right) + \epsilon\right)^2\right] \\
&= \underbrace{\mathbb{E}_{\mathcal{T}}\left[(\widehat{y} - \mathbb{E}_{\mathcal{T}}\left[\widehat{y}\right])^2\right]}_{\text{Variance}} + \underbrace{(\mathbb{E}_{\mathcal{T}}\left[\widehat{y}\right]) - f)^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}_{\mathcal{T}}\left[\epsilon^2\right]}_{\text{Irreducible error}}
\end{aligned}$$

This decomposition is only valid for $\mathbf{x}_\star$. Average over all test data points as well

$$\underbrace{\mathbb{E}_\star\left[\mathbb{E}_{\mathcal{T}}\left[(\widehat{y} - y)^2\right]\right]}_{\substack{\text{Expected MSE} \\ \text{(avaraged over } \mathbf{x}_\star\text{)}}} = \underbrace{\mathbb{E}_\star\left[\mathbb{E}_{\mathcal{T}}\left[(\widehat{y} - \mathbb{E}_{\mathcal{T}}\left[\widehat{y}\right])^2\right]\right]}_{\substack{\text{Variance} \\ \text{(avaraged over } \mathbf{x}_\star\text{)}}} + \underbrace{\mathbb{E}_\star\left[(\mathbb{E}_{\mathcal{T}}\left[\widehat{y}\right]) - f)^2\right]}_{\substack{\text{Bias}^2 \\ \text{(avaraged over } \mathbf{x}_\star\text{)}}} + \underbrace{\sigma^2}_{\substack{\text{Irreducible} \\ \text{error}}}$$

$\mathbb{E}_\star$ = *expected value over all test inputs* $\mathbf{x}_\star$
$\mathbb{E}_{\mathcal{T}}$ = *expected value over all training datasets* $\mathcal{T}$

Bias-variance decomposition II/III

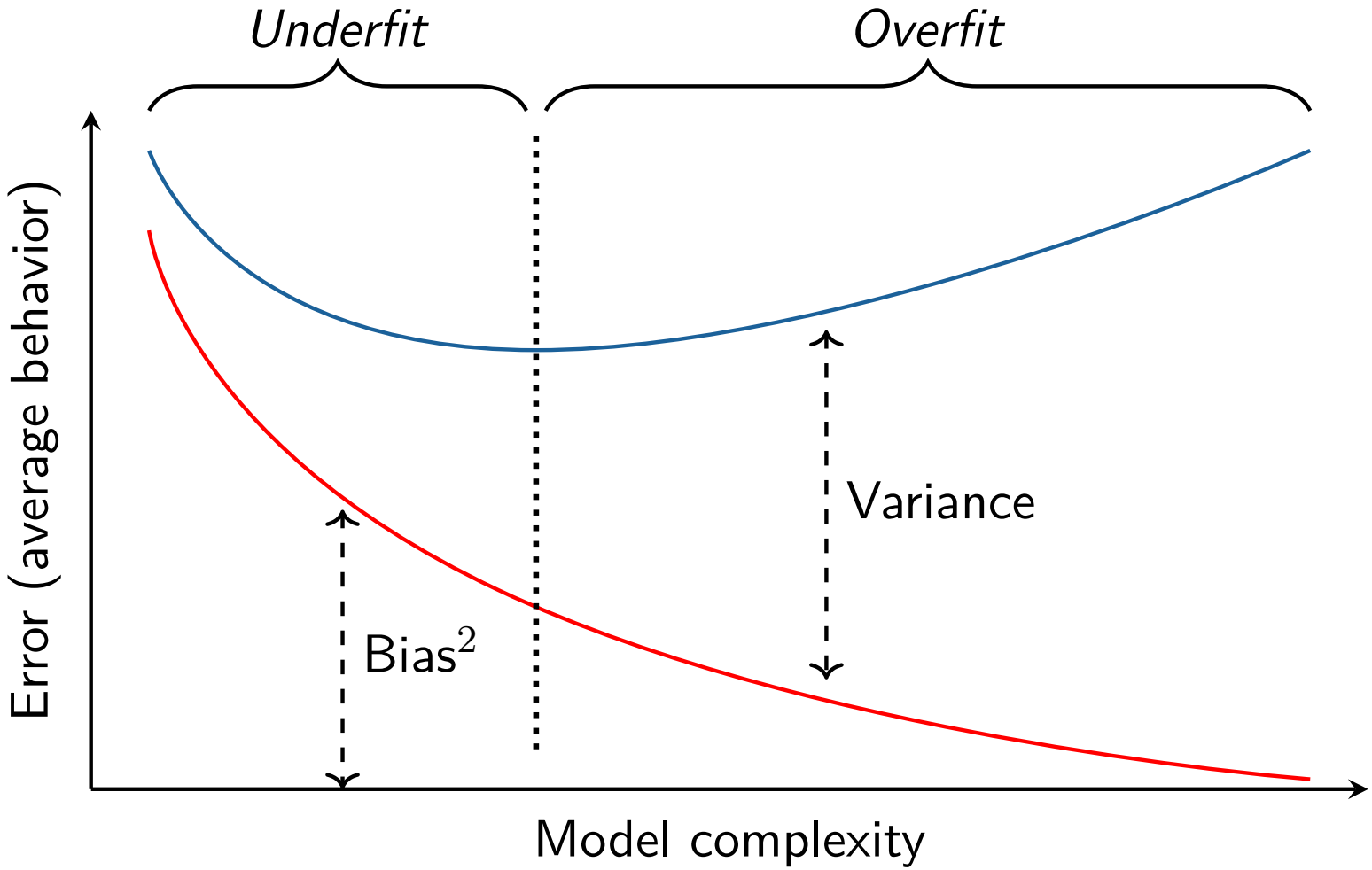
Technical interpretation:

- Bias^2: The part of expected MSE that is due to the fact that, no matter how much training data is used, the model cannot represent the true f
- Variance: The part of expected MSE that is due to the variance in the training dataset

Intuitive interpretation:

- Bias: The inability of a method to describe the complicated patterns we would like it to describe. Low model complexity.
- Variance: How sensitive a method is to the training data. High model complexity.

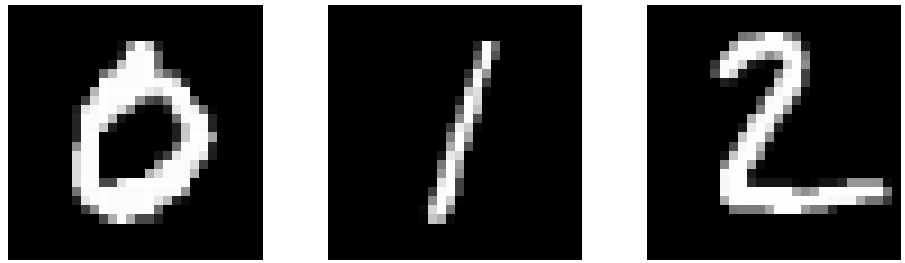
Bias-variance decomposition III/III



Bias and variance - MNIST

- Unfortunately, bias and variance are difficult to compute based on its formal definitions.
- Instead, we can monitor bias and variance from training data error and test data error.

Example MNIST:



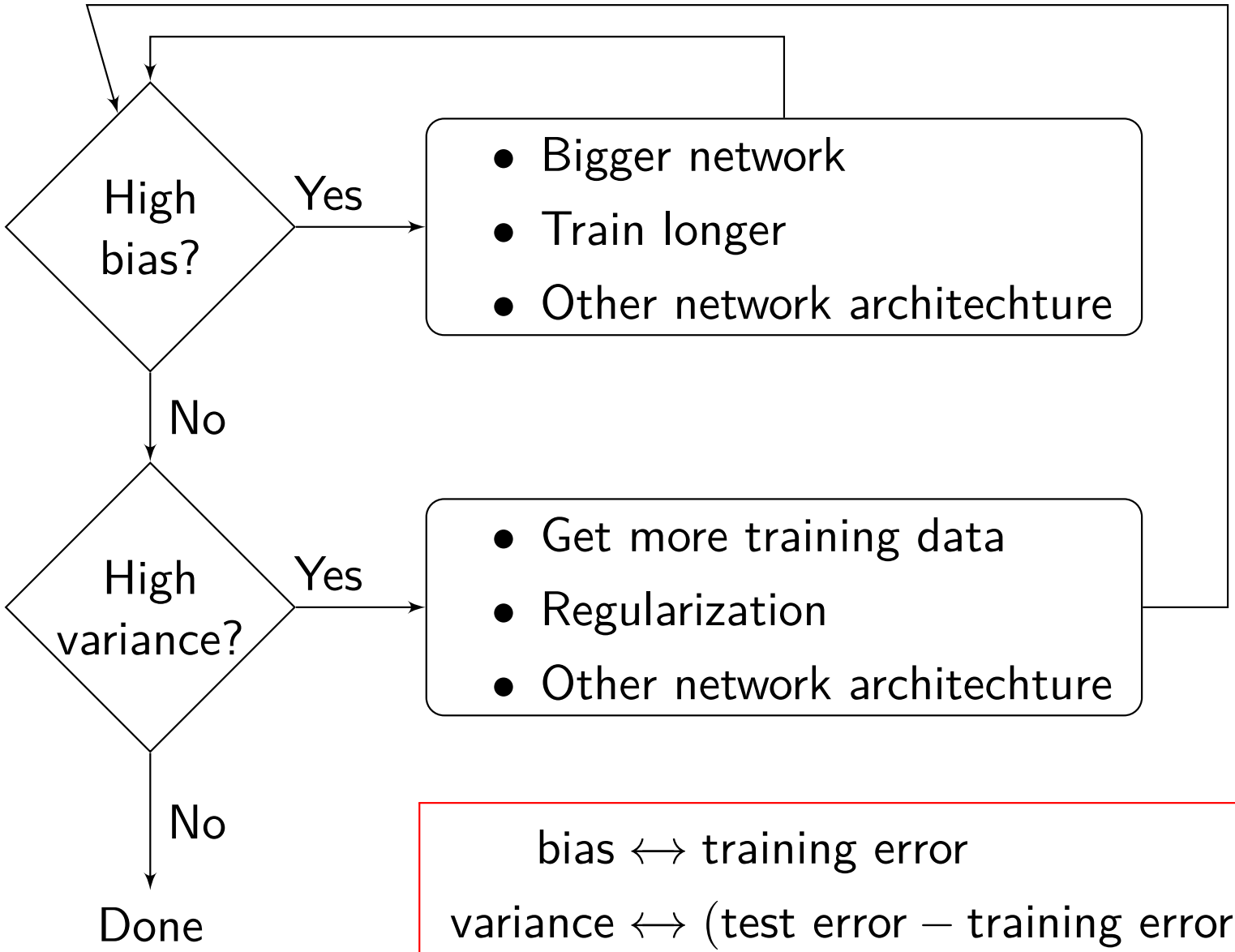
...

Training data error	Test data error	
9%	10%	High bias, low variance
1%	8%	Low bias, high variance
1%	2%	Low bias, low variance
10%	20%	High bias, high variance

$$\text{bias} \leftrightarrow \text{training error}$$

$$\text{variance} \leftrightarrow (\text{test error} - \text{training error})$$

Basic recipe of machine learning

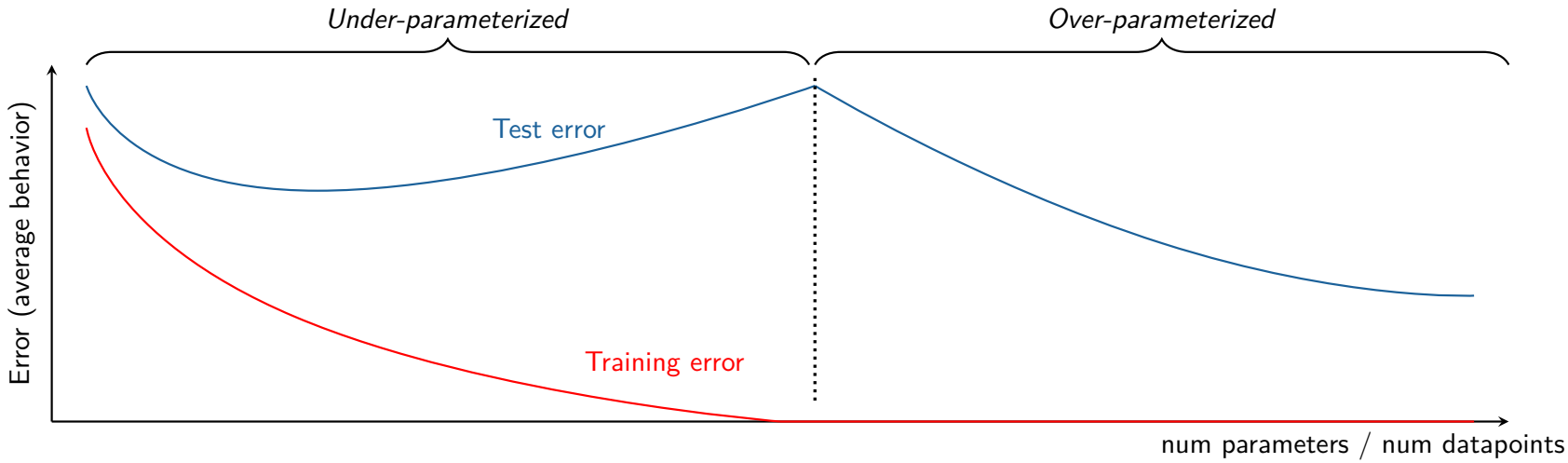


$$\text{bias} \longleftrightarrow \text{training error}$$

$$\text{variance} \longleftrightarrow (\text{test error} - \text{training error})$$

Overparameterized ML models

- Recent research has in part challenged the view of generalization
- Overparameterized ML models can get virtually zero training error and still generalize well to unseen test data.
- Some theoretical results exist, much is left to be understood.



If you are interested colleagues of mine are running a seminar course in the fall on this topic
`http://www.it.uu.se/research/systems_and_control/education/2021/overparameterized-ml`

Outline - Lecture 6

Outline:

1. Bias and variance, overfitting and underfitting
2. Regularization
    a. Weight decay
    b. Ensemble methods and bagging
    c. Dropout
    d. Data augmentation
    e. Early stopping

Weight decay

Regularize using a penalty on the Euclidean norm

$$\tilde{J}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \frac{\lambda}{2n}\|\boldsymbol{\theta}\|_2^2.$$

$\boldsymbol{\theta}$ - all parameters. May exclude offset terms.

- L2-regularization: $\frac{\lambda}{2n}\|\boldsymbol{\theta}\|_2^2 = \frac{\lambda}{2n}\sum_{j=1}\theta_j^2$
- L1-regularization: $\frac{\lambda}{n}\|\boldsymbol{\theta}\|_1 = \frac{\lambda}{n}\sum_{j=1}|\theta_j|$

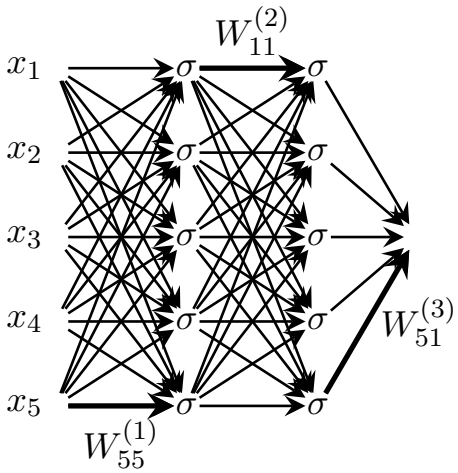
New gradient:

$$\frac{d\tilde{J}(\boldsymbol{\theta})}{d\boldsymbol{\theta}} = \frac{dJ(\boldsymbol{\theta})}{d\boldsymbol{\theta}} + \frac{\lambda}{n}\boldsymbol{\theta}.$$

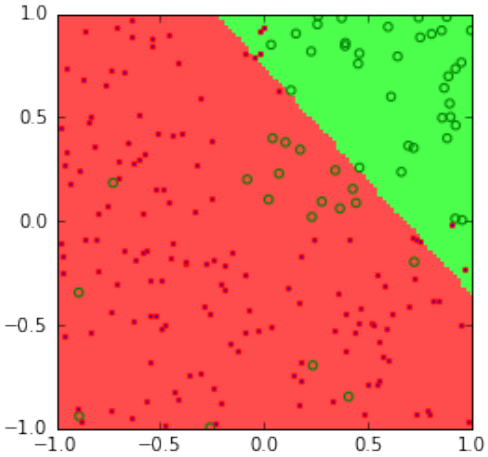
New update:

$$\begin{aligned}\boldsymbol{\theta} &:= \boldsymbol{\theta} - \gamma\frac{d\tilde{J}(\boldsymbol{\theta})}{d\boldsymbol{\theta}} \\ &= \boldsymbol{\theta} - \gamma\left(\frac{dJ(\boldsymbol{\theta})}{d\boldsymbol{\theta}} + \frac{\lambda}{n}\boldsymbol{\theta}\right) \\ &= \underbrace{\left(1 - \frac{\gamma\lambda}{n}\right)}_{<1}\boldsymbol{\theta} - \gamma\frac{dJ(\boldsymbol{\theta})}{d\boldsymbol{\theta}}\end{aligned}$$

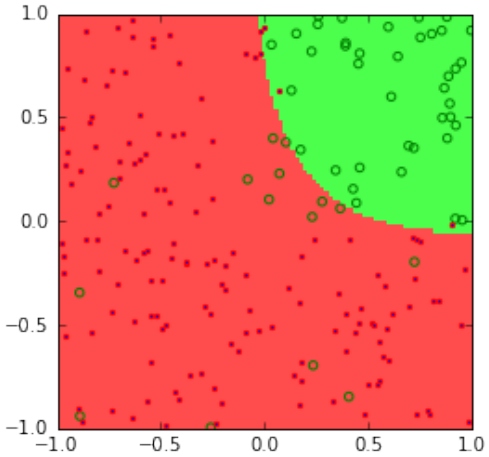
Why does weight decay reduce overfitting ?



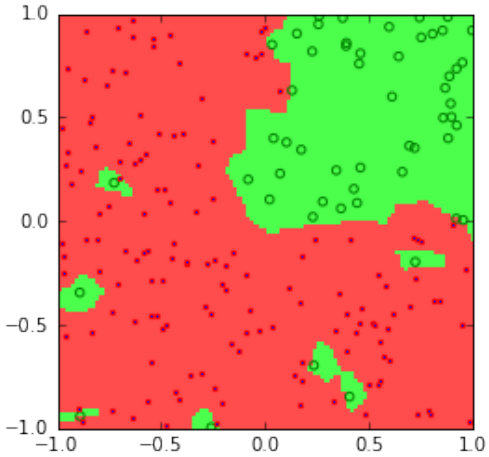
$$\tilde{J}(\boldsymbol{\theta}) = J(\boldsymbol{\theta})$$



High bias
Low model complexity
Small network

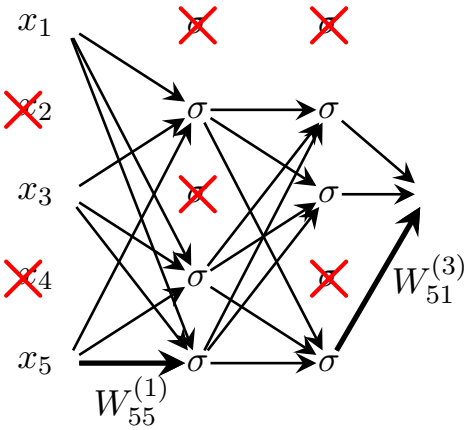


Just right

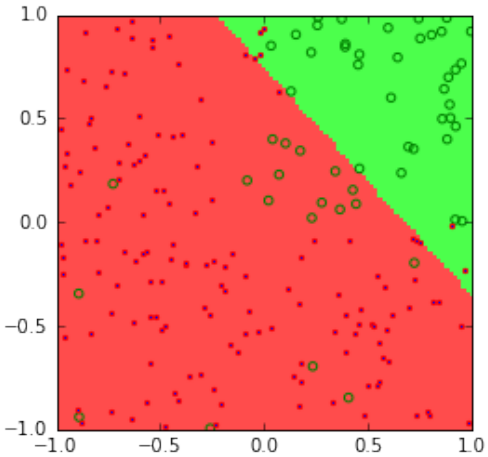


High variance
High model complexity
Big network

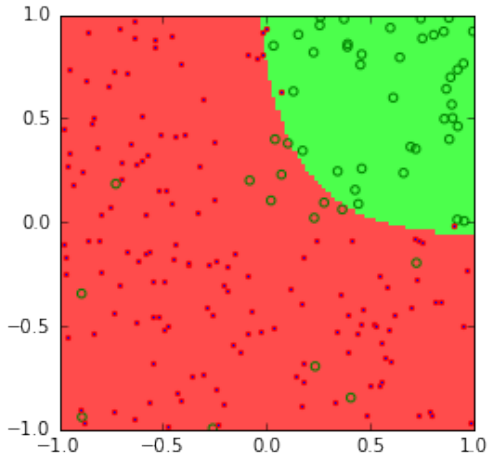
Why does weight decay reduce overfitting ?



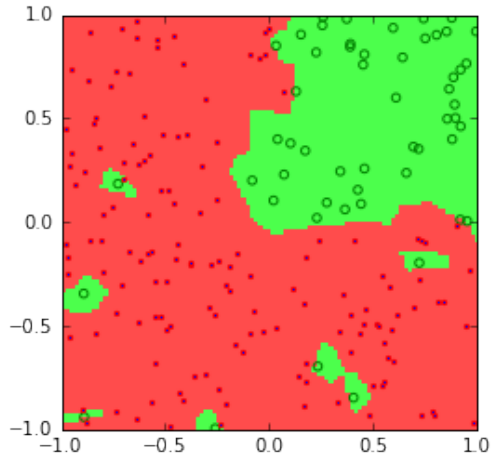
$$\tilde{J}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \frac{\lambda}{2n}\|\boldsymbol{\theta}\|_2^2$$
$$\Rightarrow$$
$$\boldsymbol{\theta} \approx 0$$



High bias
Low model complexity
Small network



Just right

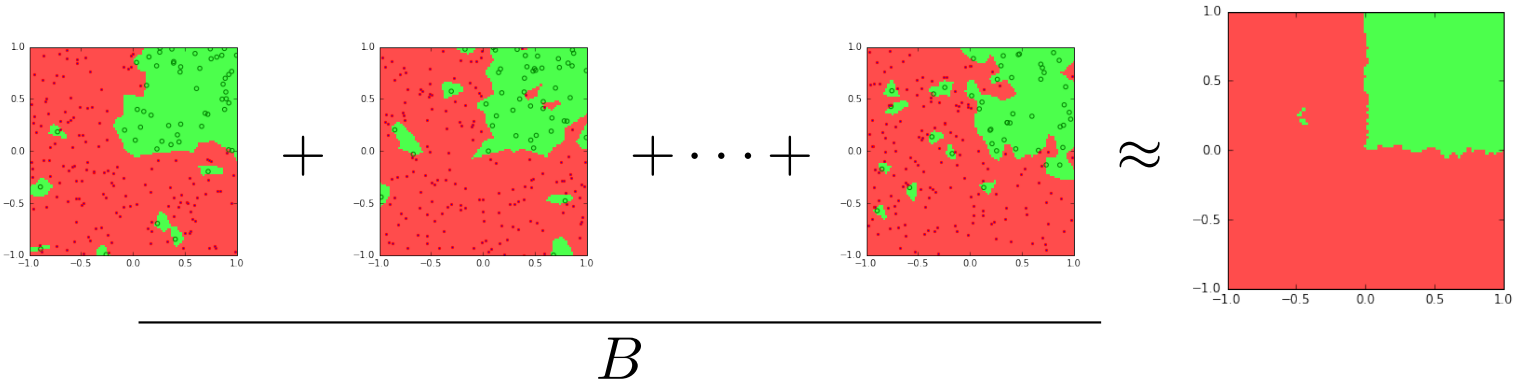


High variance
High model complexity
Big network

Ensamble methods

Ensamble methods

- **Idea** Train several models separately. Have all models vote on the output for test data
- **Assumption** Different models will not make same errors on test data
- Is useful if you have multiple high-variance, low-bias models.



$$\frac{+ \quad + \quad \cdots \quad +}{B} \approx$$

Ensamble methods - Variance by averaging

- $\hat{y}_1, \ldots, \hat{y}_B$ be predictions from B different models.
- $\hat{y}_1, \ldots, \hat{y}_B$ are identically distributed (but possibly dependent)
- $\mathbb{E}[\hat{y}_i] = \mu, \qquad \mathrm{Var}[\hat{y}_i^2] = \sigma^2, \qquad \mathrm{corr}[\hat{y}_i, \hat{y}_j] = \rho$

Then

$$\mathbb{E}\left[\frac{1}{B}\sum_{i=1}^{B}\hat{y}_i\right] = \mu,$$

$$\mathrm{Var}\left[\frac{1}{B}\sum_{i=1}^{B}\hat{y}_i\right] = \frac{1-\rho}{B}\sigma^2 + \rho\sigma^2.$$

- If $\rho = 1$ model averaging does not reduce variance, still σ^2
- If $\rho = 0$ variance is reduced to $\frac{1}{B}\sigma^2$.
- Does not affect bias regardless of ρ!

Model averaging reduces variance without hurting bias!

Ensamble methods - Bagging

- Optimal: Train ensemble members on different training datasets
- Not possible, since you only have one training dataset.
- To mimic this, bootstrap your training data!

bootstrap = sample data with replacement

Bagging

Training data

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_{10}$
$\mathbf{y}_1$	$\mathbf{y}_2$	$\mathbf{y}_3$	$\mathbf{y}_4$	$\mathbf{y}_5$	$\mathbf{y}_6$	$\mathbf{y}_7$	$\mathbf{y}_8$	$\mathbf{y}_9$	$\mathbf{y}_{10}$

- First resampled dataset

$\mathbf{x}_1$	$\mathbf{x}_3$	$\mathbf{x}_3$	$\mathbf{x}_5$	$\mathbf{x}_5$	$\mathbf{x}_5$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_9$
$\mathbf{y}_1$	$\mathbf{y}_3$	$\mathbf{y}_3$	$\mathbf{y}_5$	$\mathbf{y}_5$	$\mathbf{y}_5$	$\mathbf{y}_7$	$\mathbf{y}_8$	$\mathbf{y}_9$	$\mathbf{y}_9$

- Second resampled dataset

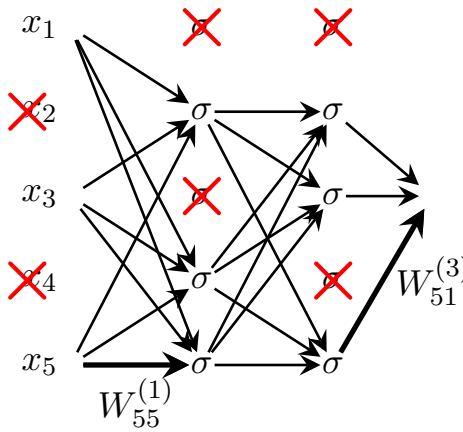
$\mathbf{x}_1$	$\mathbf{x}_3$	$\mathbf{x}_3$	$\mathbf{x}_5$	$\mathbf{x}_5$	$\mathbf{x}_5$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\mathbf{x}_9$	$\mathbf{x}_9$
$\mathbf{y}_1$	$\mathbf{y}_3$	$\mathbf{y}_3$	$\mathbf{y}_5$	$\mathbf{y}_5$	$\mathbf{y}_5$	$\mathbf{y}_7$	$\mathbf{y}_8$	$\mathbf{y}_9$	$\mathbf{y}_9$

- $\vdots$

Train an ensemble member on each of the resampled datasets.
Average their predictions.

Dropout

- Dropout is a regularization technique where we *during training* randomly drop units.
- The term "dropout" refers to dropping out units (hidden and visible) in a neural network.
- By dropping a unit out, we mean temporarily removing it from the network, along with all its incoming and outgoing connections.
- The choice of which units to drop is random.

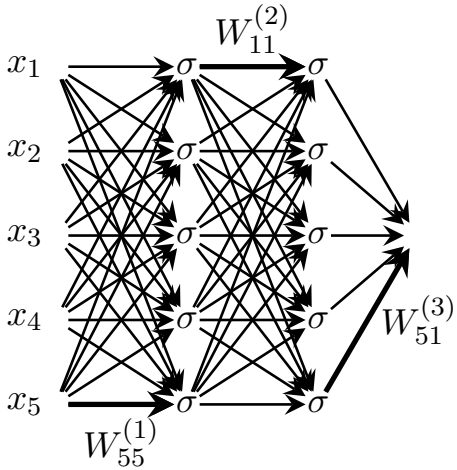


Srivastava, Nitish et al. (2014). "Dropout: A simple way to prevent neural networks from overfitting". In: The Journal of Machine Learning Research 15.1, pp. 1929-1958.

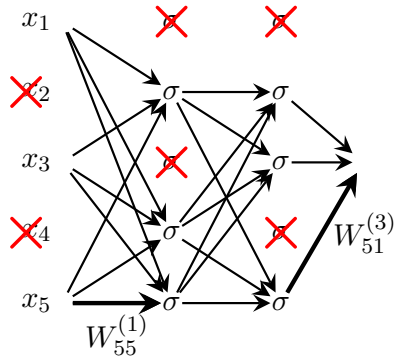
Dropout - main idea

Consider the following network to be trained

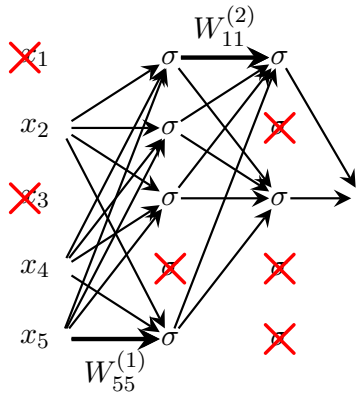
- **1st iter** Keep and update each unit with prob. p, drop remanding ones
- **2nd iter** Keep and update another random selection of units, drop remanding units.
- t**th iter** Continue in the same manner.
- **Test time** Use all units. Multiply weights by p.



1st train iter.

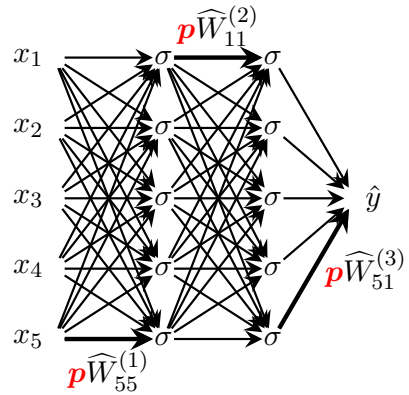


2nd train iter.



. . .

Test time

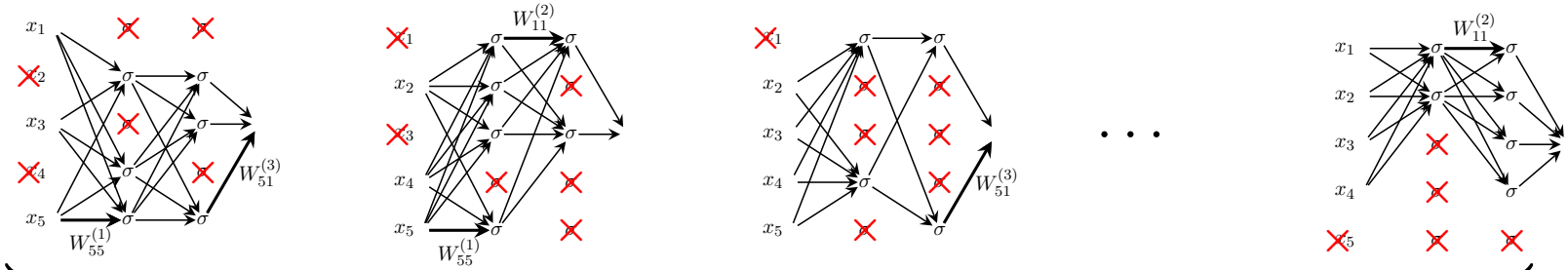


Seems ad hoc. *Why does this avoid overfitting?*

Dropout as an ensemble method

Dropout can bee seen as an ensemble method with two clever approximations.

- For a neural network with M units there are 2^M possible thinned neural networks. Consider this as our **ensamble**.



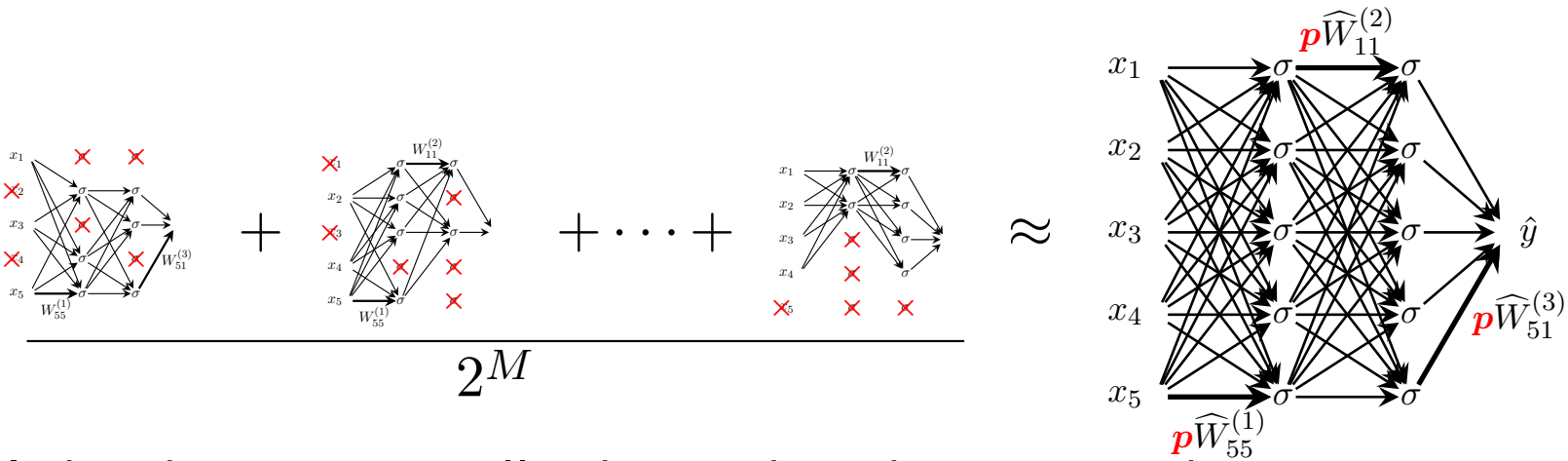
Our ensamble: 2^n"thinned" networks. They all share weights.

- **Approximation 1**: At each iteration we sample one ensemble and update it.
- Note, most of the networks will never be updated since $2^M \gg$ the number of iterations.

Dropout at test time

At test time we would need to average over all 2^M. Also this is not feasible since 2^M is huge.

- **Approximation 2**: Instead, at test time we evaluate the full neural network where the weight are multiplied by p.

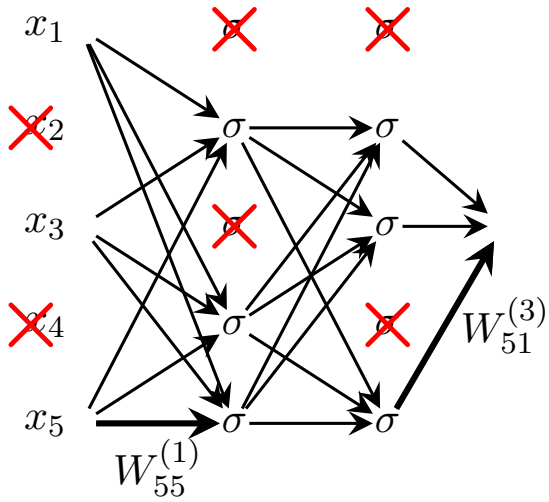


It has been empirically shown that this is a good approximation of the average of all ensemble members.

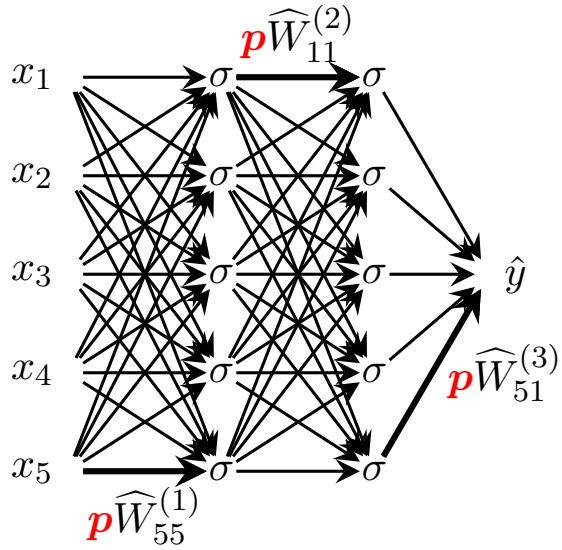
Dropout - One implementation detail

How we presented it (for a pedagogical reason):

During training



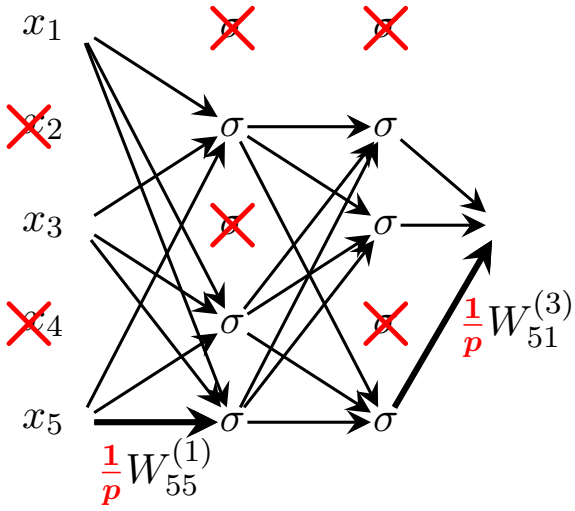
During testing



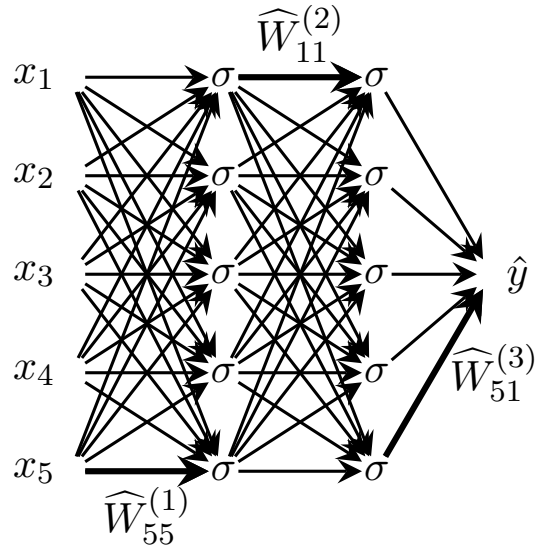
Dropout - One implementation detail

How it is usually implemented:

During training



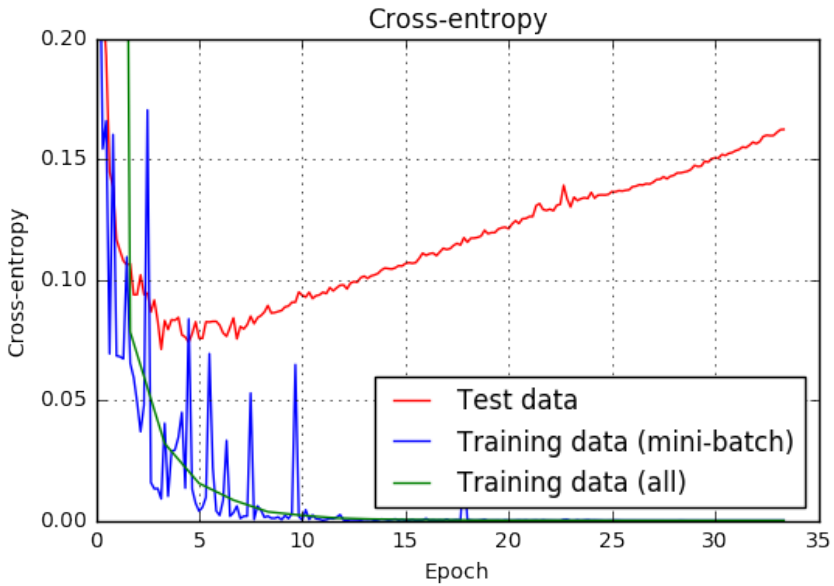
During testing

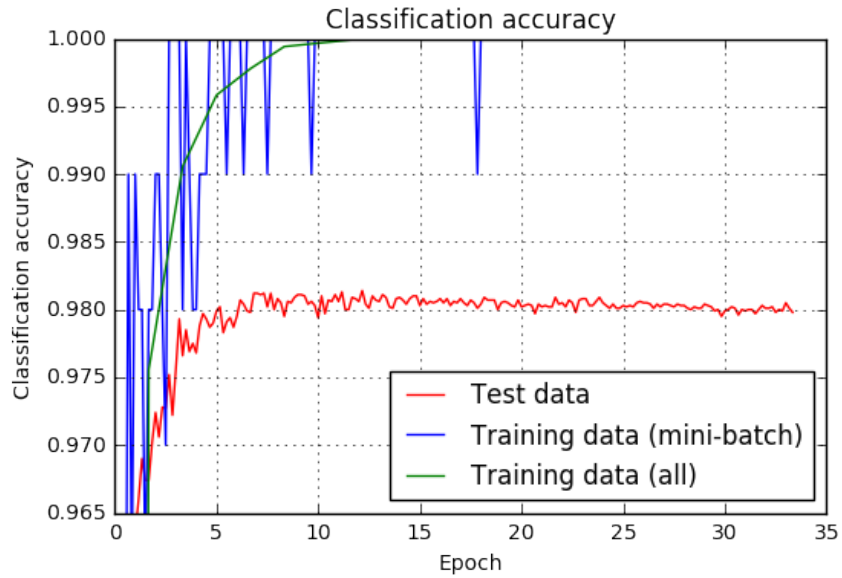


Only a difference of scale factor p during training and testing. The advantage is that we don't need to remember at test time which p we used for training.

Dropout on MNIST

Five layer NN with adaptive learning rate.
Trained for 20'000 iterations.

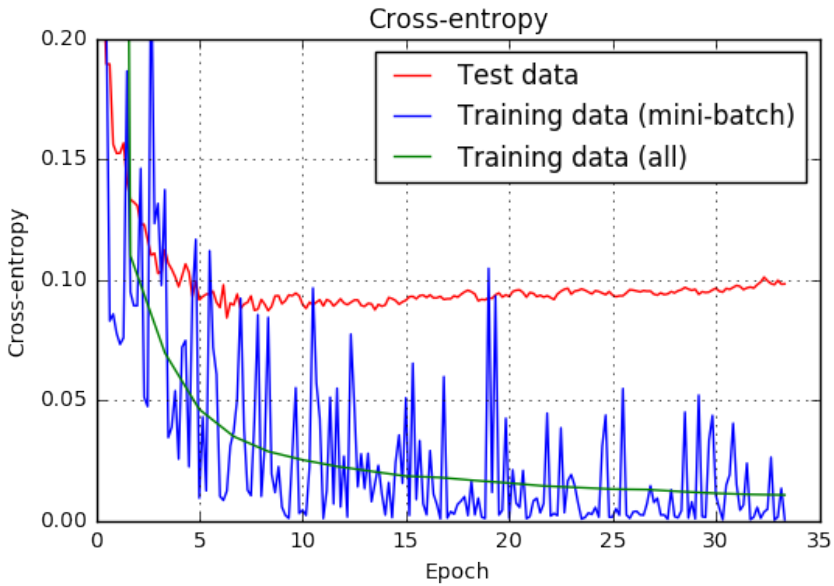


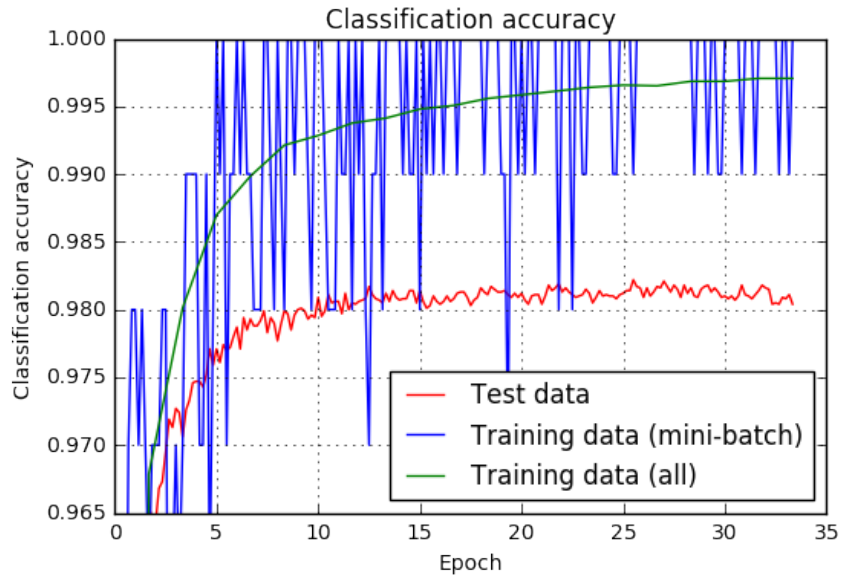


Train error: 0.0%. Test error: 2.0%

Dropout on MNIST

Five layer NN with adaptive learning rate and dropout.
Trained for 20'000 iterations.
We keep $p = 0.75$ the units in the first four layers.

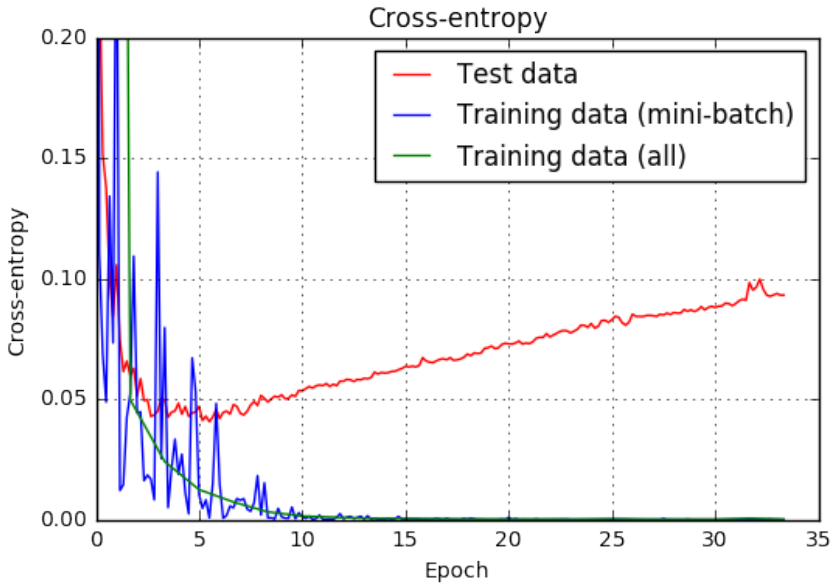


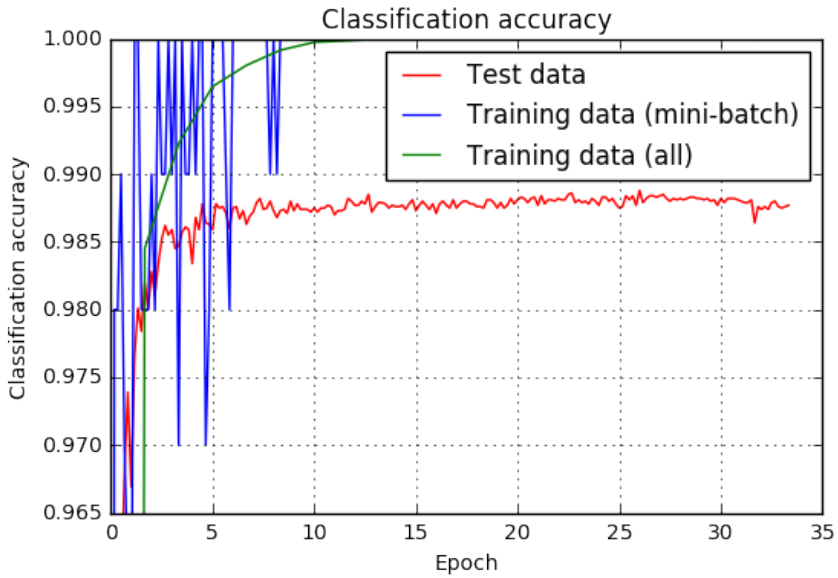


Train error: 0.3%. Test error: 1.9%
Reduced overfit according to cross entropy. No improvement in accuracy. Network structure is limiting - use CNNs!!

Dropout on MNIST

CNN - three conv layers, two fully connected layers
Kernel/units: 4-8-12-200,
Patches 5x5str1 5x5str2 4x4str2
Adaptive learning rate from $\gamma = 0.003$ to $\gamma = 0.0001$.





Train error: 0.0%. Test error: 1.2%
And now we start to overfit again ... Use dropout!

Dropout on MNIST

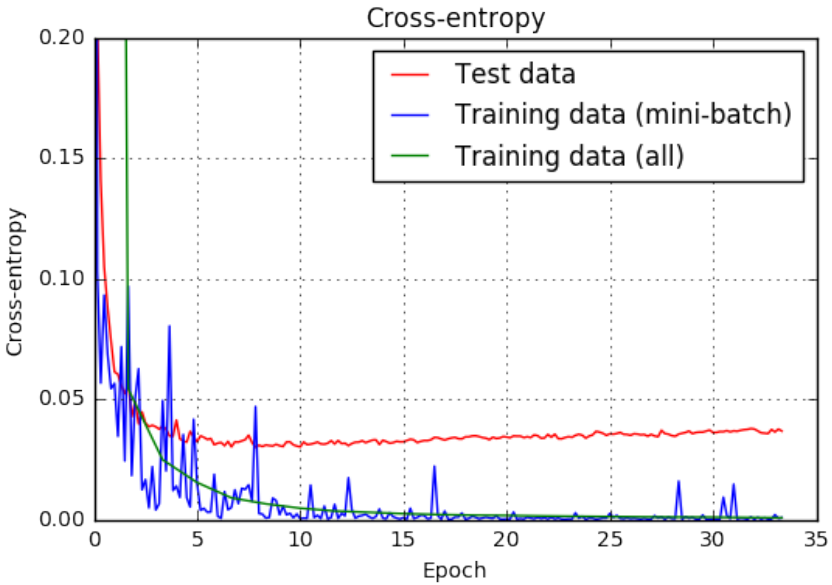
CNN - three conv layers, two fully connected layers

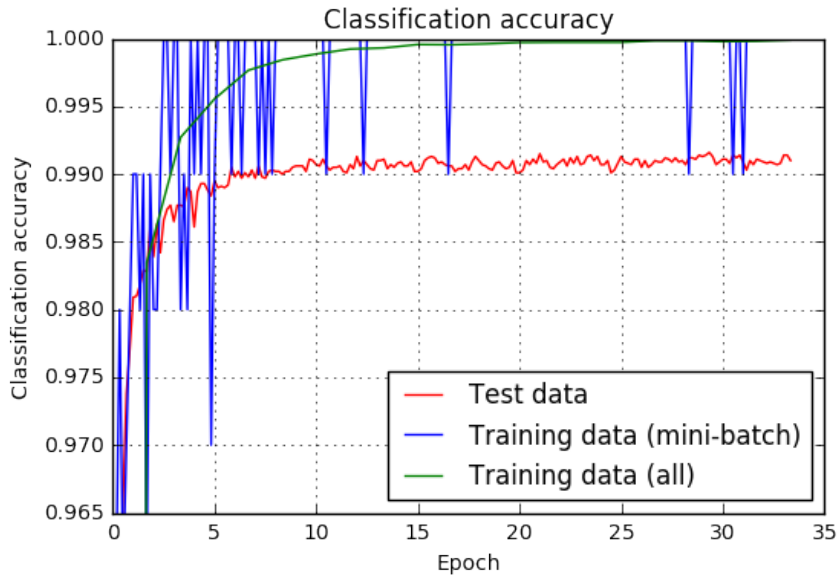
Kernel/units: 4-8-12-200,

Patches 5x5str1 5x5str2 4x4str2

Adaptive learning rate from: $\gamma = 0.003$ to $\gamma = 0.0001$

Dropout with $p = 0.75$ on units between the two fully connected laye



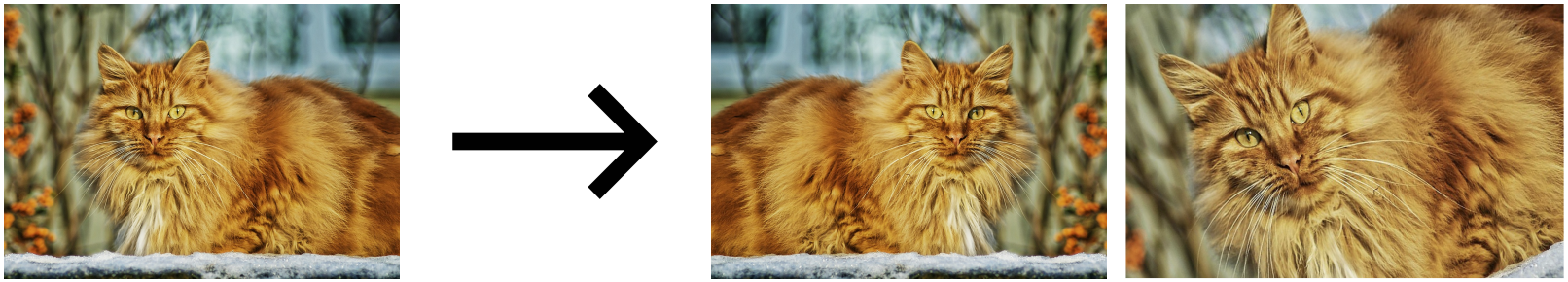


Train error: 0.01%. Test error: 0.9%

Better cross-entropy, and now also an improvement in accuracy!!

Data augmentation

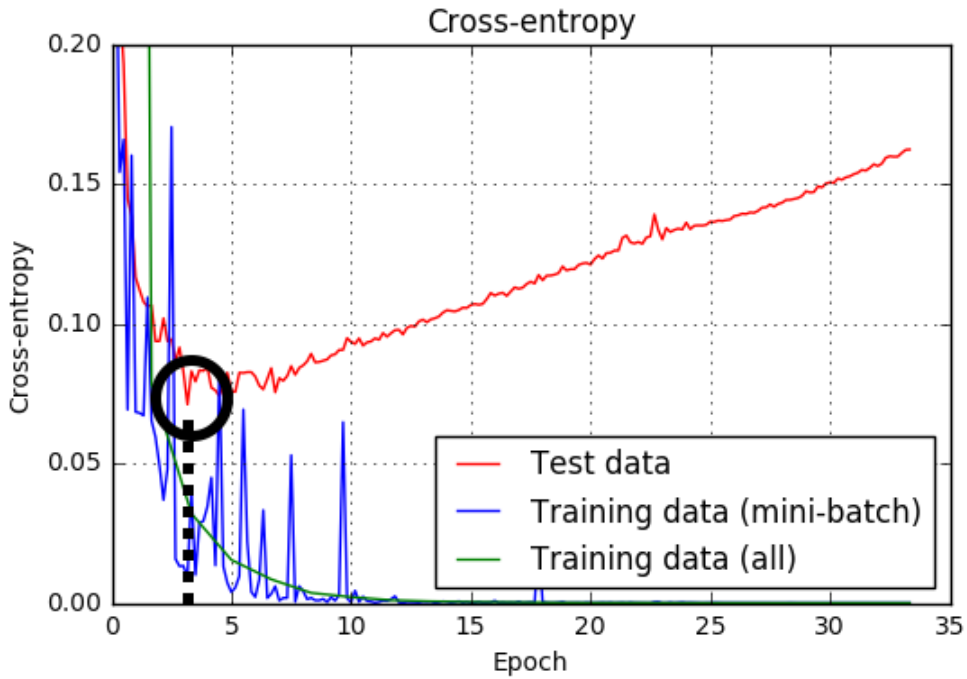
- Collecting more data reduces variance
- If we cannot collect more data, we can augment the data we have.



- Applicable transformations are problem dependent.
- Be careful not to change correct class!
- For example, vertical flip is not appropriate if you have images of ”b” and ”d”.

Early stopping

Large enough models $\Rightarrow$ test loss starts increasing



Idea: Stop training when test loss (or error) is at its minimum

- High risk that you overfit on test data (cherry-picking best model among many)
- Therefore you should have a separate datasets for picking model and for evaluating performance. See next lecture.

A few concepts to summarize lecture 6

Overfitting: Reduced generalization performance due to a too flexible model in comparison to the variety of examples in the training data.

Bias: The inability of a method to describe the true patterns in the classification or regression problem. Low model complexity.

Variance: Sensitivity to random effects (noise) in the training data. High model complexity.

Regularization: Different methods with the aim to reduce model variance in order to prevent overfitting.

Weight decay: Regularization method where the model parameters are penalized during training.

Ensemble methods: Umbrella term for methods that average or combine multiple models.

Bagging: Ensemble method based on the statistical bootstrap.

Dropout: Regularization method for neural networks based on randomly dropping units during training.

Data augmentation: Constructing new artificial data points in order to reduce variance.

Early stopping: Regularization method where we stop the training early based on