Regularization

EE 5179: Deep Learning for Imaging

Instructor: Kaushik Mitra

Generalization

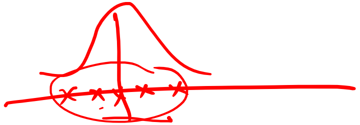
(x_i, y_i)

Train : (x_i, y_i), $i = 1 \dots N$

Testing : (x_i', y_i'), $i = 1 \dots M$

$f(x, y) \leftarrow$

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 $\min_w$

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

$$X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix}$$

$$\min_w \; E_{x,y} \left[||x^T w - y||^2 \right]$$

$$\iint_{x,y} ||x^T w - y||^2 f(x,y) \, dx \, dy$$

What we actually want

Test err (or) Generalization err

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

$$E_x[g(x)] = \int f(x) g(x) \, dx$$

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

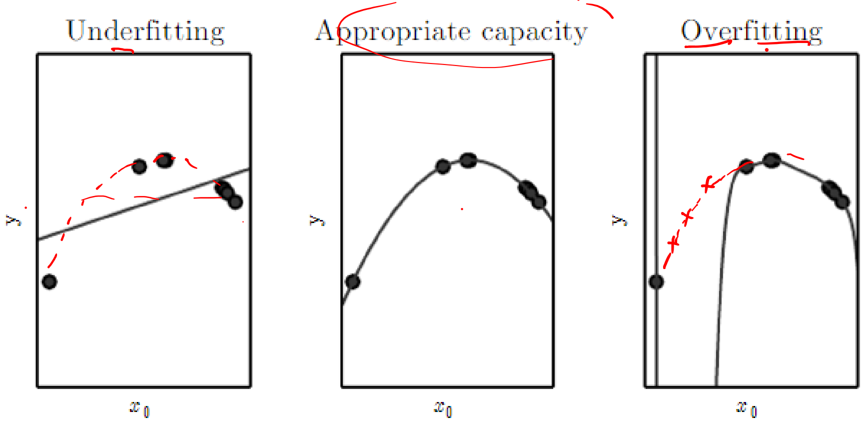
Statistical learning theory

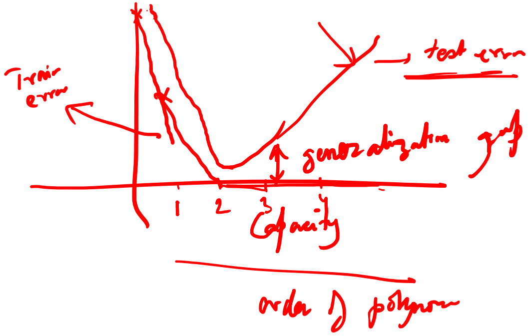
– Training and Test sets are not arbitrary
– Underlying *data generating distribution* is same

*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*





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

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

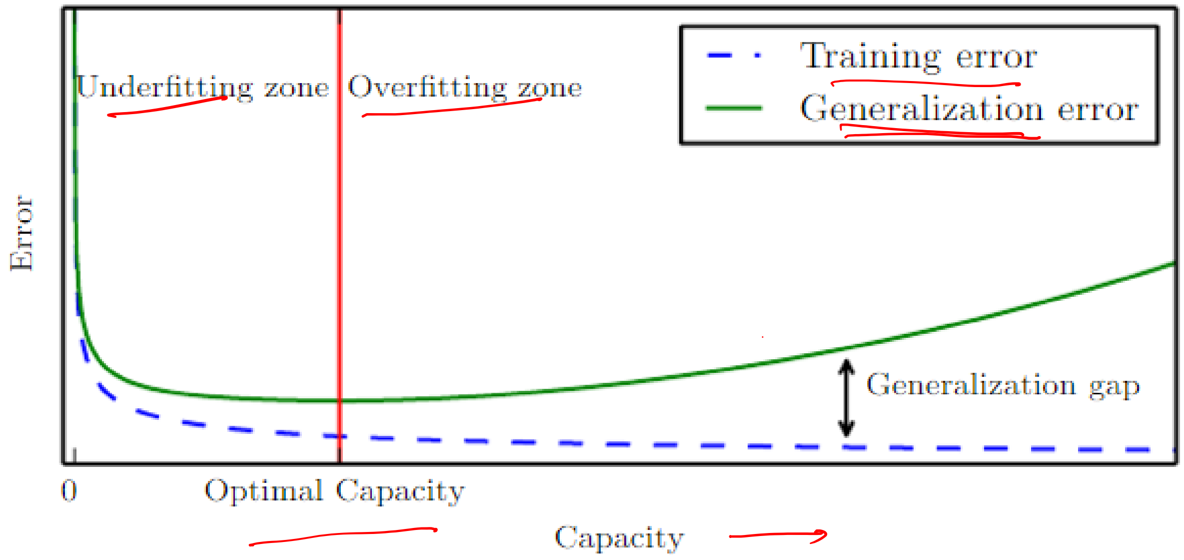
$$\hat{y} = b + \sum_{i=1}^{9} w_i x^i.$$

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

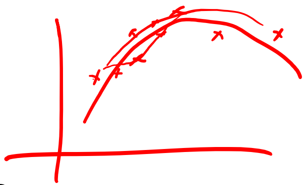
Capacity, Overfitting and Underfitting

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



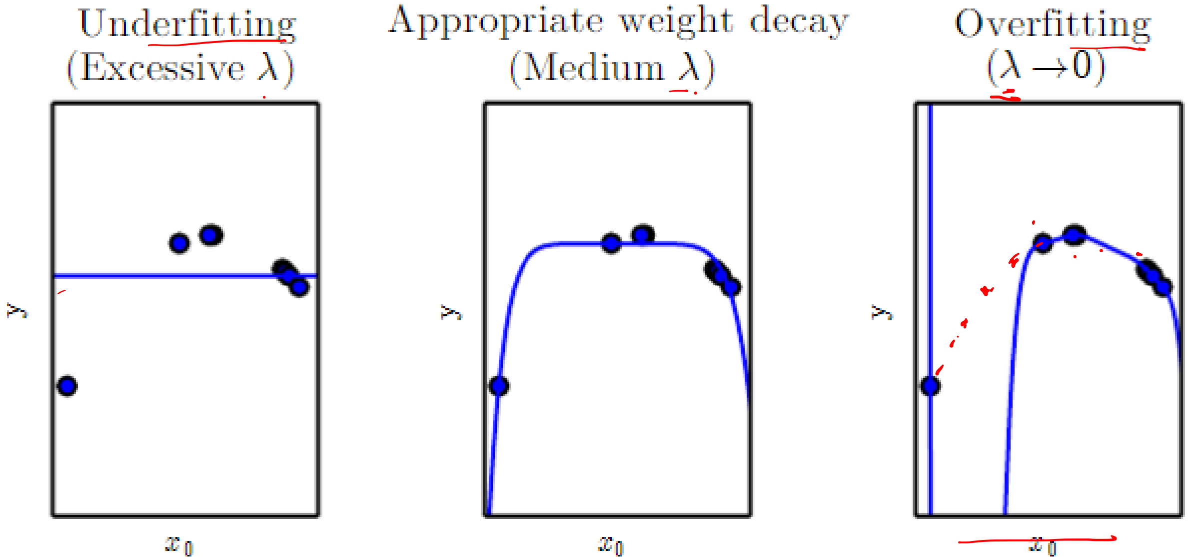
Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error

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

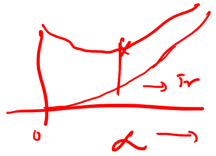
$y = w^T x + b$

Note that *biases* in general are not penalized

$\lambda \to \infty \Rightarrow w \to 0$



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



$\theta = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_n \end{pmatrix}$

Regularization - parameter norm penalties

w, b

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

$\|\theta\|_2 = \sqrt{|\theta_1|^2 + |\theta_2|^2 + \dots + |\theta_n|^2}$

$\|\theta\|_1 = |\theta_1| + |\theta_2| + \dots$

$\|\theta\|_p = \sqrt[p]{|\theta_1|^p + |\theta_2|^p + \dots}$

L_2 norm regularization
(weight decay, ridge regression)

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

$= \|w\|_2^2$

$\nabla_w \tilde{J} = \nabla_w \frac{\alpha}{2} w^T w + \nabla_w J$

$= \alpha w + \nabla_w J(w)$

Parameter update:

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

$w \leftarrow w - \epsilon \nabla_w J(w)$

$$w \leftarrow w - \epsilon \left(\alpha w + \nabla_w J(w; X, y) \right).$$

$$w \leftarrow (1 - \epsilon\alpha) w - \epsilon \nabla_w J(w; X, y).$$

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

Regularization - parameter norm penalties

L_2 norm regularization $\tilde{J}(w; X, y) = \frac{\alpha}{2} w^\top w + J(w; X, y)$,

Consider: $w^* = \arg\min_w J(w)$. *Unregularized solution*

- *Taylor expansion of J at w*,*

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

H is hessian of J wrt w at w^*; ***linear term?***

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

- *Add the L_2 regularization grad*

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

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

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

- *Since H is real and symmetric*

$$\tilde{w} = (Q\Lambda Q^\top + \alpha I)^{-1} Q\Lambda Q^\top w^*$$

$$= \left[Q(\Lambda + \alpha I)Q^\top\right]^{-1} Q\Lambda Q^\top w^*$$

$$= Q\underbrace{(\Lambda + \alpha I)^{-1}\Lambda}_{\lambda_i / (\lambda_i + \alpha)} Q^\top w^*.$$

$f(x_1, x_2)$ $H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial f \partial f}{\partial x_1 \partial x_2} \\ \frac{\partial f \partial f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix}$

$\tilde{w} = \arg\min_w \tilde{J}(w)$

$\tilde{w} = Q w^*$

$QQ^T = I$

$\Lambda = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_N \end{bmatrix}$

$Q = [v_1 \cdots v_N]$ $Q^T Q = I$

$\tilde{J}(w) = J(w^*) + \frac{1}{2}(w - w^*)^T H (w - w^*) + \frac{\alpha}{2} w^T w$

$\nabla \tilde{J}(\tilde{w}) = 0$

$\nabla \tilde{J}(w) = H(w - w^*) + \alpha w$

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

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

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

assume $Q = I$ $H = Q\Lambda Q^T$

$\begin{bmatrix} \frac{\lambda_1}{\lambda_1 + \alpha} & & \\ & \ddots & \\ & & \end{bmatrix} w^*$ $\lambda \geq 0$

$\tilde{w}_i = \frac{\lambda_i}{\lambda_i + \alpha} w_i^*$

$w = w^* + w - w^*$

$\hat{J}(w) = J(w^*) + \frac{1}{2}(w - w^*)^T H (w - w^*)$

$f(x + \Delta x) = f(x) + \Delta x f'(x) + \frac{\Delta x^2}{2!} f''(x) + \frac{\Delta x^3}{3!} f'''(x)$

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

Regularization - parameter norm penalties

$y = w^T x$

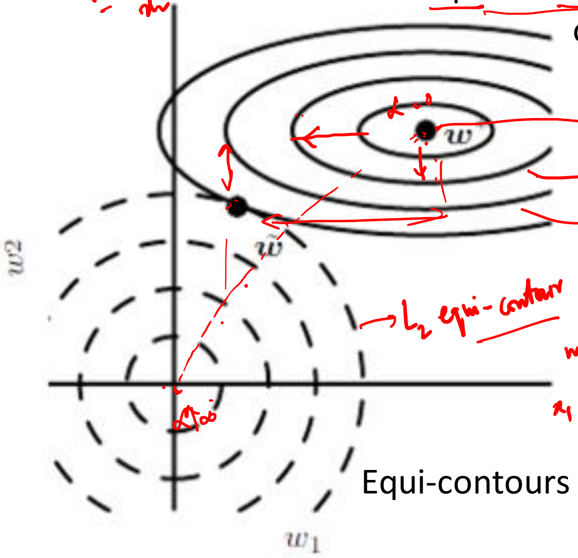
$w_1 x_1 + w_2 x_2 + \dots$

L_2 norm regularization

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

$w^* = \arg\min_w J(w)$

Equi-contours of unregularized objective J



$\min_w \frac{\alpha}{2} w^T w$

$\lambda_2 > \lambda_1$

Equi-contours of L_2 regularizer

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

$$\tilde{w} = (Q\Lambda Q^\top + \alpha I)^{-1} Q\Lambda Q^\top w^*$$

$$= \left[Q(\Lambda + \alpha I)Q^\top\right]^{-1} Q\Lambda Q^\top w^*$$

$$= Q\underbrace{(\Lambda + \alpha I)^{-1}\Lambda}_{\lambda_i / (\lambda_i + \alpha)} Q^\top w^*.$$

Regularizer effectively rescales $\boldsymbol{w^*}$ 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

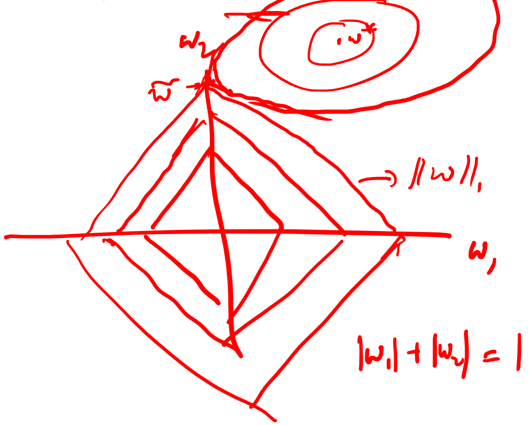
L_1 norm regularization

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

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

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

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



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

Regularization - parameter norm penalties

L_1 norm regularization

$\tilde J(w) = J(w) + \alpha\|w\|_1$

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

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

$$\boxed{\textbf{\textit{H}} \text{ is diagonal, with } h_{ii} > 0, \text{ for all } i}$$

$\sum \frac{1}{2} H_{ii}(w_i - w_i^*)^2$

- Revisiting L_2 with diagonal ***H***

$\nabla \tilde J(\tilde w) = 0$

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

$H_{ii}(w_i - w_i^*) + \alpha\, \mathrm{sign}(w_i)$

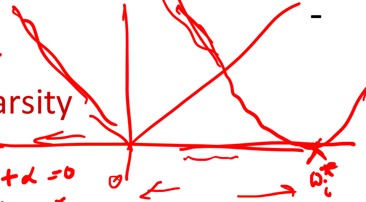
$\nabla \tilde J(\tilde w) = 0$

$\frac{d|x|}{dx} = \begin{cases} +1 & \text{if } x > 0 \\ -1 & \text{if } x < 0 \end{cases}$ sign(x)

Still it only scales the weights - not sparsity

$H_{ii}(\tilde w_i - w_i^*) + \alpha = 0$

$\tilde w_i = w_i^* - \alpha/H_{ii}$



- *Now, the* L_1 *regularized objective*

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

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

$w_i^* > \alpha/H_{ii}$

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

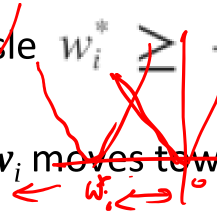
$w_i^* < \alpha/H_{ii}$ $\tilde w_i = 0$

$w_i^* >, 0$

  - if $w_i^* \le \frac{\alpha}{H_{ii}}$; w_i $\max\left\{w_i^* - \frac{\alpha}{H_{ii}}, 0\right\}$
  - esle $w_i^* \ge \frac{\alpha}{H_{ii}}$

$w_i^* < 0$ $|w_i^*| > \frac{\alpha}{H_{ii}}$

$\mathrm{sign}(w_i^*)\left(|w_i^*| - \frac{\alpha}{H_{ii}}\right)$



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

$|w_i^*| < \alpha/H_{ii}$

L_1 regularization enforces **sparsity** in the solution

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

Regularization - other methods

Ensemble methods

Bagging - bootstrap aggregating

$y_1 = x^T w_1 + \epsilon_1$

$y = x^T w_2 + \epsilon_2$

$y = x^T w_k + \epsilon_k$

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

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

$c \le v$

- *Avg. error by **k** models*

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

$(\sum \epsilon_i)^2$, $(\epsilon_1 + \dots + \epsilon_k)(\epsilon_1 + \dots + \epsilon_k)$

$\frac{1}{k^2}\left(k\,E(\epsilon_i^2) + k(k-1)c\right)$

$\frac{v}{k} + \frac{(k-1)}{k}c$

- *Expected squared error of the ensemble predictor*

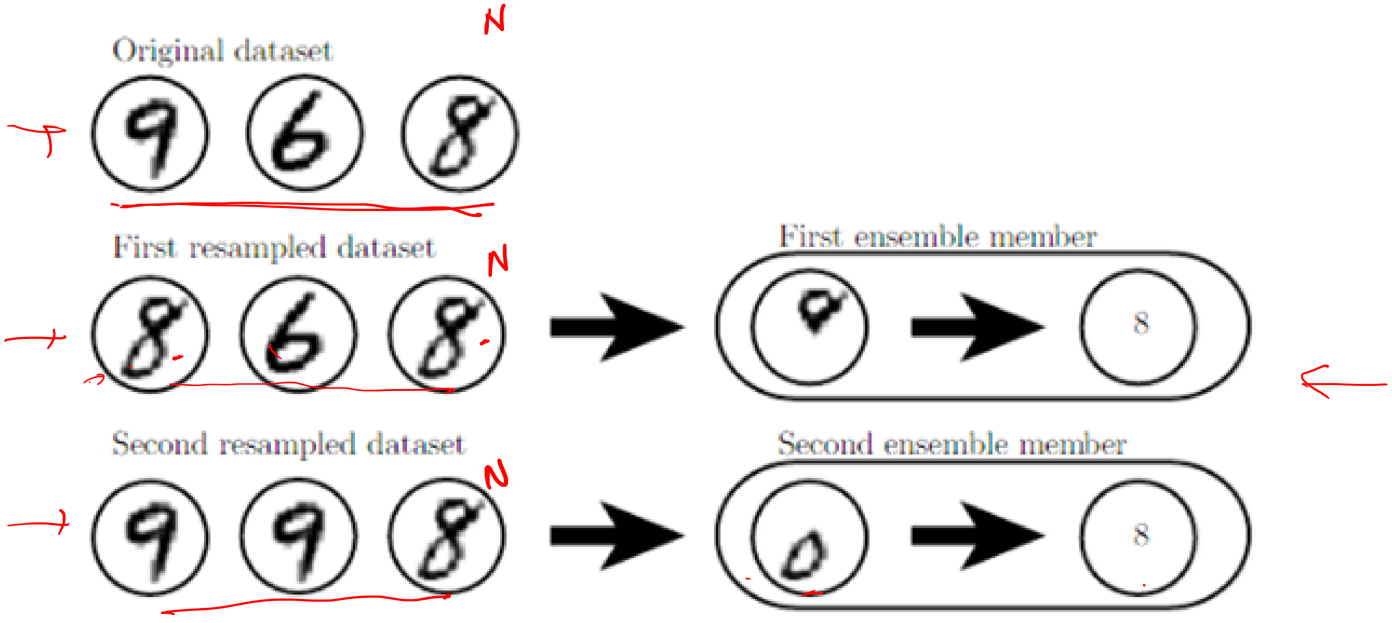
$$\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{1}{k^2}\mathbb{E}\left[\sum_i\left(\epsilon_i^2 + \sum_{j\neq i}\epsilon_i\epsilon_j\right)\right]$$

$$= \frac{1}{k}v + \frac{k-1}{k}c.$$

  - 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 certain probability

$$h^{(k)} = f(Wh^{(k-1)} + b^{(k-1)})$$

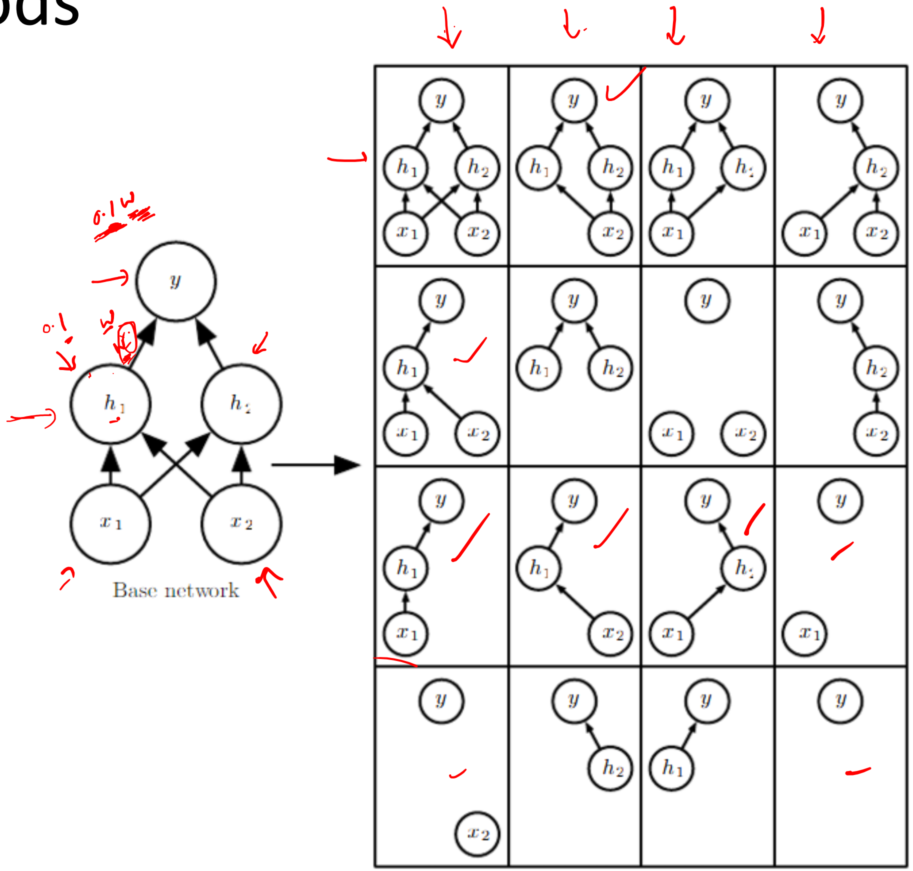
$$\hat{h}^{(k)} = \mu^{(k)} \odot h^{(k)}$$

$\{0, 1\}$

$p(\mu) = \lambda$

$0 \leq \lambda \leq 1$

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

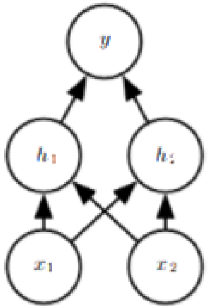


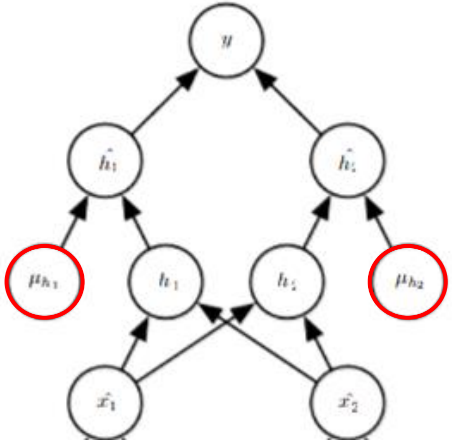
Base network

Ensemble of subnetworks

*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 certain probability

$$h^{(k)} = f(Wh^{(k-1)} + b^{(k-1)})$$

$$\hat{h}^{(k)} = \mu^{(k)} \odot h^{(k)}$$

– Inference, $p(y|x)$

Bagging

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

Drop-out

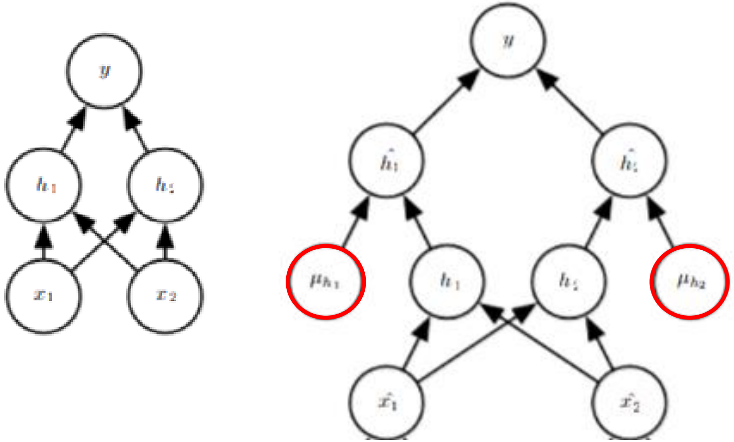
$$\sum_{\boldsymbol{\mu}} p(\boldsymbol{\mu}) p(y \mid \boldsymbol{x}, \boldsymbol{\mu})$$

$p(\boldsymbol{\mu})$ - 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 certain probability

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

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

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 certain probability

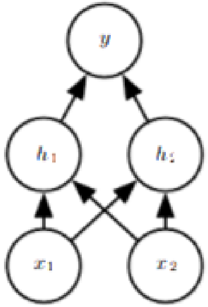
$$h^{(k)} = f(Wh^{(k-1)} + b^{(k-1)})$$

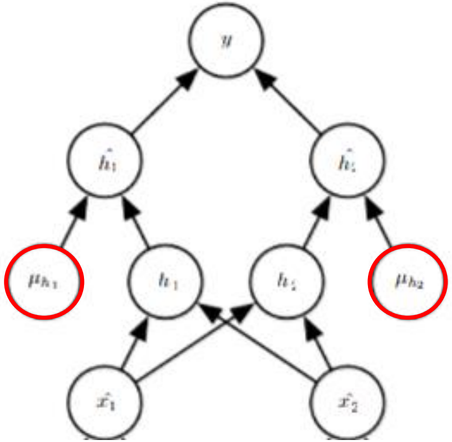
$$\hat{h}^{(k)} = \mu^{(k)} \odot h^{(k)}$$

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 $\boldsymbol{\mu}$

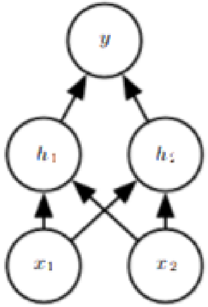
- Not easy to evaluate, why?
- Do sample averaging

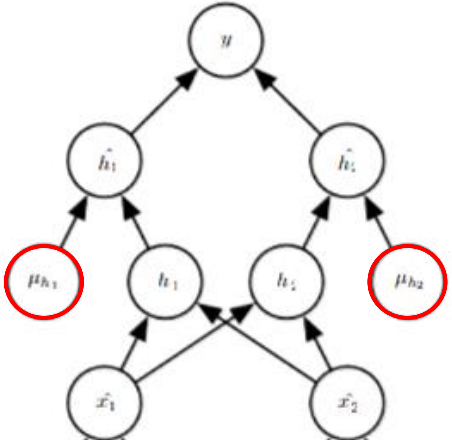
*Slide courtesy, Ian Goodfellow et al., deep learning book

Regularization - other methods

Drop-out (Srivastava et al., 2014)

Stochastically turn the activation of the hidden unit off with a certain probability





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

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

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

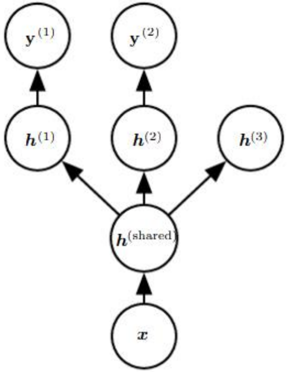
$$\begin{aligned}
\tilde{P}_{\text{ensemble}}(\mathrm{y}=y \mid \mathbf{v}) &= \sqrt[2^n]{\prod_{\boldsymbol{d}\in\{0,1\}^n} P(\mathrm{y}=y \mid \mathbf{v};\boldsymbol{d})} \\
&= \sqrt[2^n]{\prod_{\boldsymbol{d}\in\{0,1\}^n} \operatorname{softmax}\left(\boldsymbol{W}(\boldsymbol{d}\odot\mathbf{v}) + \boldsymbol{b}\right)_y} \\
&= \sqrt[2^n]{\prod_{\boldsymbol{d}\in\{0,1\}^n} \frac{\exp\left(\boldsymbol{W}_{y,:}^\top(\boldsymbol{d}\odot\mathbf{v}) + b_y\right)}{\sum_{y'}\exp\left(\boldsymbol{W}_{y',:}^\top(\boldsymbol{d}\odot\mathbf{v}) + b_{y'}\right)}} \\
&= \frac{\sqrt[2^n]{\prod_{\boldsymbol{d}\in\{0,1\}^n}\exp\left(\boldsymbol{W}_{y,:}^\top(\boldsymbol{d}\odot\mathbf{v}) + b_y\right)}}{\sqrt[2^n]{\prod_{\boldsymbol{d}\in\{0,1\}^n}\sum_{y'}\exp\left(\boldsymbol{W}_{y',:}^\top(\boldsymbol{d}\odot\mathbf{v}) + b_{y'}\right)}}
\end{aligned}$$

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

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

Regularization - other methods

Multi-task learning

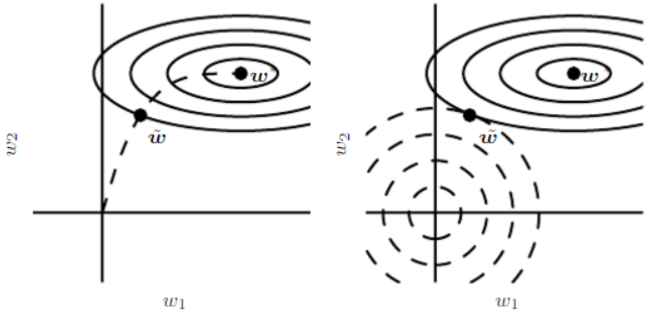


Dataset augmentation

Flipping the image for classification

*pic courtesy, [web]

Early stopping



Parameter sharing and tying

Most extensively employed with
Convolutional Neural Nets (CNN)

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

End