Regularization

Professor Marie Roch

San Diego State University

Regularization

Goal: Reduce generalization error

Strategies:

- Constraints on parameter values
  e.g., try to keep parameters from being too small/large
- Preferences for simpler models
- Guidance for underspecified problems
- Combine multiple hypotheses (ensemble methods)

Parameter norm penalties

Attempt to limit capacity $\tilde{J}(\theta|X,y) = \underbrace{J(\theta|X,y)}_{\text{loss/ objective function}} + \underbrace{\alpha\Omega(\theta)}_{\text{penalty depends on model } \theta}$

- $\alpha \in [0, \inf)$ is user settable
- Common to use a L^p norm for $\Omega(\Theta)$
- Model Θ comprised of weights w; $\tilde{J}(w|X,y)$ is equivalent

$$L^p, \text{ or } \|x\|_p = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$$ (Goodfellow et al. 2.5)

Parameter norm penalties

- Common to ignore bias
  - only shifts position
  - penalizing frequently results in underfitting
- Separate α per layer is possible, but…
  - complicates hyperparameter search
  - reasonable to use global α.

Parameter norm penalties

We will discuss two:

- L2 – Causes weights to get smaller
  - Shrinkage proportional to weight
  - AKA “weight decay”
- L1 – Makes weight vector “sparse”
  - Pulls weights towards zero by constant factors

$L^2 = \frac{1}{2}\|w^T w\|_2^2$ penalty

- Weight decay* penalty $\Omega(\theta) = \alpha_0 \frac{1}{2} w^T w$

$$\tilde{J}(\theta \mid X, y) = J(\theta \mid X, y) + \alpha\Omega(\theta) \quad \alpha = \frac{1}{2}\alpha_0$$

$$= J(\theta \mid X, y) + \alpha w^T w$$

- $\nabla_w \left(\alpha_0 \frac{1}{2} w^T w\right) = \alpha_0 \frac{2}{2} w$, so

$$\nabla_w \tilde{J}(\theta \mid X, y) = \nabla_w J(\theta \mid X, y) + \alpha_0 w$$

*Also known as ridge regression & Tikhonov regularization

L^2 penalty

- Consider weight update

$$w \leftarrow w - \epsilon \nabla_w \tilde{J}(\theta|X,y)$$

$$\begin{aligned} & w - \epsilon \nabla_w \tilde{J}(\theta|X,y) \\ &= w - \epsilon(\nabla_w (J(\theta|X,y)) + \alpha_0 w) \\ &= w - \epsilon\alpha_0 w - \epsilon\nabla_w J(\theta|X,y) \\ &= (1 - \epsilon\alpha_0)w - \epsilon\nabla_w J(\theta|X,y) \end{aligned}$$

$$w \leftarrow (1 - \epsilon\alpha_0)w - \epsilon\nabla_w J(\theta|X,y)$$

Tends to shrink weights (with appropriate α, ε)

L1 penalty

- L1 penalty $\Omega(\theta) = \sum_i |w_i|$

$$\tilde{J}(\theta \mid X, y) = J(\theta \mid X, y) + \alpha\Omega(\theta)$$
$$= J(\theta \mid X, y) + \alpha \sum_i |w_i|$$

- Gradient

$$\nabla_w \tilde{J}(\theta \mid X, y) = \nabla_w J(\theta \mid X, y) + \alpha \operatorname{sign}(w)$$

L1 penalty

- As weights are pulled towards zero, fewer will be active.
- Leads to more zeros, or a *sparse representation*
- Can be thought of as a type of feature selection and is used in one popular algorithm (LASSO).

Keras and L1/L2 penalties

Penalties are specified when constructing layers, example

```
from tensorflow.keras import regularizers
# other imports…

Dense(N, kernel_regularizer=regularizers.l2(α), …)
OR
Dense(N, kernel_regularizer=regularizers.l1(α), …)
```

Starting point for α? Perhaps 0.01

Dataset augmentation

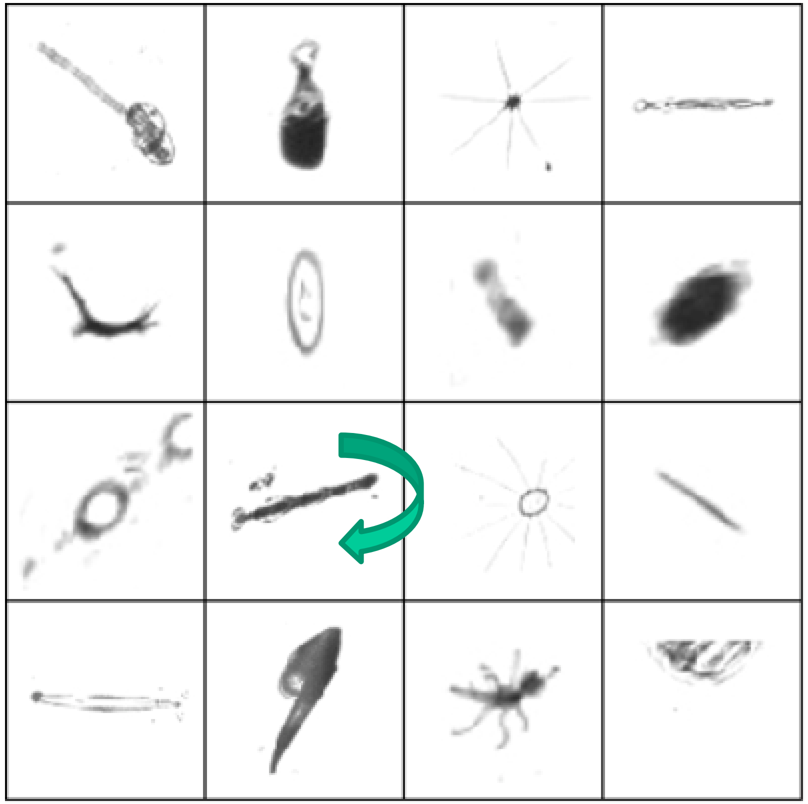
- We can improve classifiers by increasing training data.
- Data are expensive (most of the time)

- Solution:

fake data...

Dataset augmentation

- Primary application: classification tasks
- Basic idea
  - Transform inputs slightly
  - Frequently used in image processing
  - Transforms such as rotation, scale, shift



Plankton classification challenge http://www.datasciencebowl.com

SAN DIEGO STATE UNIVERSITY

Dataset augmentation for audio

- Can be a bit harder
- Some strategies
  - vocal tract length perturbation (Jaitly & Hinton 2013)
  - stretching, compressing
  - enhancements, e.g. adding noise (Prisyach et al. 2015) (small perturbations of inputs can be shown to be equivalent to L^p penalties on weights)

Noise robustness

- We have already seen input perturbation (dataset augmentation)
- We can add noise to other parts of the network
- One approach is to add noise to the weights, e.g. $N(0, \eta I)$

Weight perturbation interpretation

- Bayesian view: Weight values have a distribution and we are drawing from these
- With MSE cost functions and small variance (η), perturbation ≡ to adding penalty $||\nabla_w \hat{y}(x)||^2$
- Encourages optimization to find areas in parameter space where small weight changes have little influence on output (flat valleys in loss space)

Label Smoothing

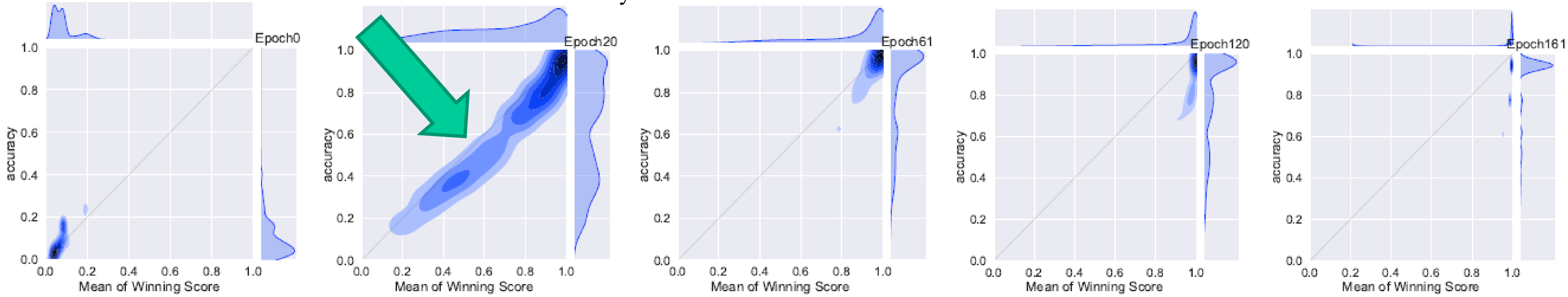
- Inject noise at output targets
- Maximizing $\log P(y|x)$ can lead to overfitting.
- Common to inject noise onto output target.
- Rob Peter to pay Paul… we use the standard softmax cost function with cross entropy & modified targets:
  - one hot target $1-\epsilon$
  - others $\dfrac{\epsilon}{N-1}$

San Diego State University

Overconfidence

- Networks have a tendency to overpredict after training

Desirable: best score related to accuracy



(Thulasidasan et al 2019)

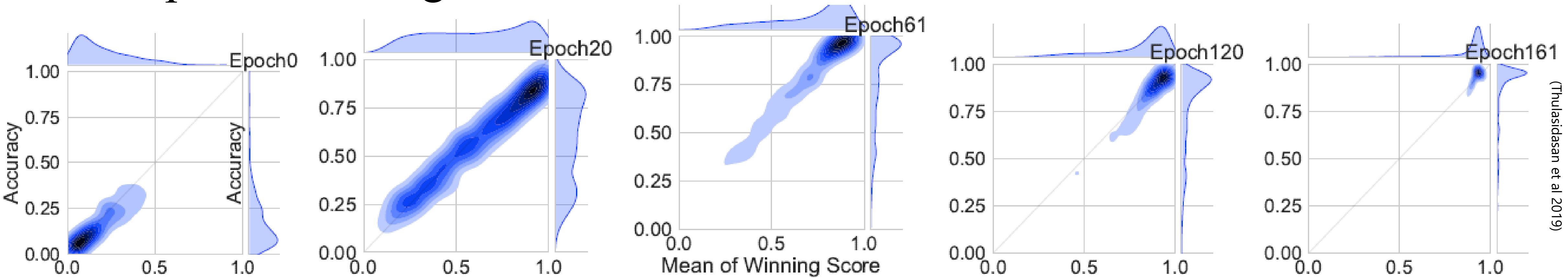
- Accuracy on CIFAR-100 (100 image clases) vs the mean of the highest prediction

Mixup

- Combine multiple examples together (Zhang et al. 2017)
  - One used as usual
  - One has been made weaker (e.g., attenuation for audio)
  - Examples are selected in the vicinity of the target example
- Labels are adjusted to account for the mixture

Mixup

- Results in
  - better calibration of predictions
  - performance gains



(Thulasidasan et al 2019)

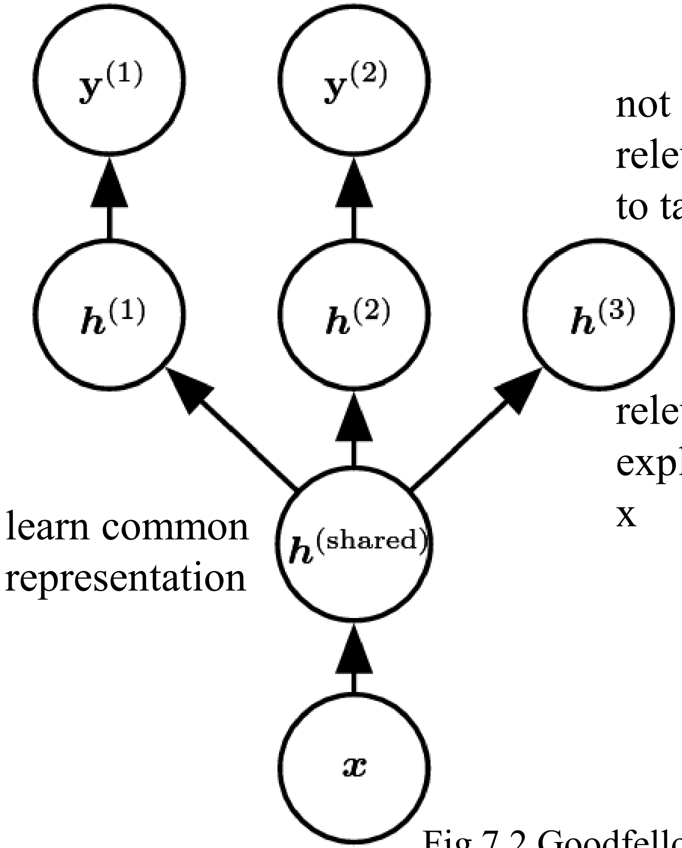
Thulasidasan, S., Chennupati, G., Bilmes, J., Bahattacharya, T., and Michalak, S. (2019). "On Mixup Training: Improved Calibration and Predictive Uncertainty for Deep Neural Networks," in *Neur Info Proc Sys (NeurIPS)* (Vancouver, Canada, Dec. 10-12), pp. 15.

Multitask learning

- Learn different functions from same data
- Pool portions of the network to learn common things
- Decreases generalization error

example

identity word sequence

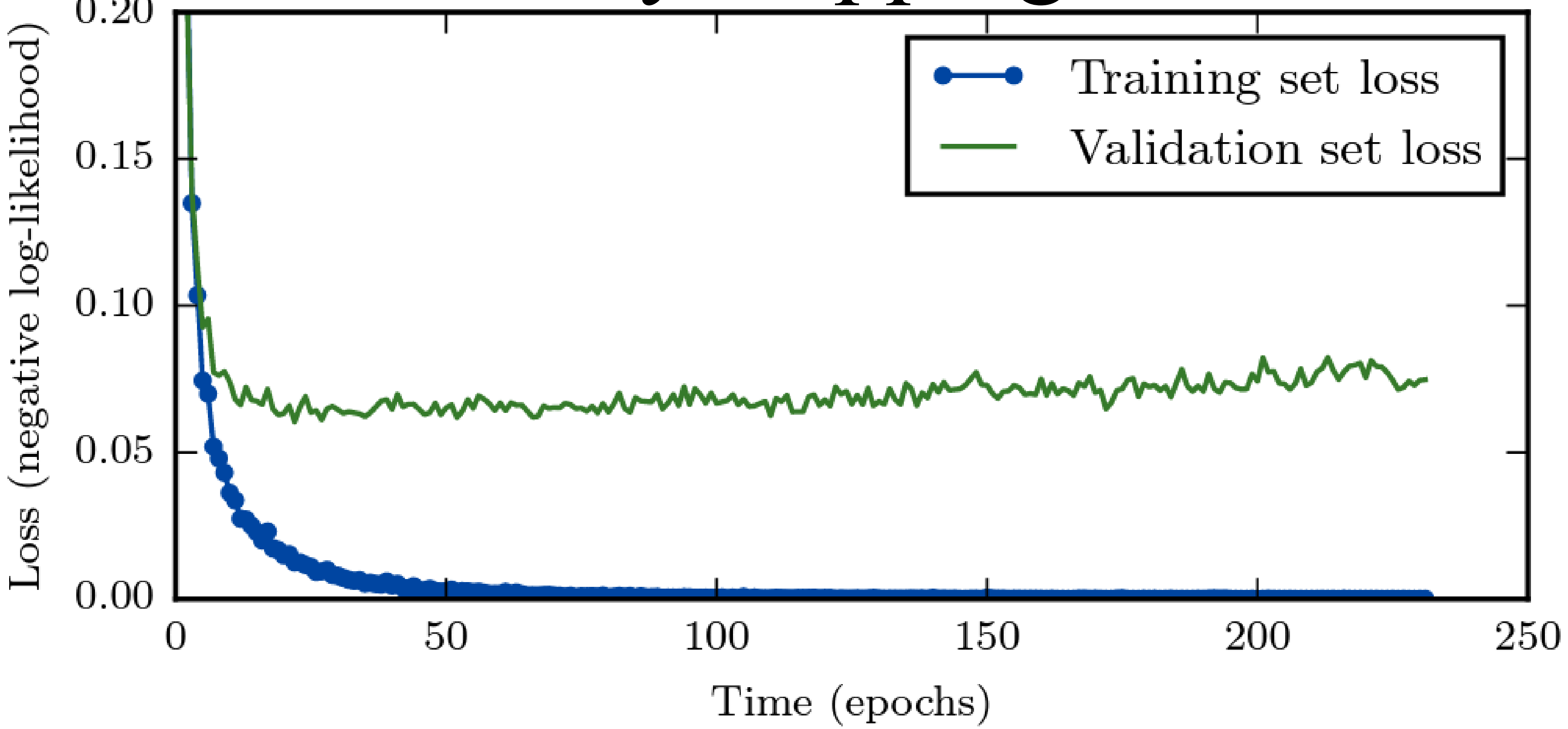


not relevant to tasks but relevant to explaining x

learn common representation

Fig 7.2 Goodfellow et al.

Early stopping



- Use other data to determine when to stop

Early stopping

```
Set loss to ∞
while we still have patience
        Update model Θ by n steps
        Check new loss on separate data
        If new loss < loss:
                store new loss, model, iteration
                restore patience
        else
                decrease patience
return best model
```

SAN DIEGO STATE UNIVERSITY

see Alg. 7.1 for details

Early stopping

- Variants exist
- One such variant:

  Create new model and retrain for the same number of steps up to stop with *all* data.

  Alg. 7.2

San Diego State University

Parameter tying

- Similar to multitask learning
- Learn two similar functions f_A and f_B with parameters Θ_A and Θ_B.
- Assume it makes sense for $w_A \in \theta_A$ and $w_B \in \theta_B$ to have similar weights

key idea: similar features may be used

Parameter tying

- Force weights to be similar with penalty, e.g. L2 penalty:

$$\Omega\left(w^{(A)}, w^{(B)}\right) = \left\|w^{(A)} - w^{(B)}\right\|_2^2$$

Parameter sharing

- Similar idea, weights in nodes are learning similar things. e.g. a feature in two networks A and B.
- Differs in that the *same* set of weights are used in both networks.

Sparse representation

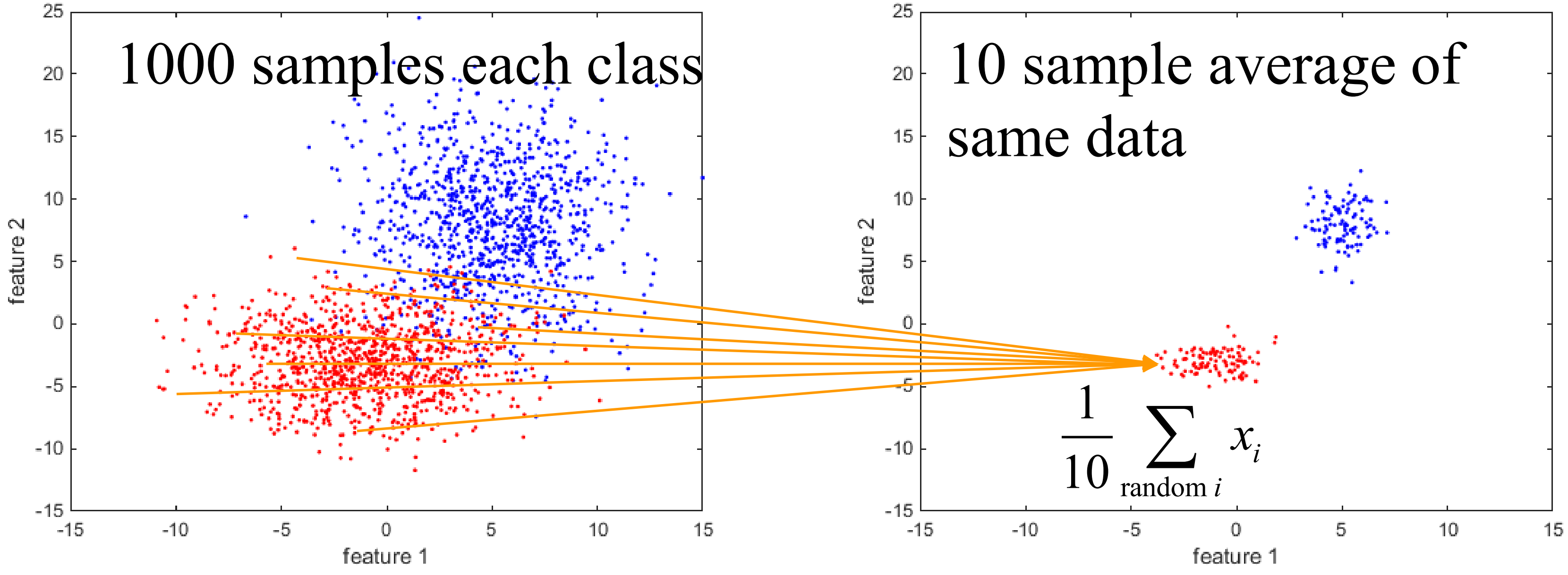
- L1 penalties cause weights to tend towards zero, giving a *sparse parameterization*.
- *Sparse representations* occur when many of the *parameters* tend towards zero.
- Can be accomplished with a L1 penalty on layer outputs: $\alpha\Omega(h) = \alpha\frac{1}{m}\sum_i h^{(i)}$, other methods exist

Ensemble methods

- We know that models make mistakes.
- We hope that the error is randomly distributed…

- If so, averaging model outputs should reduce the noise.

Why this works
Example averaging features
(as opposed to model outputs)



SAN DIEGO STATE UNIVERSITY

Bootstrap aggregation
Bagging, or model aggregation

- Ensemble method
- Train multiple networks
  - Usually with different data
  - Due to randomness in neural nets, can be done with same data
- Classify and vote/average output

Bagging expected variance

- Suppose k models $m_1, m_2, \ldots, m_k$
  - Assume errors $\epsilon_i \sim n(0, v)$
  - It follows that $\mathrm{E}[(\epsilon_i - 0)^2] = E\left[\epsilon_i^2\right] = v$
- Suppose covariances of errors between models are c

$$E\left[(\epsilon_i - 0)(\epsilon_j - 0)\right] = E[\epsilon_i \epsilon_j] = c$$

- Consider the variance of the mean error across models:

$$\mathrm{E}\left[\left(\frac{1}{k}\sum_i (\epsilon_i - 0)^2\right)\right]$$

Expected bagging variance

$$E\left[\sum_i \left(\frac{1}{k}(\epsilon_i - 0)\right)^2\right]$$

$$= \frac{1}{k^2} E\left[\left(\sum_i \epsilon_i\right)^2\right]$$

$$= \frac{1}{k^2} E[(\epsilon_1 + \epsilon_2 + \epsilon_2 + \cdots + \epsilon_k)^2]$$

$$= \frac{1}{k^2} E\big[({\epsilon_1}^2 + \epsilon_1\epsilon_2 + \cdots + \epsilon_1\epsilon_k) + (\epsilon_2\epsilon_1 + \epsilon_2^2 + \epsilon_2\epsilon_3 + \cdots + \epsilon_2\epsilon_k)$$
$$+ \cdots + \left(\cdots + \epsilon_{k-1}\epsilon_k + \epsilon_k^2\right)\big]$$

$$= \frac{1}{k^2} E\left[\sum_i \left(\epsilon_i^2 + \sum_{i \neq j} \epsilon_i\epsilon_j\right)\right]$$

Expected bagging variance

$$= \frac{1}{k^2} E\left[\sum_i \left(\epsilon_i^2 + \sum_{i \neq j} \epsilon_i \epsilon_j\right)\right]$$

$$= \frac{1}{k^2}\left(E\left[\sum_i \epsilon_i^2\right] + E\left[\sum_i \sum_{i \neq j} \epsilon_i \, \epsilon_j\right]\right)$$

$$= \frac{1}{k^2}\left(\sum_i E\left[\epsilon_i^2\right] + \sum_i \sum_{i \neq j} E\left[\epsilon_i \epsilon_j\right]\right)$$

$$= \frac{1}{k^2}\left(\sum_i v + \sum_i \sum_{i \neq j} c\right) \quad \text{as } E[\epsilon_i] = v, E[\epsilon_i \epsilon_j] = c$$

$$= \frac{1}{k^2}(kv + k(k-1)c) = \frac{1}{k} v + \frac{k-1}{k} c$$

Expected bagging variance

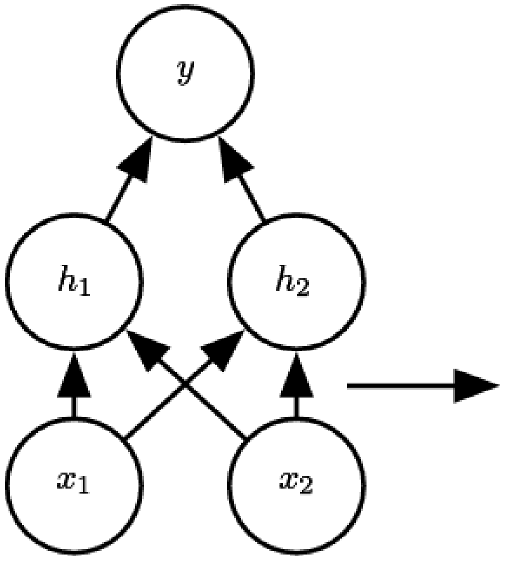
$$E\left[\left(\frac{1}{k}\sum_i(\epsilon_i - 0)^2\right)\right] = \frac{1}{k}v + \frac{k-1}{k}c = \frac{v + (k-1)c}{k}$$

- What if errors are perfectly correlated?
  - Then we would expect $E[\epsilon_i \epsilon_j] = v$ and would be back where we started with a mean variance of v.
- If errors are uncorrelated, $E[\epsilon_i \epsilon_j] = 0$, we have shrunk variance by a factor of k
- As we vary the data or models, we expect somewhere between the extremes, reducing the variance.

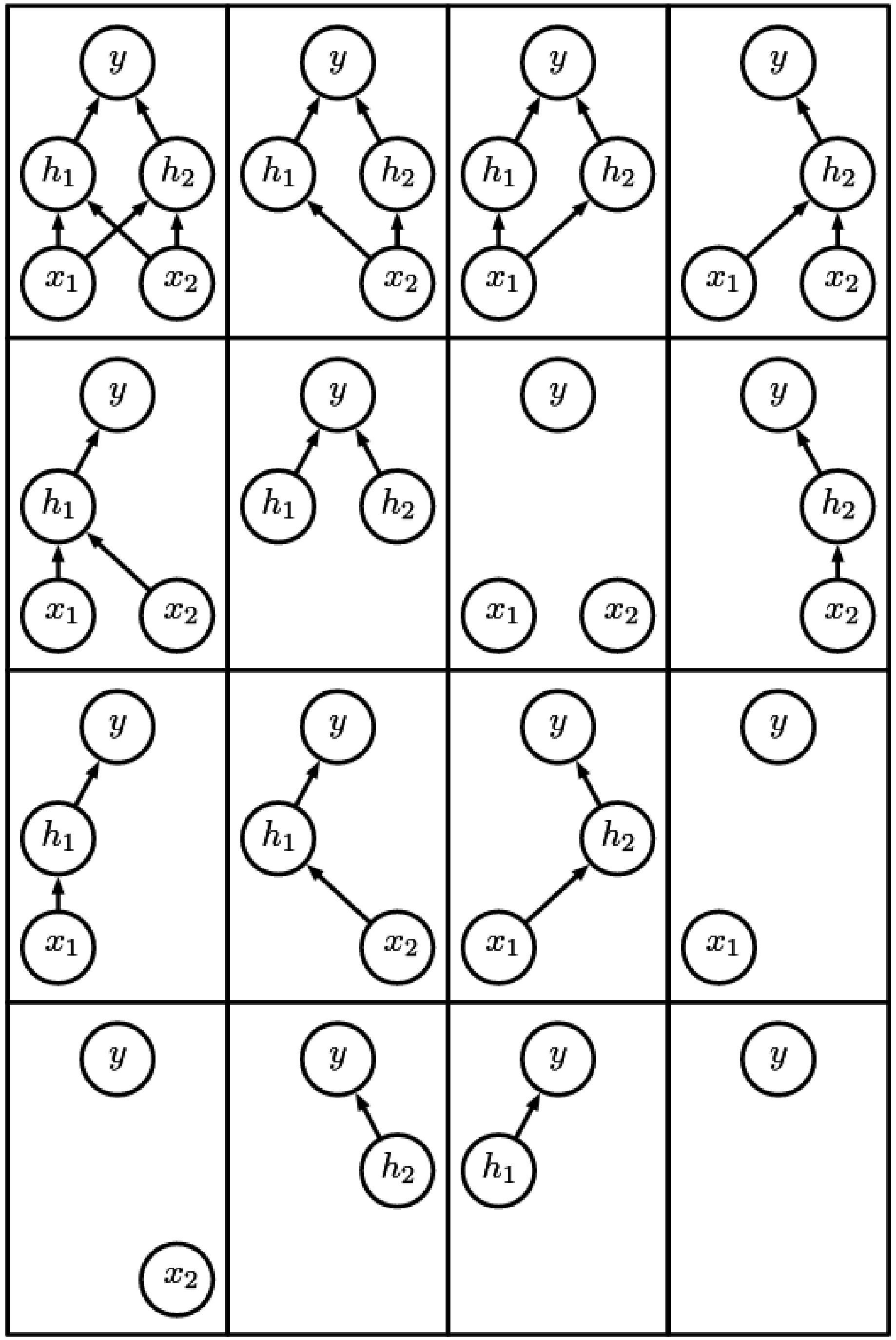
Dropout

- Related to bootstrap aggregation
- Builds implicitly different models as opposed to explicit ones
- Reduces neurons dependence on one another

Dropout



Base network



Ensemble of subnetworks

Goodfellow et al. Fig. 7.6

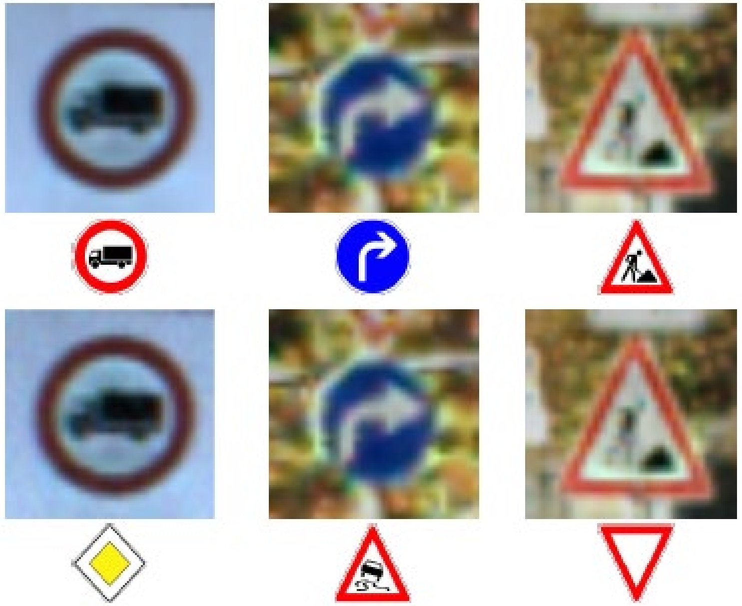
Each training example uses a sampled subnet

Dropout

- Each sampled model shares weights with all other models
- Prediction weights each unit’s output by probability of it being dropped
  (know as the weight scaling inference rule)
- Tends to work well with maxnorm constraints

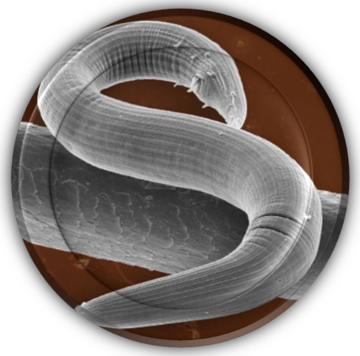
Adversarial training

Sometimes, moving away from an example in feature space can cause radical changes in labeling



Papernot et al. 2017 ACM Conf Comp Sec

SAN DIEGO STATE UNIVERSITY



cdn.scinews.com

Uh-oh…

© WWF / Martin Harvey

Moves away from the cost gradient can accomplish this:

 $+\, .007 \times$  $=$ 

Goodfellow et al. Fig. 7.8

$\boldsymbol{x}$	$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$	$\boldsymbol{x} + \epsilon\, \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
$y =$"panda" w/ 57.7% confidence	"nematode" w/ 8.2% confidence	"gibbon" w/ 99.3% confidence

Adversarial training

- Specific type of dataset augmentation
- Find examples close to the data that have different predicted labels
- Train on them with the correct label
  - If label is not verified, it is called a virtual adversarial example