Regularization

Norm regularization (L_2, L_1)

Generalization

The central challenge of machine learning is to perform well on the - *unseen test* data, not just the *training data*

While training the model

Train err $\frac{1}{m^{(\text{train})}}||\boldsymbol{X}^{(\text{train})}\boldsymbol{w} - \boldsymbol{y}^{(\text{train})}||_2^2,$

What we actually want

Test err (or)
Generalization err $\frac{1}{m^{(\text{test})}}||\boldsymbol{X}^{(\text{test})}\boldsymbol{w} - \boldsymbol{y}^{(\text{test})}||_2^2.$

How can we say something about the test data by only seeing the train data?

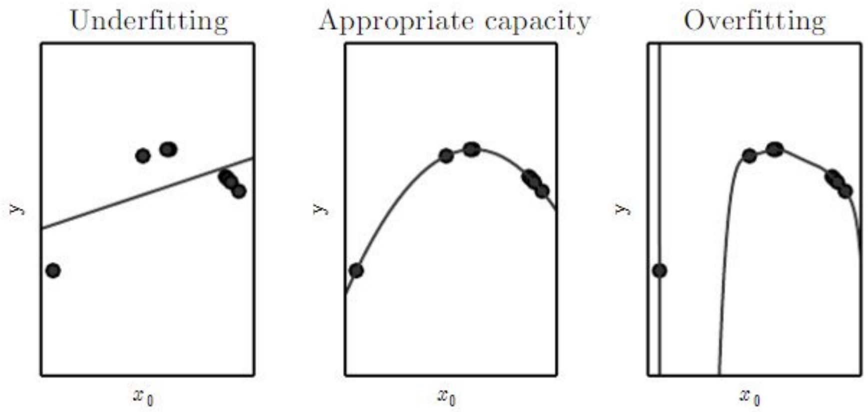
Statistical learning theory

- Training and Test sets are not arbitrary
- Underlying *data generating distribution* is same

*Slide courtesy, Ian Goodfellow et al., deep learning book

Capacity, Overfitting and Underfitting

The central challenge of machine learning is to perform well on the *unseen test* data, not just the *training data*



$$\hat{y} = b + wx.$$

$$\hat{y} = b + w_1 x + w_2 x^2.$$

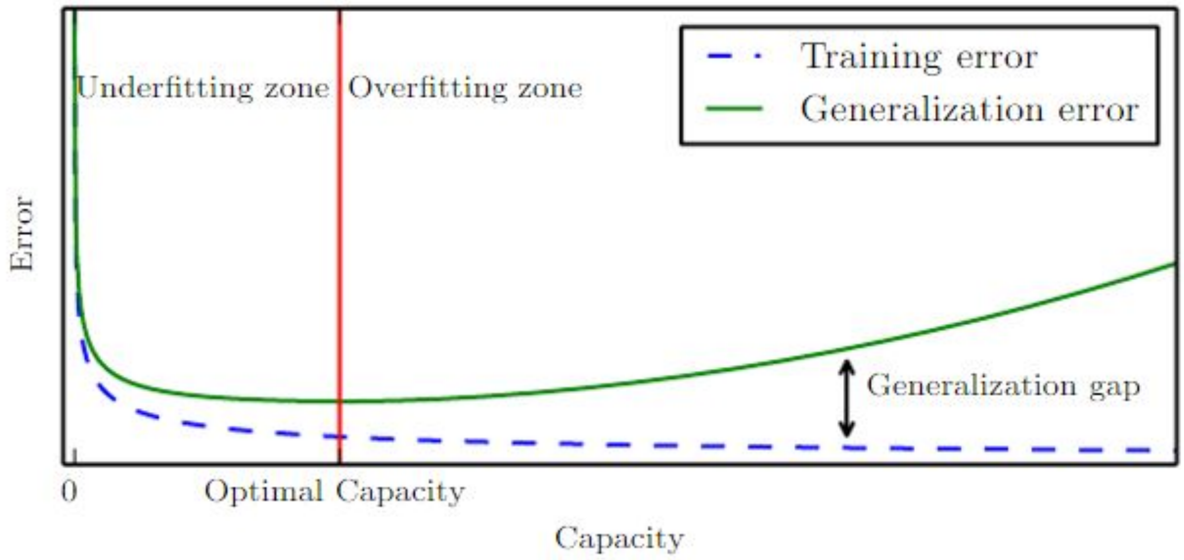
$$\hat{y} = b + \sum_{i=1}^{9} w_i x^i.$$

Occam's razor: This principle states that among competing hypotheses that explain known observations equally well, one should choose the "simplest" one.

*Slide courtesy, Ian Goodfellow et al., deep learning

Capacity, Overfitting and Underfitting

The central challenge of machine learning is to perform well on the *unseen test* data, not just the *training data*



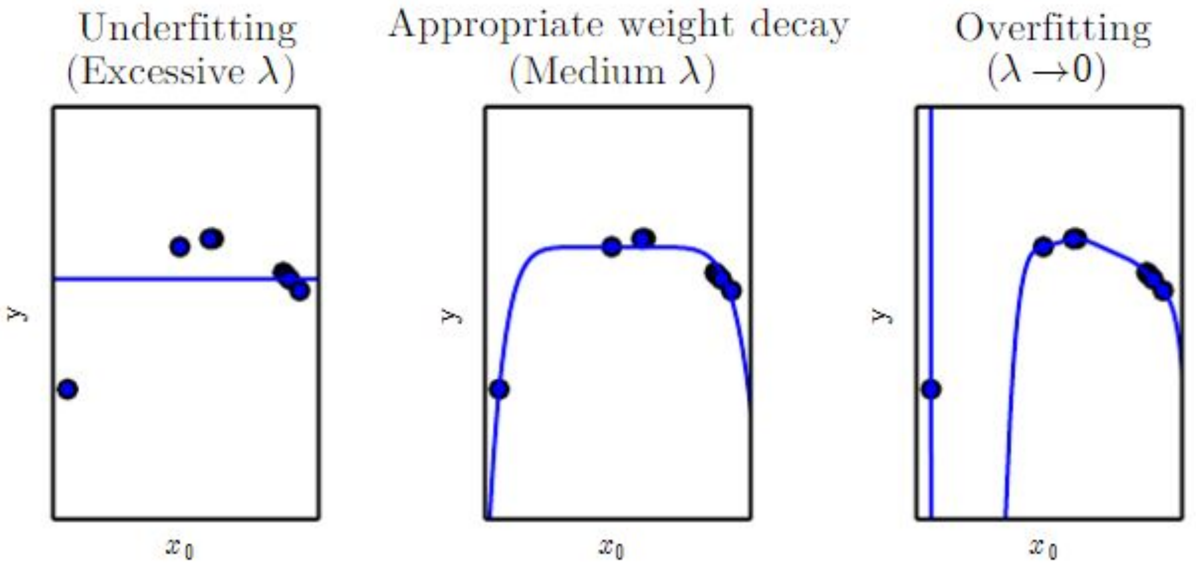
*Slide courtesy, Ian Goodfellow et al., deep learning book

Regularization

Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error

$$J(\boldsymbol{w}) = \text{MSE}_{\text{train}} + \lambda \boldsymbol{w}^\top \boldsymbol{w},$$

Note that *biases* in general are not penalized



*Slide courtesy, Ian Goodfellow et al., deep learning book

Regularization - parameter norm penalties

L_2 norm regularization
(weight decay, ridge regression)

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha \Omega(\boldsymbol{\theta})$$

$$\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \frac{\alpha}{2} \boldsymbol{w}^\top \boldsymbol{w} + J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}),$$

Parameter update:

$$\nabla_{\boldsymbol{w}} \tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \alpha \boldsymbol{w} + \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}).$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \epsilon \left(\alpha \boldsymbol{w} + \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) \right).$$

$$\boldsymbol{w} \leftarrow (1 - \epsilon \alpha) \boldsymbol{w} - \epsilon \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}).$$

*Slide courtesy, Ian Goodfellow et al., deep learning book

Regularization - parameter norm penalties

L_2 norm regularization

$$\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \frac{\alpha}{2}\boldsymbol{w}^\top \boldsymbol{w} + J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}),$$

Consider:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} J(\boldsymbol{w}).$$ *Unregularized solution*

- *Taylor expansion of* J *at* w^*,

$$\hat{J}(\boldsymbol{\theta}) = J(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^\top \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*),$$

$\boldsymbol{H}$ is hessian of J wrt w at w^*; ***linear term?***

$$\nabla_{\boldsymbol{w}} \hat{J}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*)$$

- *Add the L_2 regularization grad*

$$\alpha \tilde{\boldsymbol{w}} + \boldsymbol{H}(\tilde{\boldsymbol{w}} - \boldsymbol{w}^*) = 0$$

$$(\boldsymbol{H} + \alpha \boldsymbol{I})\tilde{\boldsymbol{w}} = \boldsymbol{H}\boldsymbol{w}^*$$

$$\tilde{\boldsymbol{w}} = (\boldsymbol{H} + \alpha \boldsymbol{I})^{-1}\boldsymbol{H}\boldsymbol{w}^*.$$

- *Since* $\boldsymbol{H}$ *is real and symmetric*

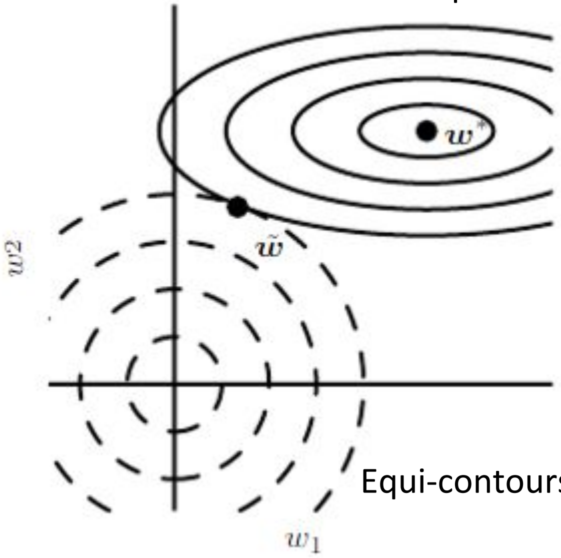
$$\begin{aligned}
\tilde{\boldsymbol{w}} &= (\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top + \alpha \boldsymbol{I})^{-1}\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top \boldsymbol{w}^* \\
&= \left[\boldsymbol{Q}(\boldsymbol{\Lambda} + \alpha \boldsymbol{I})\boldsymbol{Q}^\top\right]^{-1}\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top \boldsymbol{w}^* \\
&= \boldsymbol{Q}\underbrace{(\boldsymbol{\Lambda} + \alpha \boldsymbol{I})^{-1}\boldsymbol{\Lambda}}_{\lambda_i / (\lambda_i + \alpha)}\boldsymbol{Q}^\top \boldsymbol{w}^*.
\end{aligned}$$

*Slide courtesy, Ian Goodfellow et al., deep learning book

Regularization - parameter norm penalties

L_2 norm regularization

$$\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \frac{\alpha}{2}\boldsymbol{w}^\top \boldsymbol{w} + J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}),$$



Equi-contours of unregularized objective $\boldsymbol{J}$

Equi-contours of L_2 regularizer

- *Since $\boldsymbol{H}$ is real and symmetric*

$$\begin{aligned}\tilde{\boldsymbol{w}} &= (\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top + \alpha \boldsymbol{I})^{-1}\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top \boldsymbol{w}^* \\ &= \left[\boldsymbol{Q}(\boldsymbol{\Lambda} + \alpha \boldsymbol{I})\boldsymbol{Q}^\top\right]^{-1}\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top \boldsymbol{w}^* \\ &= \boldsymbol{Q}\underbrace{(\boldsymbol{\Lambda} + \alpha \boldsymbol{I})^{-1}\boldsymbol{\Lambda}}_{\lambda_i / (\lambda_i + \alpha)}\boldsymbol{Q}^\top \boldsymbol{w}^*.\end{aligned}$$

Regularizer effectively rescales ***w**** along eigenvectors of $\boldsymbol{H}$

- $\boldsymbol{\lambda}_i >> \boldsymbol{\alpha}$, regularizer effect is relatively less
- $\boldsymbol{\lambda}_i << \boldsymbol{\alpha}$, weights shrink to zero

*Slide courtesy, Ian Goodfellow et al., deep learning book

Regularization - parameter norm penalties

L_1 norm regularization

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha\Omega(\boldsymbol{\theta})$$

$$\Omega(\boldsymbol{\theta}) = ||\boldsymbol{w}||_1 = \sum_i |w_i|,$$

$$\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \alpha||\boldsymbol{w}||_1 + J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}),$$

$$\nabla_{\boldsymbol{w}}\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \alpha \text{sign}(\boldsymbol{w}) + \nabla_{\boldsymbol{w}} J(\boldsymbol{X}, \boldsymbol{y}; \boldsymbol{w})$$

*Slide courtesy, Ian Goodfellow et al., deep learning book

Regularization - parameter norm penalties

L_1 norm regularization

- *Taylor expansion of J at w^*,*

$$\hat{J}(\boldsymbol{\theta}) = J(\boldsymbol{w}^*) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}^*)^\top \boldsymbol{H}(\boldsymbol{w} - \boldsymbol{w}^*),$$

$\boldsymbol{H}$ is diagonal, with $h_{ii} > 0$, for all i

- Revisiting L_2 with diagonal $\boldsymbol{H}$

$$\tilde{w}_i = \frac{H_{i,i}}{H_{i,i}+\alpha} w_i^*$$

Still it only scales the weights - not sparsity

- *Now, the L_1 regularized objective*

$$\hat{J}(\boldsymbol{\theta}) = J(\boldsymbol{w}^*; \boldsymbol{X}, \boldsymbol{y}) + \sum_i \left[\frac{1}{2} H_{i,i}(\boldsymbol{w}_i - \boldsymbol{w}_i^*)^2 + \alpha |w_i|\right]$$

$$w_i = \operatorname{sign}(w_i^*) \max\left\{|w_i^*| - \frac{\alpha}{H_{i,i}}, 0\right\}$$. solution

- *Let* $w_i^* > 0$, for all i

  - if $w_i^* \leq \frac{\alpha}{H_{ii}}$; $w_i = 0$
  - esle $w_i^* \geq \frac{\alpha}{H_{ii}}$

  w_i moves towards *0* by $\frac{\alpha}{H_{ii}}$

L_1 regularization enforces **sparsity** in the solution

*Slide courtesy, Ian Goodfellow et al., deep learning book

Regularization - other methods

Bagging - bootstrap aggregating

- *Say k regression models*
  - Say each of them makes ϵ_i error
  - Error is drawn from Multivariate normal distribution

$$\mathbb{E}[\epsilon_i^2] = v; \quad \mathbb{E}[\epsilon_i \epsilon_j] = c$$

- *Avg. error by k models*

$$(1/k)\sum_i \epsilon_i$$

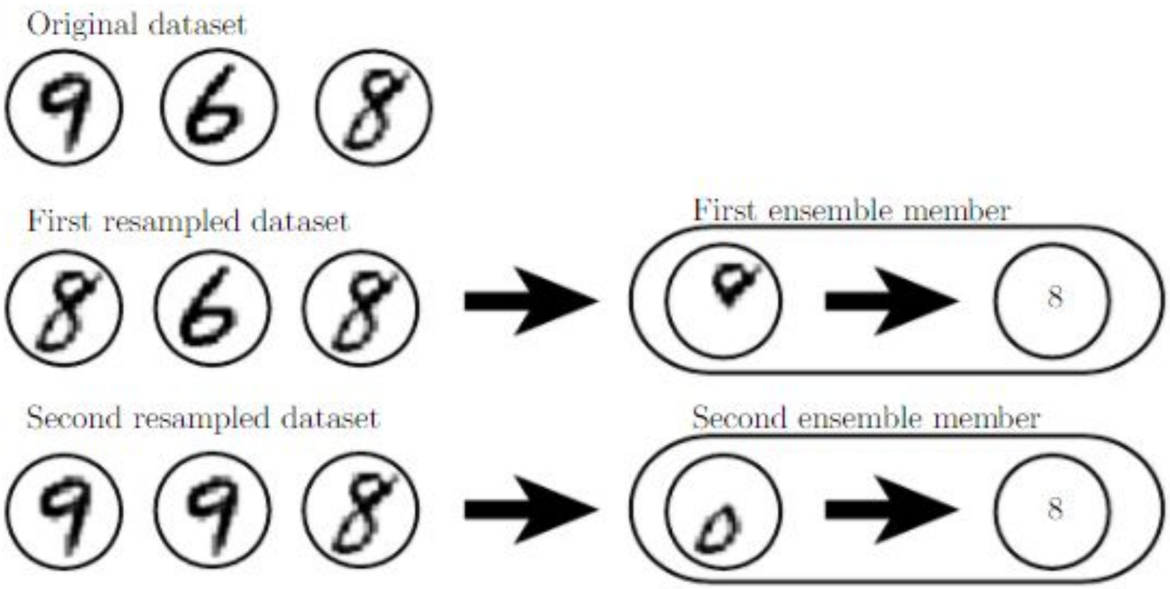
- *Expected squared error of the ensemble predictor*

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

  - If the models are perfectly correlated and $c = v$, error reduces to v
  - If perfectly uncorrelated, $c = 0$, *error reduces to v/k*

Regularization - other methods

Bagging - bootstrap aggregating (Breiman, 1994)

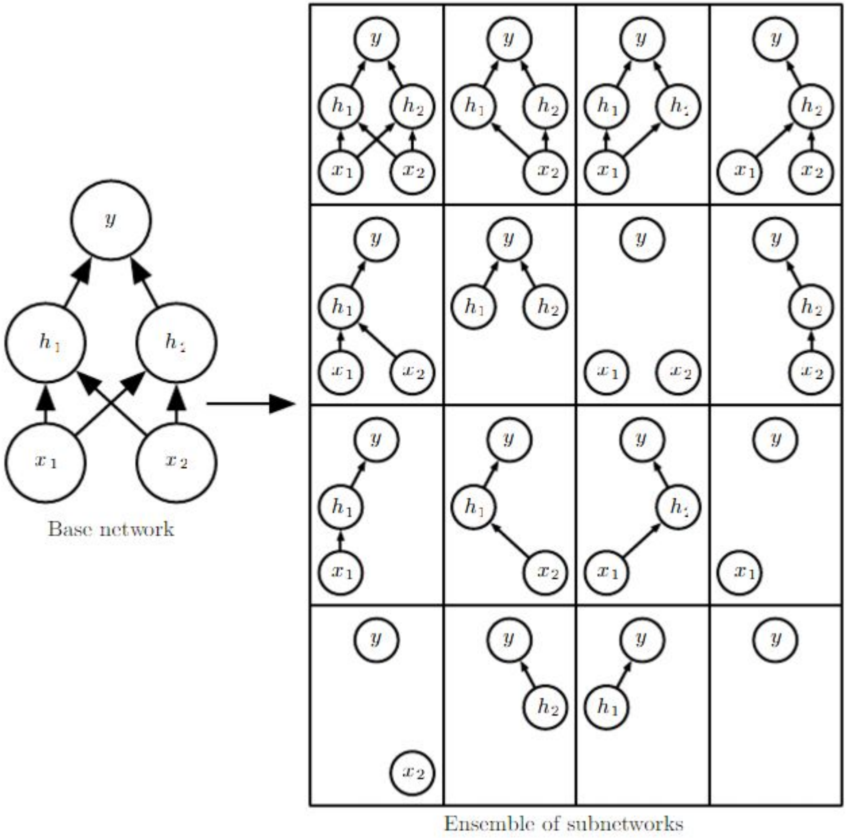


Regularization - other methods

Drop-out (Srivastava et al., 2014)

Stochastically turn the activation of the hidden unit off with a probability, $\boldsymbol{p}$

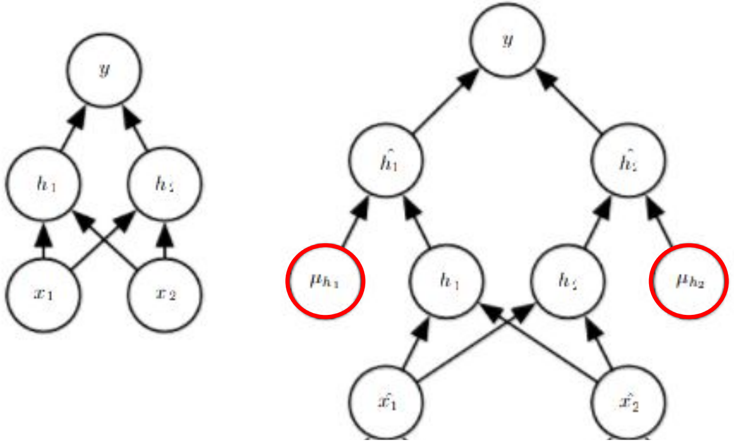
$$h^{(k)} = f(Wh^{(k-1)} + b^{(k-1)})$$
$$\hat{h}^{(k)} = \mu^{(k)} \odot h^{(k)}$$



- How to train it?
- Is this same as bagging?

*Slide courtesy, Ian Goodfellow et al., deep learning book

Regularization - other methods



Drop-out (Srivastava et al., 2014)

Stochastically turn the activation of the hidden unit off with a probability, $\boldsymbol{p}$

$$h^{(k)} = f(Wh^{(k-1)} + b^{(k-1)})$$
$$\hat{h}^{(k)} = \mu^{(k)} \odot h^{(k)}$$

– Inference, $p(y|x)$

Bagging

$$\frac{1}{k}\sum_{i=1}^{k} p^{(i)}(y \mid \boldsymbol{x}).$$

Drop-out

$$\sum_{\boldsymbol{\mu}} p(\boldsymbol{\mu}) p(y \mid \boldsymbol{x}, \boldsymbol{\mu})$$

$p(\boldsymbol{\mu})$ - Distribution used to sample $\boldsymbol{\mu}$

- Not easy to evaluate, why?
- Do sample averaging

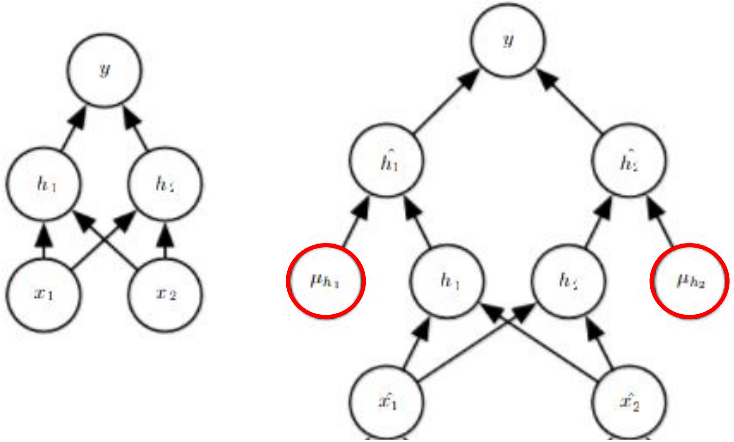
*Slide courtesy, Ian Goodfellow et al., deep learning book

Regularization - other methods

Drop-out (Srivastava et al., 2014)

Stochastically turn the activation of the hidden unit off with a probability, $\boldsymbol{p}$

$$h^{(k)} = f(Wh^{(k-1)} + b^{(k-1)})$$
$$\hat{h}^{(k)} = \mu^{(k)} \odot h^{(k)}$$



Drop-out

$$\sum_{\boldsymbol{\mu}} p(\boldsymbol{\mu}) p(y \mid \boldsymbol{x}, \boldsymbol{\mu})$$

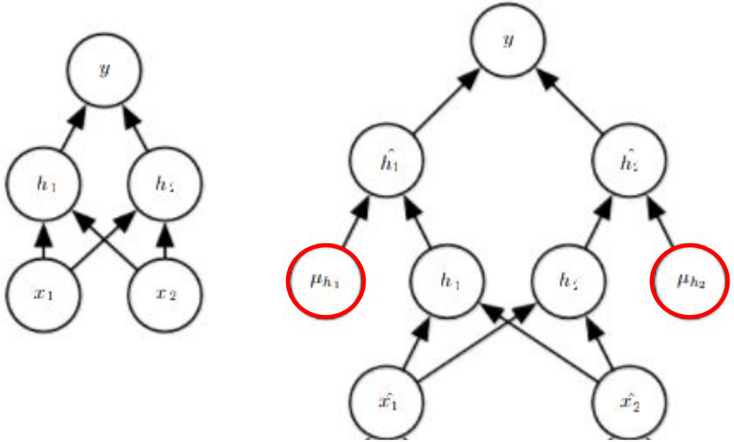
$p(\boldsymbol{\mu})$ - Distribution used to sample $\boldsymbol{\mu}$

- Not easy to evaluate, why?
- Do sample averaging

We will look at a simple weight scaling result which ***approximates*** the *geometric mean* of models prediction in one forward pass

*Slide courtesy, Ian Goodfellow et al., deep learning book

Regularization - other methods



Drop-out (Srivastava et al., 2014)

Stochastically turn the activation of the hidden unit off with a probability, $\boldsymbol{p}$

$$h^{(k)} = f(Wh^{(k-1)} + b^{(k-1)})$$
$$\hat{h}^{(k)} = \mu^{(k)} \odot h^{(k)}$$

Weight rescaling (Hinton et al., 2012)

To evaluate $p(y|x)$ with all units

- Multiply weights going out of unit i with probability of including unit i

Drop-out

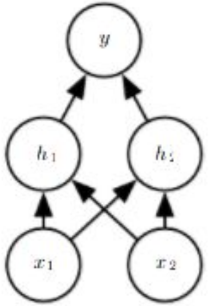
$$\sum_{\boldsymbol{\mu}} p(\boldsymbol{\mu}) p(y \mid \boldsymbol{x}, \boldsymbol{\mu})$$

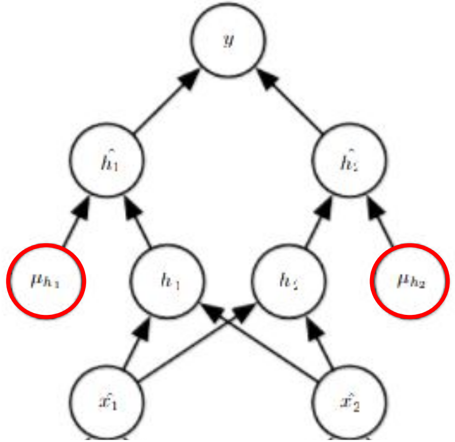
$p(\boldsymbol{\mu})$ - Distribution used to sample $\boldsymbol{\mu}$

- Not easy to evaluate, why?
- Do sample averaging

*Slide courtesy, Ian Goodfellow et al., deep learning book

Regularization - other methods





Drop-out (Srivastava et al., 2014)

Stochastically turn the activation of the hidden unit off with a probability, ***p***

unnormalized probability

$$\tilde{p}_{\text{ensemble}}(y \mid \boldsymbol{x}) = \sqrt[2^d]{\prod_{\boldsymbol{\mu}} p(y \mid \boldsymbol{x}, \boldsymbol{\mu})}$$

Uniform probability of masking

$$p_{\text{ensemble}}(y \mid \boldsymbol{x}) = \frac{\tilde{p}_{\text{ensemble}}(y \mid \boldsymbol{x})}{\sum_{y'} \tilde{p}_{\text{ensemble}}(y' \mid \boldsymbol{x})}.$$

*Slide courtesy, Ian Goodfellow et al., deep learning book

Drop-out (Srivastava et al., 2014)

In case of linear hidden units, the weight scale inference is exact.
For example, consider a softmax regression classifier

$$P(\mathrm{y} = y \mid \mathbf{v}) = \mathrm{softmax}\left(\boldsymbol{W}^{\top}\mathbf{v} + \boldsymbol{b}\right)_y$$

$$P(\mathrm{y} = y \mid \mathbf{v}; \boldsymbol{d}) = \mathrm{softmax}\left(\boldsymbol{W}^{\top}(\boldsymbol{d} \odot \mathbf{v}) + \boldsymbol{b}\right)_y$$

$$\tilde{P}_{\text{ensemble}}(\mathrm{y} = y \mid \mathbf{v}) = \sqrt[2^n]{\prod_{\boldsymbol{d} \in \{0,1\}^n} P(\mathrm{y} = y \mid \mathbf{v}; \boldsymbol{d})}$$

$$= \sqrt[2^n]{\prod_{\boldsymbol{d} \in \{0,1\}^n} \mathrm{softmax}\left(\boldsymbol{W}(\boldsymbol{d} \odot \mathbf{v}) + \boldsymbol{b}\right)_y}$$

$$= \sqrt[2^n]{\prod_{\boldsymbol{d} \in \{0,1\}^n} \frac{\exp\left(\boldsymbol{W}_{y,:}^{\top}(\boldsymbol{d} \odot \mathbf{v}) + b_y\right)}{\sum_{y'} \exp\left(\boldsymbol{W}_{y',:}^{\top}(\boldsymbol{d} \odot \mathbf{v}) + b_{y'}\right)}}$$

$$= \frac{\sqrt[2^n]{\prod_{\boldsymbol{d} \in \{0,1\}^n} \exp\left(\boldsymbol{W}_{y,:}^{\top}(\boldsymbol{d} \odot \mathbf{v}) + b_y\right)}}{\sqrt[2^n]{\prod_{\boldsymbol{d} \in \{0,1\}^n} \sum_{y'} \exp\left(\boldsymbol{W}_{y',:}^{\top}(\boldsymbol{d} \odot \mathbf{v}) + b_{y'}\right)}}$$

$$\tilde{P}_{\text{ensemble}}(\mathrm{y} = y \mid \mathbf{v}) \propto \sqrt[2^n]{\prod_{\boldsymbol{d} \in \{0,1\}^n} \exp\left(\boldsymbol{W}_{y,:}^{\top}(\boldsymbol{d} \odot \mathbf{v}) + b_y\right)}$$

$$= \exp\left(\frac{1}{2^n}\sum_{\boldsymbol{d} \in \{0,1\}^n} \boldsymbol{W}_{y,:}^{\top}(\boldsymbol{d} \odot \mathbf{v}) + b_y\right)$$

$$= \exp\left(\underbrace{\frac{1}{2}\boldsymbol{W}_{y,:}^{\top}}_{\text{Weight rescale}}\mathbf{v} + b_y\right).$$

*Slide courtesy, Ian Goodfellow et al., deep learning book

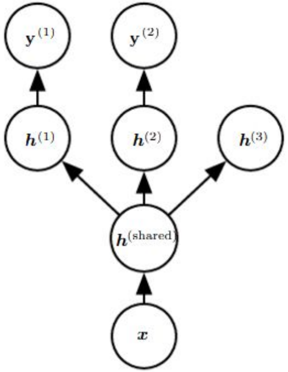
Regularization - other methods

Dataset augmentation

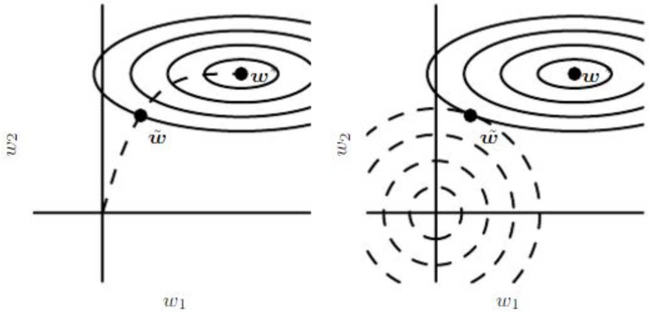
Flipping the image for classification

*pic courtesy, web

Multi-task learning



Early stopping



Parameter sharing and tying

Most extensively employed with
Convolutional Neural Nets (CNN)

*Slide courtesy, Ian Goodfellow et al., deep learning book

End