Regularization（正則化） for Deep Learning

Chapter 7

楊　森（Sen Yang） 2019/3/4

基礎①：Capacity（容量），Overfitting and Underfitting

- 仮説：data-generating process: i.i.d. assumptions (independent, identically distributed（training setとtest set）)

- data-generating distribution: p_{data}

- 目標：make the training error small→課題：underfitting

- 目標：make the gap between training and test error small→課題：overfitting

- capacity: hypothesis space（表現力）、representational capacity、effective capacity

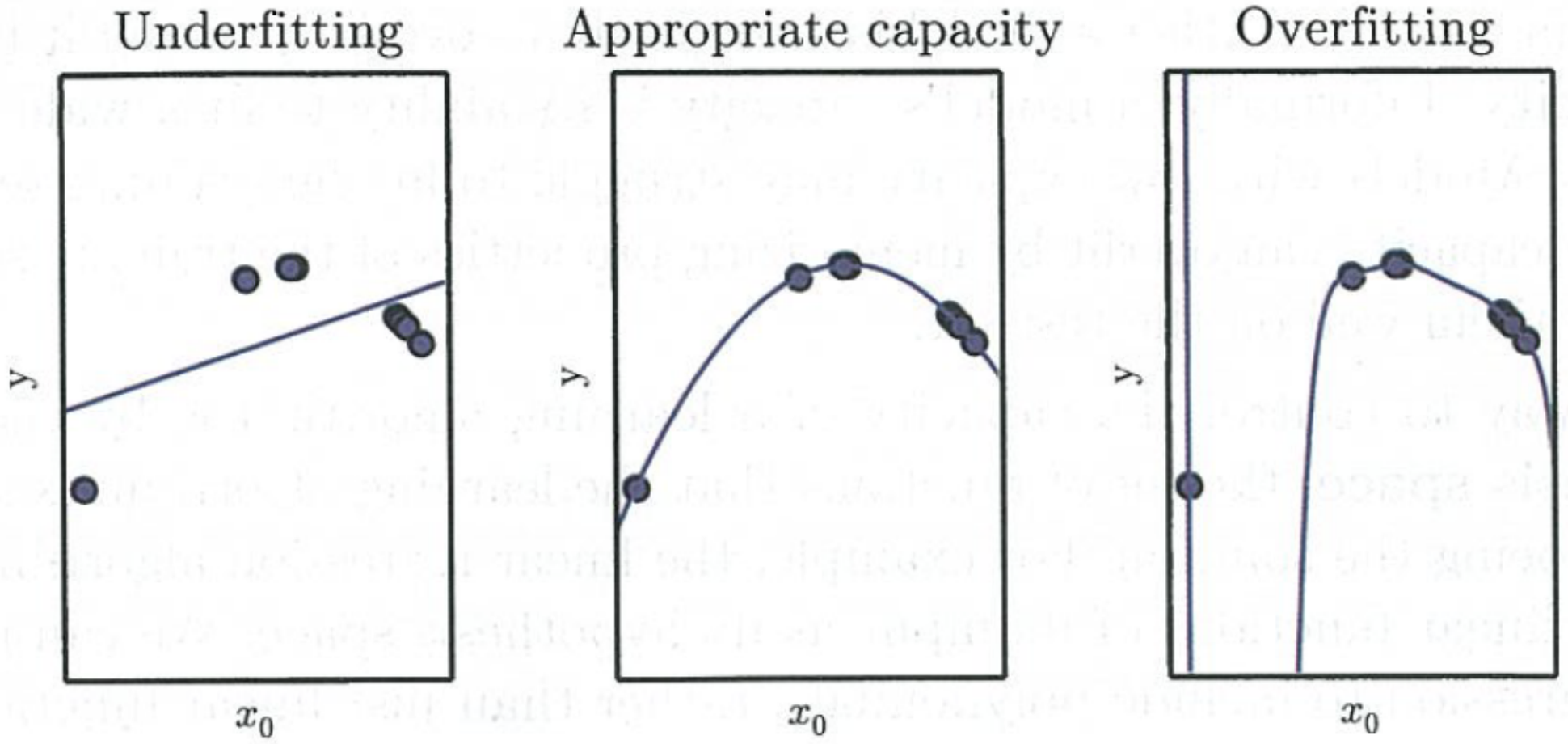


Figure 5.2: We fit three models to this example training set. The training data was generated synthetically, by randomly sampling x values and choosing y deterministically by evaluating a quadratic function. *(Left)*A linear function fit to the data suffers from underfitting—it cannot capture the curvature that is present in the data. *(Center)*A quadratic function fit to the data generalizes well to unseen points. It does not suffer from a significant amount of overfitting or underfitting. *(Right)*A polynomial of degree 9 fit to the data suffers from overfitting. Here we used the Moore-Penrose pseudoinverse to solve the underdetermined normal equations. The solution passes through all the training points exactly, but we have not been lucky enough for it to extract the correct structure. It now has a deep valley between two training points that does not appear in the true underlying function. It also increases sharply on the left side of the data, while the true function decreases in this area.

MSE (mean squared error)

Training error

$$\frac{1}{m^{(train)}}\|\hat{y}^{(train)} - y^{(train)}\|_2^2$$

Generalization error

$$\frac{1}{m^{(test)}}\|\hat{y}^{(test)} - y^{(test)}\|_2^2$$

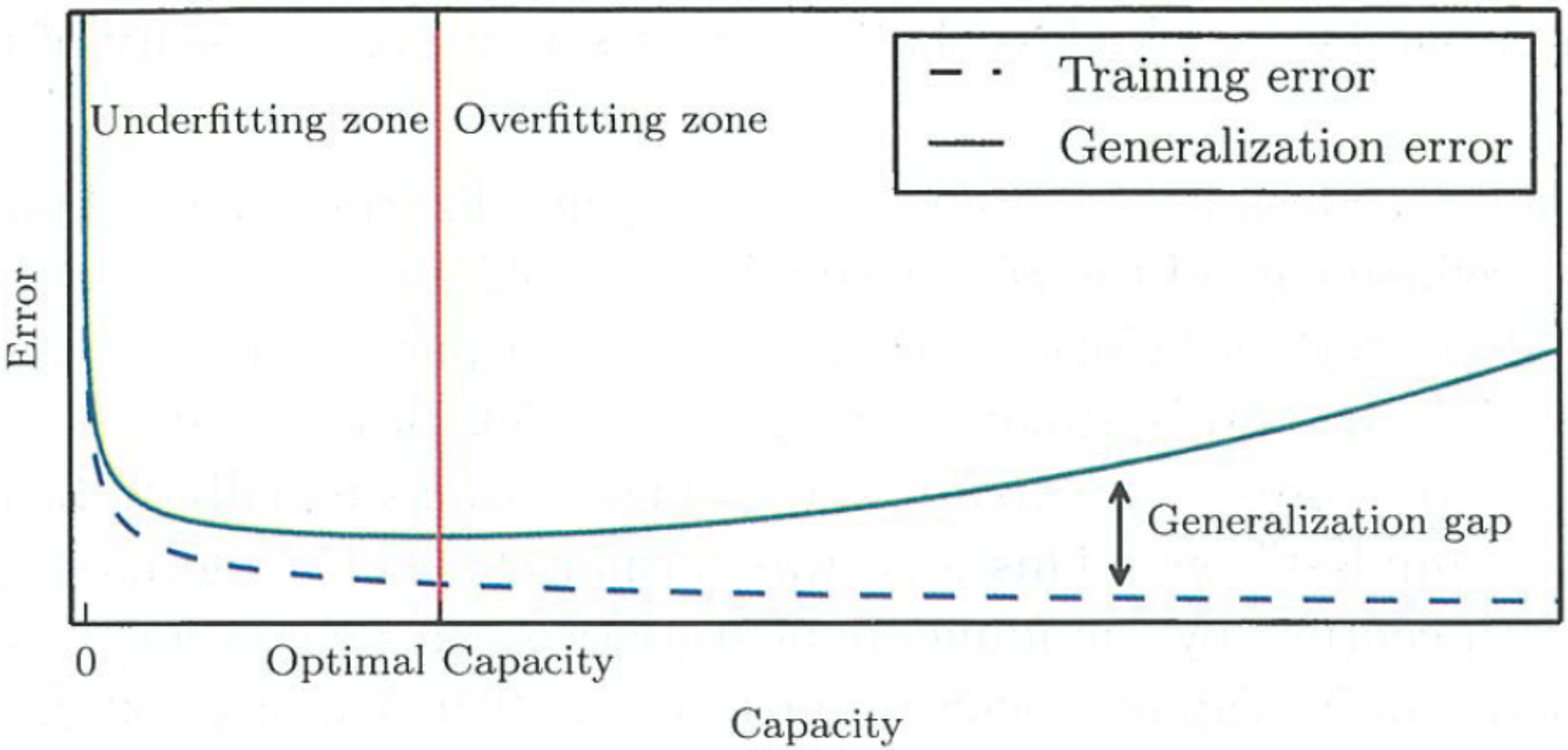
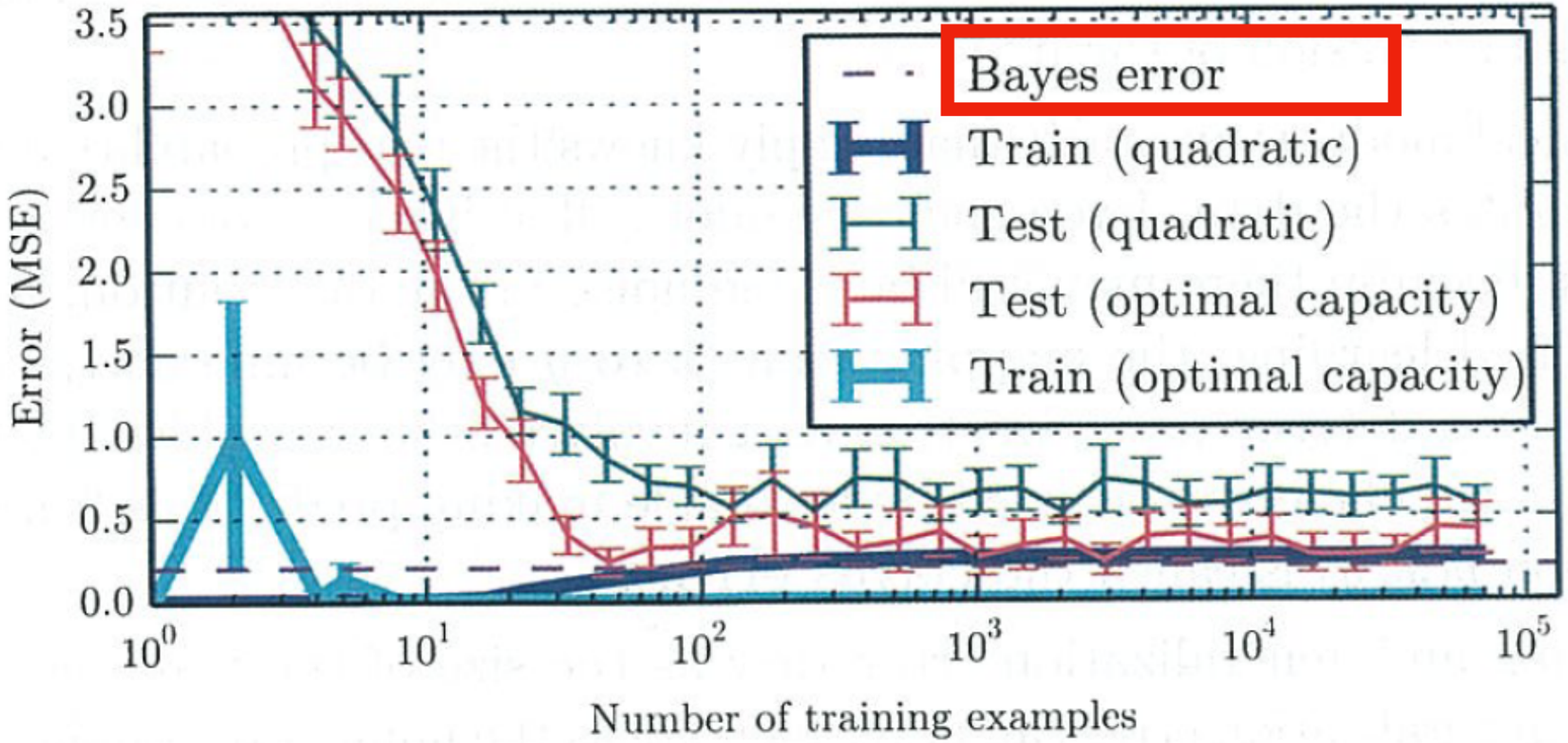
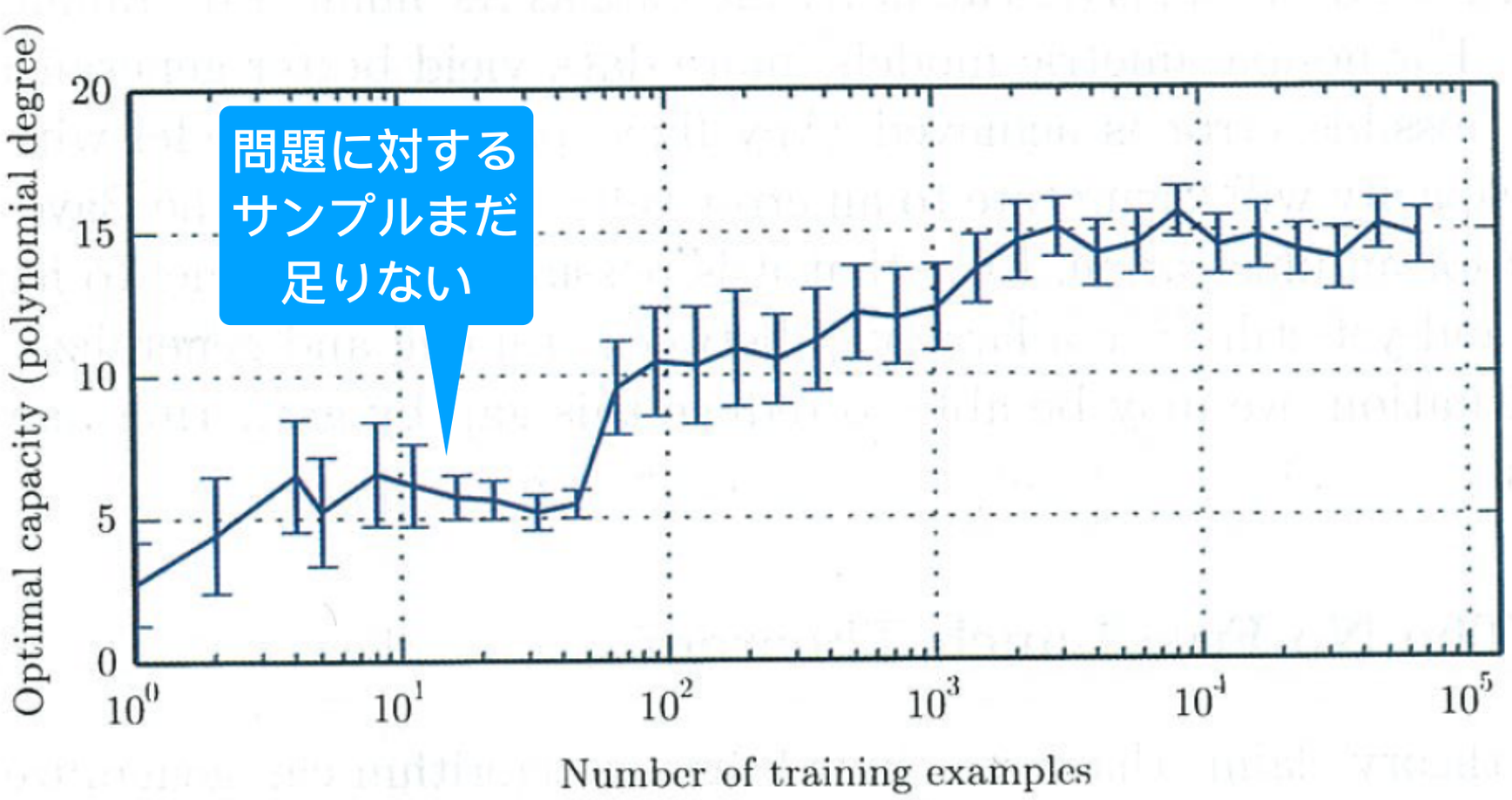


Figure 5.3: Typical relationship between capacity and error. Training and test error behave differently. At the left end of the graph, training error and generalization error are both high. This is the **underfitting regime**. As we increase capacity, training error decreases, but the gap between training and generalization error increases. Eventually, the size of this gap outweighs the decrease in training error, and we enter the **overfitting regime**, where capacity is too large, above the **optimal capacity**.





Bayes error: the error incurred by making predictions from the true distribution p(x,y)
(本当の分布から予測しても生じる誤差)

原因①：the mapping from x to y may be inherently stochastic.（本質的な確率的なプロセス）

原因②：y may be a deterministic function that involves other variables besides those included in x.（x以外の影響要素も存在するかも）

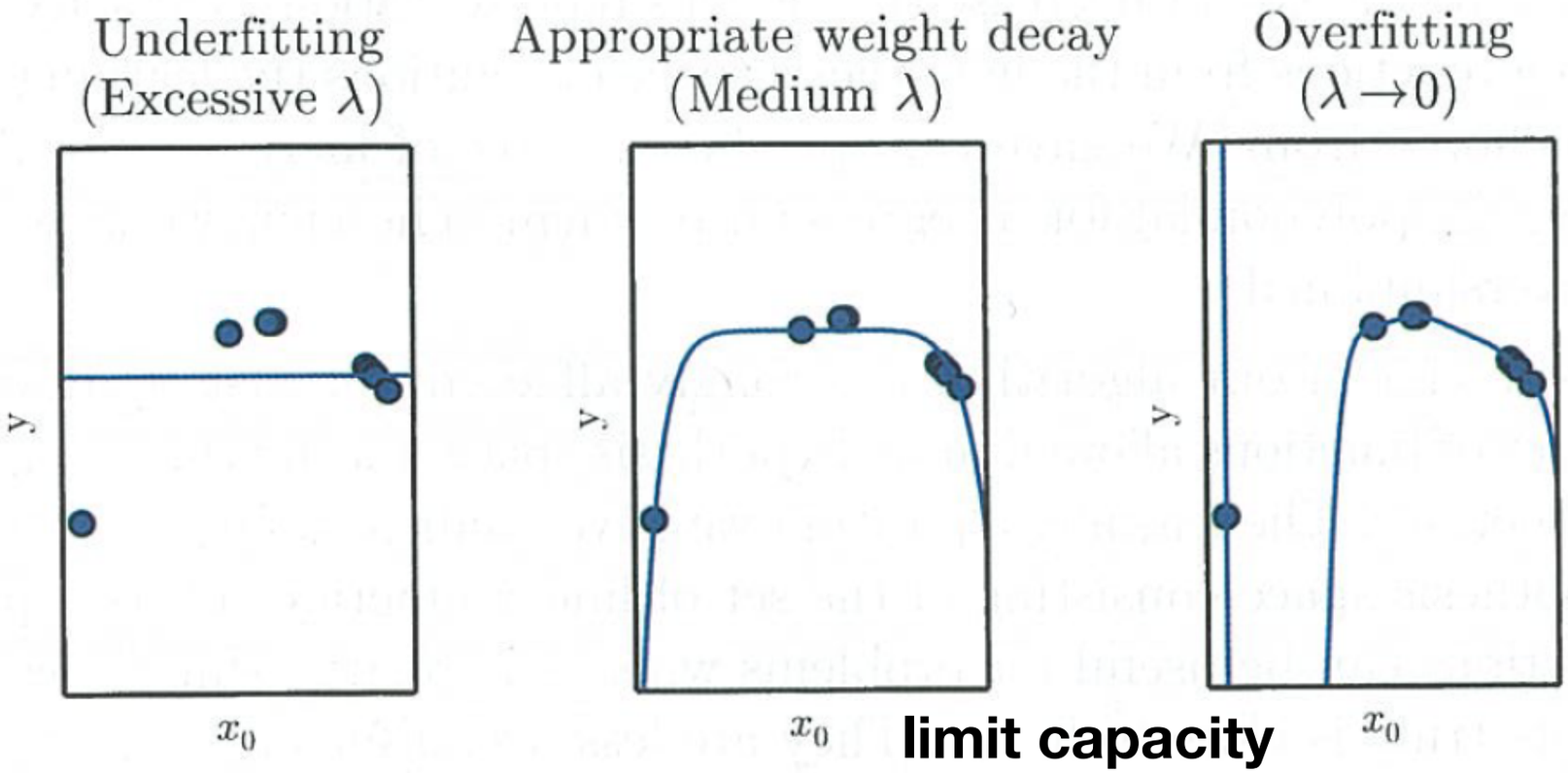
Figure 5.4: The effect of the training dataset size on the train and test error, as well as on the optimal model capacity. We constructed a synthetic regression problem based on

squared L^2 norm

$$J(\omega) = MSE_{train} + \lambda\omega^T\omega$$

weight decay（重み減衰）： tradeoff between fitting the training data and being small

regularizer / regularization : express preferences（傾向）



limit capacity

Figure 5.5: We fit a high-degree polynomial regression model to our example training set from figure 5.2. The true function is quadratic, but here we use only models with degree 9. We vary the amount of weight decay to prevent these high-degree models from overfitting. *(Left)*With very large λ, we can force the model to learn a function with no slope at all. This underfits because it can only represent a constant function. *(Center)*With a medium value of λ, the learning algorithm recovers a curve with the right general shape. Even though the model is capable of representing functions with much more complicated shapes, weight decay has encouraged it to use a simpler function described by smaller coefficients. *(Right)*With weight decay approaching zero (i.e., using the Moore-Penrose pseudoinverse to solve the underdetermined problem with minimal regularization), the degree-9 polynomial overfits significantly, as we saw in figure 5.2.

Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.
（正則化は、training errorではなく、generalization errorを減らす手法。training errorを減らすのは、Optimization）

p117

基礎②：Estimators（推定量、推定器），Bias and Variance

- Bias定義：$bias(\hat{\theta}_m) = \mathbb{E}(\hat{\theta}_m) - \theta$

θ: data-generating distributionを定義するパラメータ

- Variance定義：$Var(\hat{\theta})$

- Bias measures the expected deviation from the true value of the function or parameter. Biasは本当の値からの予想中偏差を評価する。

- Variance measures the deviation from the expected estimator value that any particular sampling of the data is likely to cause. Varianceは、予想中値を中心として、サンプリング結果がどのぐらい分散するかを評価する。

- MSEがBiasとVariance両方を評価できる

$$MSE = \mathbb{E}[(\hat{\theta}_m - \theta)^2] = Bias(\hat{\theta}_m)^2 + Var(\hat{\theta}_m)$$

-

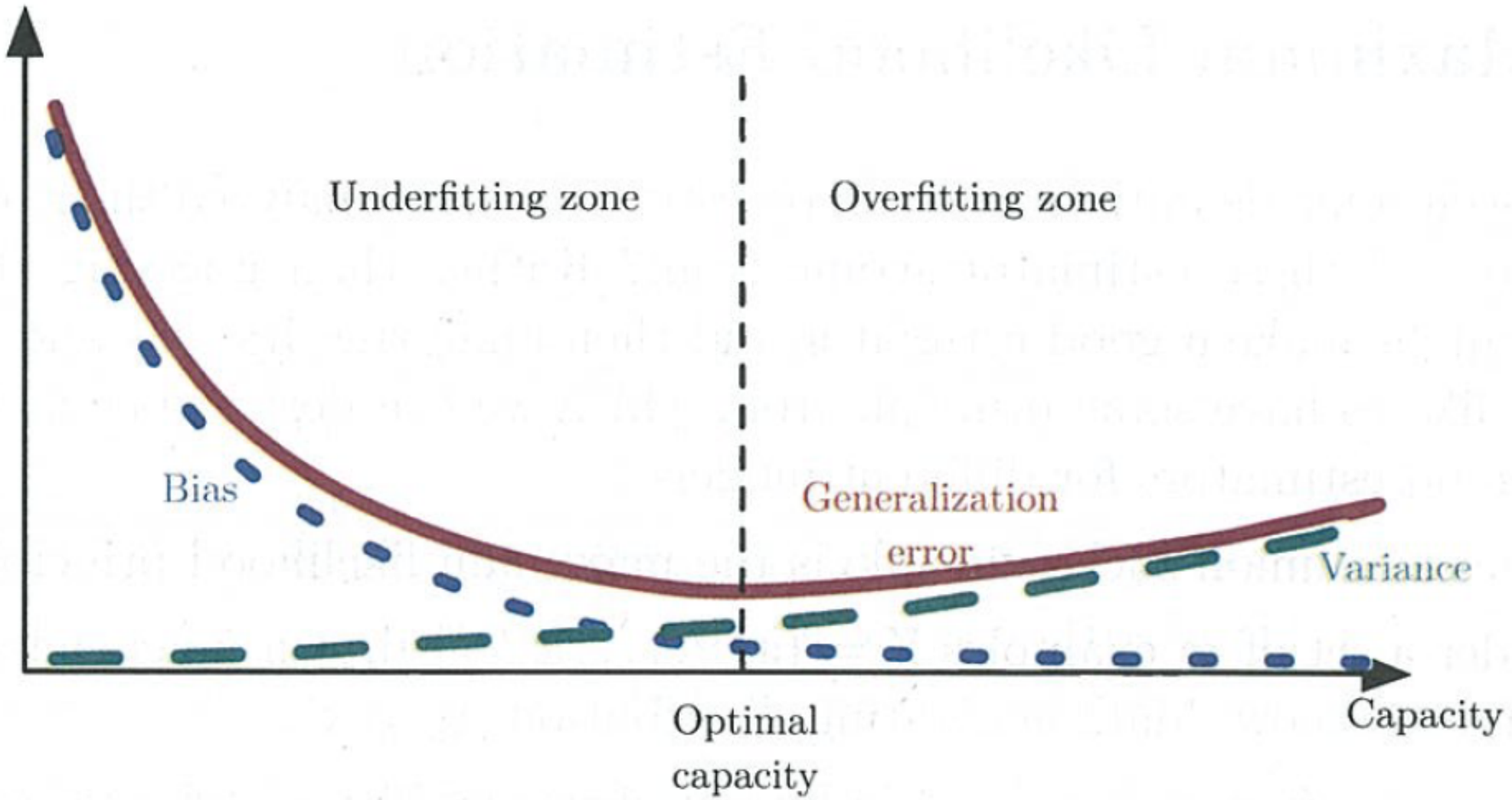


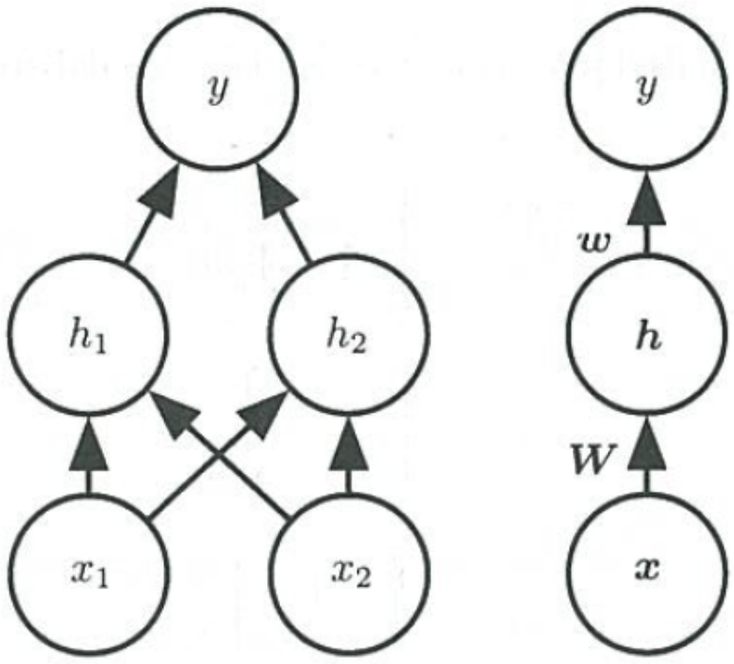
Figure 5.6: As capacity increases (x-axis), bias (dotted) tends to decrease and variance (dashed) tends to increase, yielding another U-shaped curve for generalization error (bold curve). If we vary capacity along one axis, there is an optimal capacity, with underfitting when the capacity is below this optimum and overfitting when it is above. This relationship is similar to the relationship between capacity, underfitting, and overfitting, discussed in section 5.2 and figure 5.3.

Parameter Norm Penalties

- regularized objective function:

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

説明②：全ての層が同じαを共有している



- $h = g(W^T x + b)$

説明①：θで書いてあるが、実際Wだけを正則化する、bはしない
理由①：bはxと関係ない、xがどう変わっても、bによって生じるvarianceがない
理由②：bを正則化すると、大きいunderfittingが生じるかもしれない

7.1 p223

L^2 Parameter Regularizationが何をしている?

①: gradient descent（勾配降下法）への影響

簡略化のため、biasが省略されている

objective function:

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

gradient:

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

update weights:

$$\omega \leftarrow \omega - \epsilon(\alpha\omega + \nabla_\omega J(\omega; X, y))$$

$$\omega \leftarrow (1 - \epsilon\alpha)\omega - \epsilon\nabla_\omega J(\omega; X, y))$$

通常gradient更新をする前に、重みを縮める

L^2 Parameter Regularizationが何をしている?
②Hessian matrix（ヘッセ行列）から見ると

w*はweight decayなしのminimumを得る所の重み。Jacobian matrix（ヤコビ行列）が０。

Hessian matrix: J関数局面のcurvature（曲率）を表す

Taylor expansion（テイラー展開）： $\hat{J}(\omega) = J(\omega^*) + \frac{1}{2}(\omega - \omega^*)^T H(\omega - \omega^*)$

gradient: $\nabla_\omega \hat{J}(\omega) = H(\omega - \omega^*)$

weight decayありのminimum条件:

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

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

Hを分解する（Hは対称matrix）： $H = Q\Lambda Q^T$

orthogonal matrix（直交行列）：列はHの固有ベクトル（eigenvector）

diagonal matrix（対角行列）：(i, i)-要素がHの固有値（eigenvalue）

L^2 Parameter Regularizationが何をしている？③weight decayなしの最適重みへの影響

分解したHを代入する：

$$\begin{aligned}\tilde{\omega} &= (Q\Lambda Q^T + \alpha I)^{-1} Q\Lambda Q^T \omega^* \\ &= [Q(\Lambda + \alpha I)Q^T]^{-1} Q\Lambda Q^T \omega^* \\ &= Q(\Lambda + \alpha I)^{-1}\Lambda Q^T \omega^*\end{aligned}$$

weight decayが何をしているかと言うと：

① w*を、Hの固有ベクトルが定義している方向で、rescaleする。具体的に、w*の、i番目固有ベクトル方向での成分を、$\frac{\lambda_i}{\lambda_i + \alpha}$ 倍にrescaleする。

② $\lambda_i \gg \alpha$ の場合、この成分がほとんど変わらない。

③ $\lambda_i \ll \alpha$ の場合、この成分がほぼ0までになっちゃう。

曲面の曲率が緩やかな方向の成分は、0近い所に引かれた。

曲面の曲率が急な方向の成分への影響が小さい

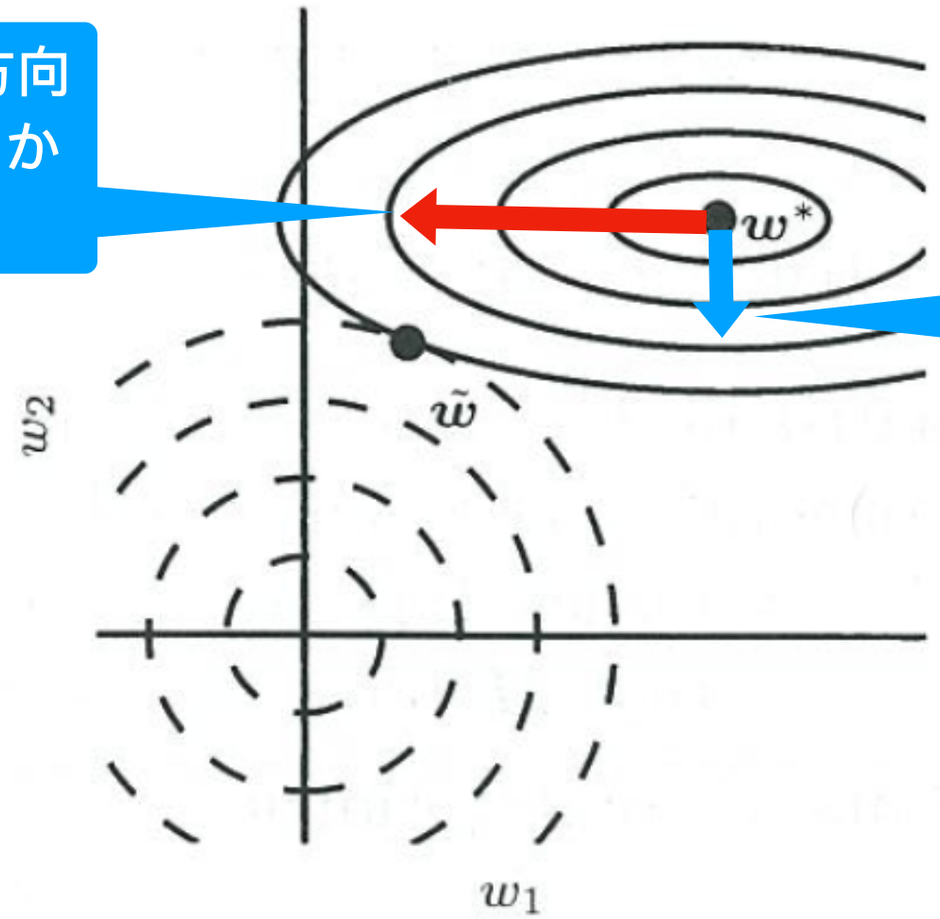


Figure 7.1: An illustration of the effect of L^2 (or weight decay) regularization on the value of the optimal $\boldsymbol{w}$. The solid ellipses represent contours of equal value of the unregularized objective. The dotted circles represent contours of equal value of the L^2 regularizer. At the point $\tilde{\boldsymbol{w}}$, these competing objectives reach an equilibrium. In the first dimension, the eigenvalue of the Hessian of J is small. The objective function does not increase much when moving horizontally away from $\boldsymbol{w}^*$. Because the objective function does not express a strong preference along this direction, the regularizer has a strong effect on this axis. The regularizer pulls w_1 close to zero. In the second dimension, the objective function is very sensitive to movements away from $\boldsymbol{w}^*$. The corresponding eigenvalue is large, indicating high curvature. As a result, weight decay affects the position of w_2 relatively little.

L^1 Regularizationが何をしている?

重み絶対値の足し算

objective function:

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

gradient:

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

weight decayなしの最適重みへの影響

$$\omega_i = sign(\omega_i^*) max\{|\omega_i^*| - \frac{\alpha}{H_{i,i}}, 0\}$$

L1正則化の一番大切なプロパーティー：
Sparsity。（一部重みが0になる。）
L2正則化では、０近い所に引かれるが、
0にはならない

Dataset Augmentation（データセット増強）

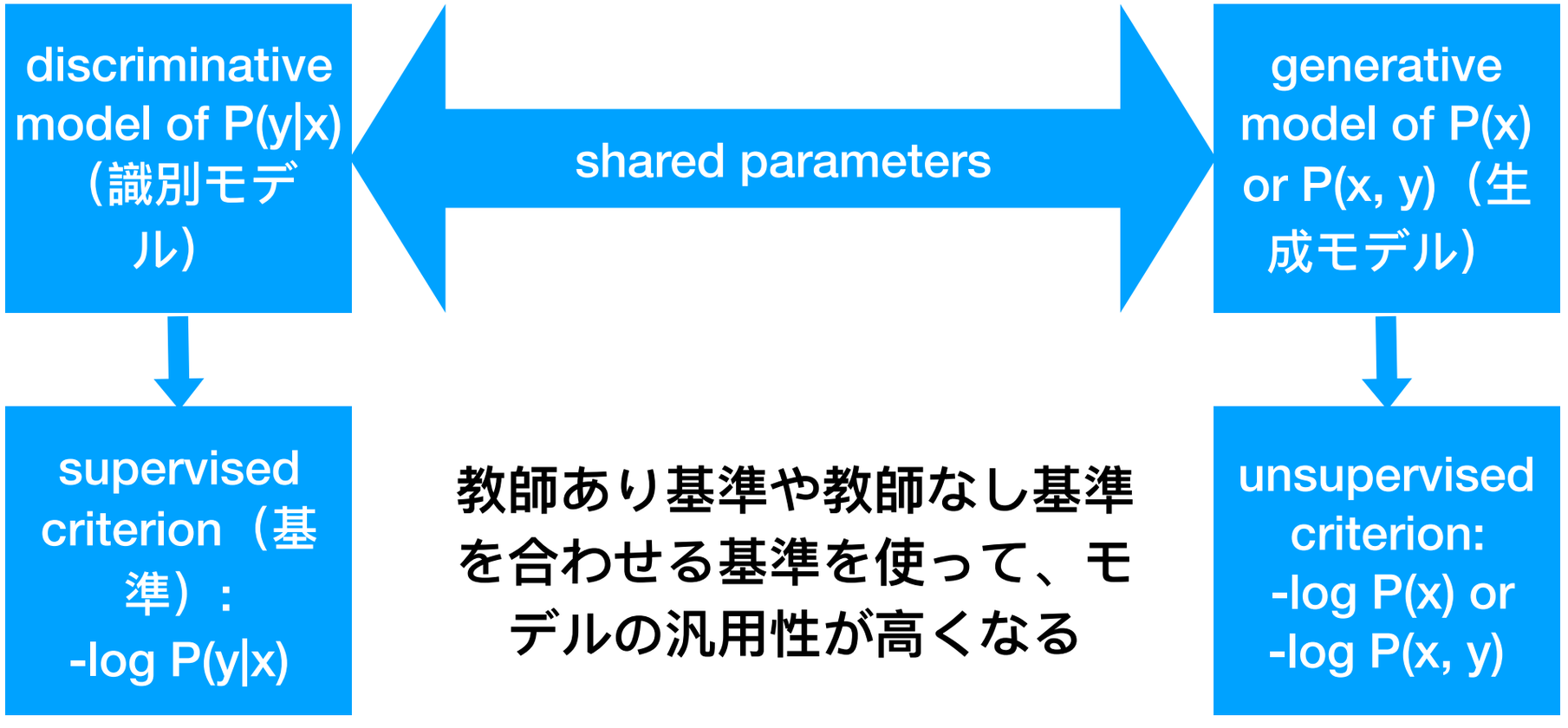
- 考え方：一番良い汎用化方法は、もっと多いデータでトレーニングすること

- apply transformation：画像を何ピクセルで移動したり、回転したり、スケーリングしたりする

- inject noise：ランダムノイズをinputとか、hidden unitsとか（Dropout、後で紹介する）に入れる

-

Semi-Supervised Learning（半教師あり学習）

- 考え方：P(x)（unlabeled examples）がP(y|x)をヘルプする
- 半教師あり学習の目標：learn a representation h=f(x) so that examples from the same class have similar representations.
- 

Multitask Learning

- 考え方：マルチタスクからのサンプルで学習する。マルチタスクのサンプルは、学習機がマルチタスクで汎用できるようにpushする。

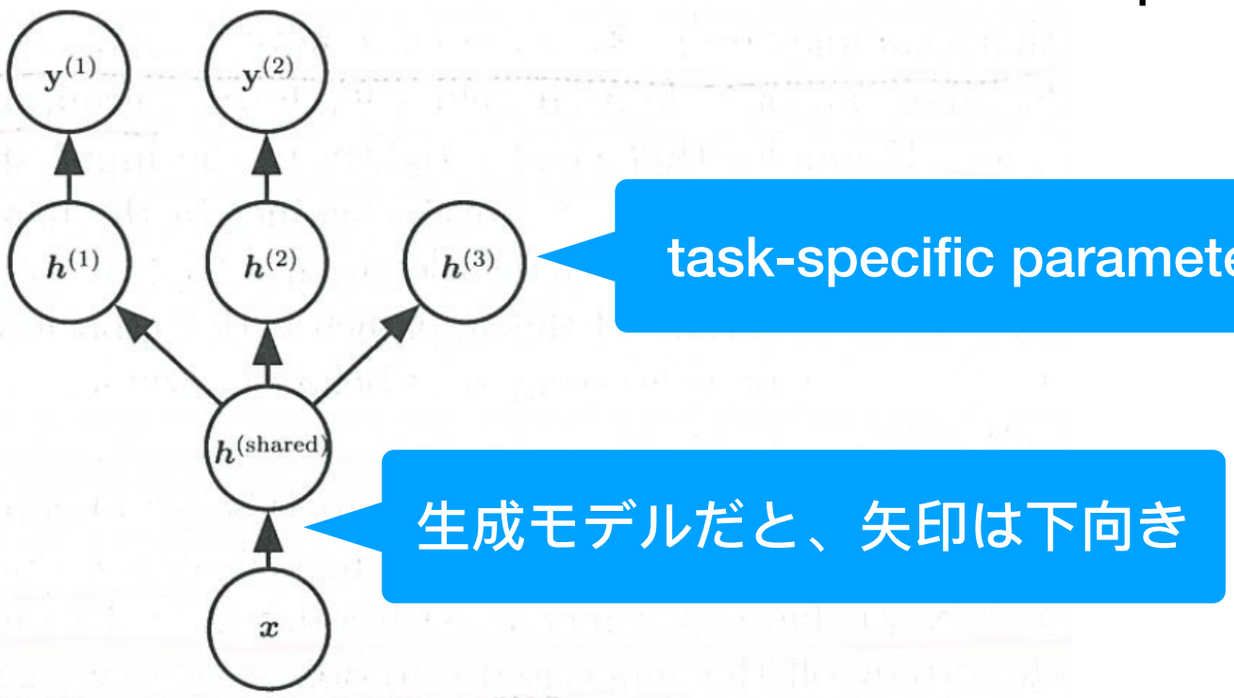


Figure 7.2: Multitask learning can be cast in several ways in deep learning frameworks, and this figure illustrates the common situation where the tasks share a common input but involve different target random variables. The lower layers of a deep network (whether it is supervised and feedforward or includes a generative component with downward arrows) can be shared across such tasks, while task-specific parameters (associated respectively with the weights into and from $\boldsymbol{h}^{(1)}$ and $\boldsymbol{h}^{(2)}$) can be learned on top of those yielding a shared representation $\boldsymbol{h}^{(\text{shared})}$. The underlying assumption is that there exists a common pool of factors that explain the variations in the input $\mathbf{x}$, while each task is associated with a subset of these factors. In this example, it is additionally assumed that top-level hidden units $\boldsymbol{h}^{(1)}$ and $\boldsymbol{h}^{(2)}$ are specialized to each task (respectively predicting $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(2)}$), while some intermediate-level representation $\boldsymbol{h}^{(\text{shared})}$ is shared across all tasks. In the unsupervised learning context, it makes sense for some of the top-level factors to be associated with none of the output tasks ($\boldsymbol{h}^{(3)}$): these are the factors that explain some of the input variations but are not relevant for predicting $\mathbf{y}^{(1)}$ or $\mathbf{y}^{(2)}$.

-

Early Stopping①背景や考え方

Validation set errorが一番小さい時のパラメータが欲しい。Validation set errorが下がる度、パラメータを保存する。

一定のiteration回数内Validationエラーが、最優記録より下がってなかったら、学習が終了

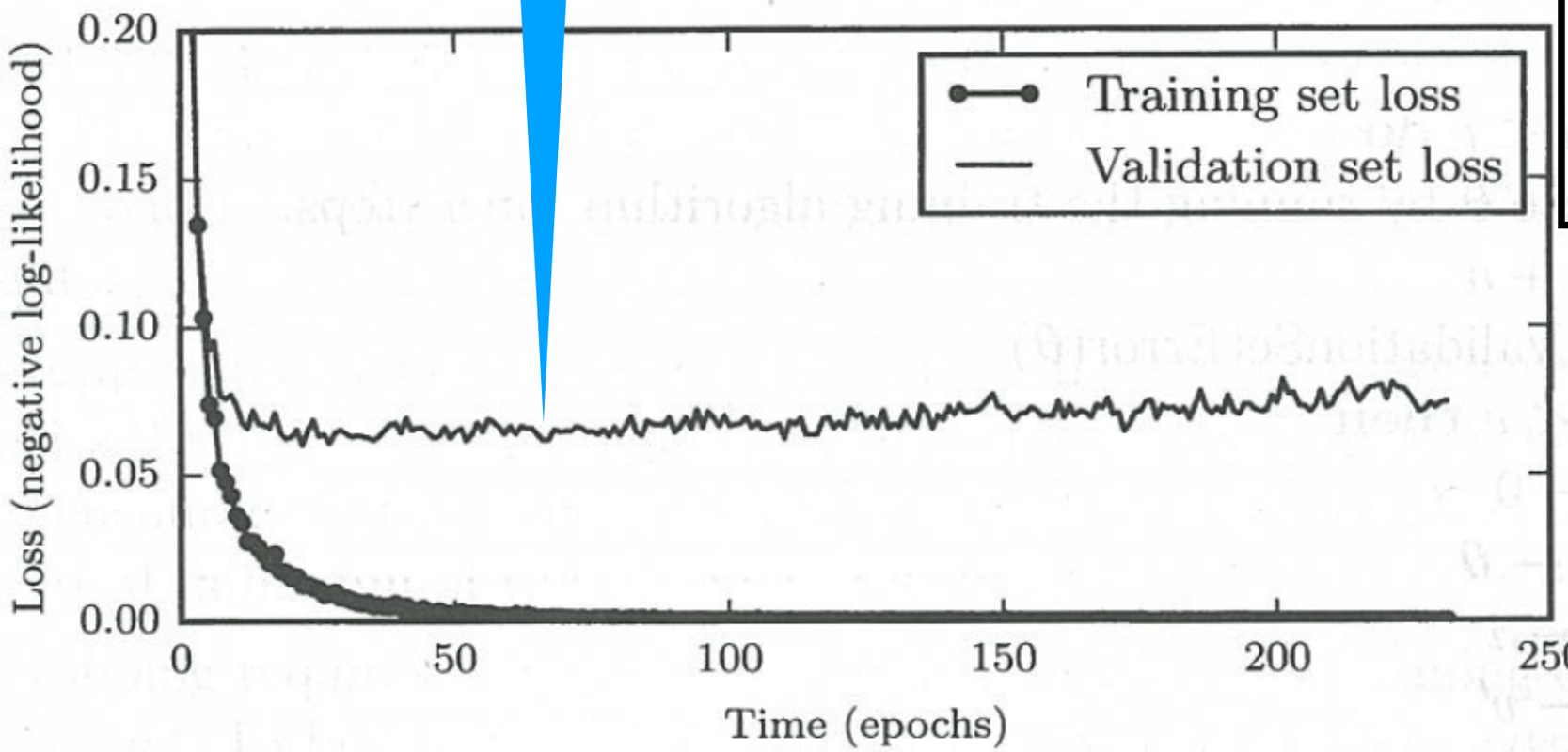


Figure 7.3: Learning curves showing how the negative log-likelihood loss changes over time (indicated as number of training iterations over the dataset, or **epochs**). In this example, we train a maxout network on MNIST. Observe that the training objective decreases consistently over time, but the validation set average loss eventually begins to increase again, forming an asymmetric U-shaped curve.

Early Stopping②なぜearly stoppingがregularizerになれるか

iteration毎のgradient descent再帰式:

$$\omega^{(\tau)} = \omega^{(\tau-1)} - \epsilon \nabla_{\omega} \hat{J}(\omega^{(\tau-1)})$$

Hessian matrixを入れた再帰式:

$$\omega^{(\tau)} - \omega* = (I - \epsilon H)(\omega^{(\tau-1)} - \omega^*)$$

分解したHを入れた再帰式:

$$Q^T(\omega^{(\tau)} - \omega^*) = (I - \epsilon\Lambda)Q^T(\omega^{(\tau-1)} - \omega^*)$$

再帰式を展開する:

$$Q^T\omega^{(\tau)} = [I - (I - \epsilon\Lambda)^{\tau}]Q^T\omega*$$

L2 weight decayと等価の条件:

$$(I - \epsilon\Lambda)^{\tau} = (\Lambda + \alpha I)^{-1}\alpha$$

全ての固有値が小さければ:

$$\tau \approx \frac{1}{\epsilon\alpha}$$

early stoppingのパラメータ（training iteration回数）の効果は、L2 weight decayのパラメータ（α）の効果と逆。

何でこんな形になるかまだわからない。しかし、L1 weight decayと似ている。

weight decayと比べて、early stoppingのメリットは、early stoppingがvalidation set errorを見ているので、自動的にregularizationの程度を決められる。しかし、weight decayはαを決めるために、たくさんトレーニングが必要。

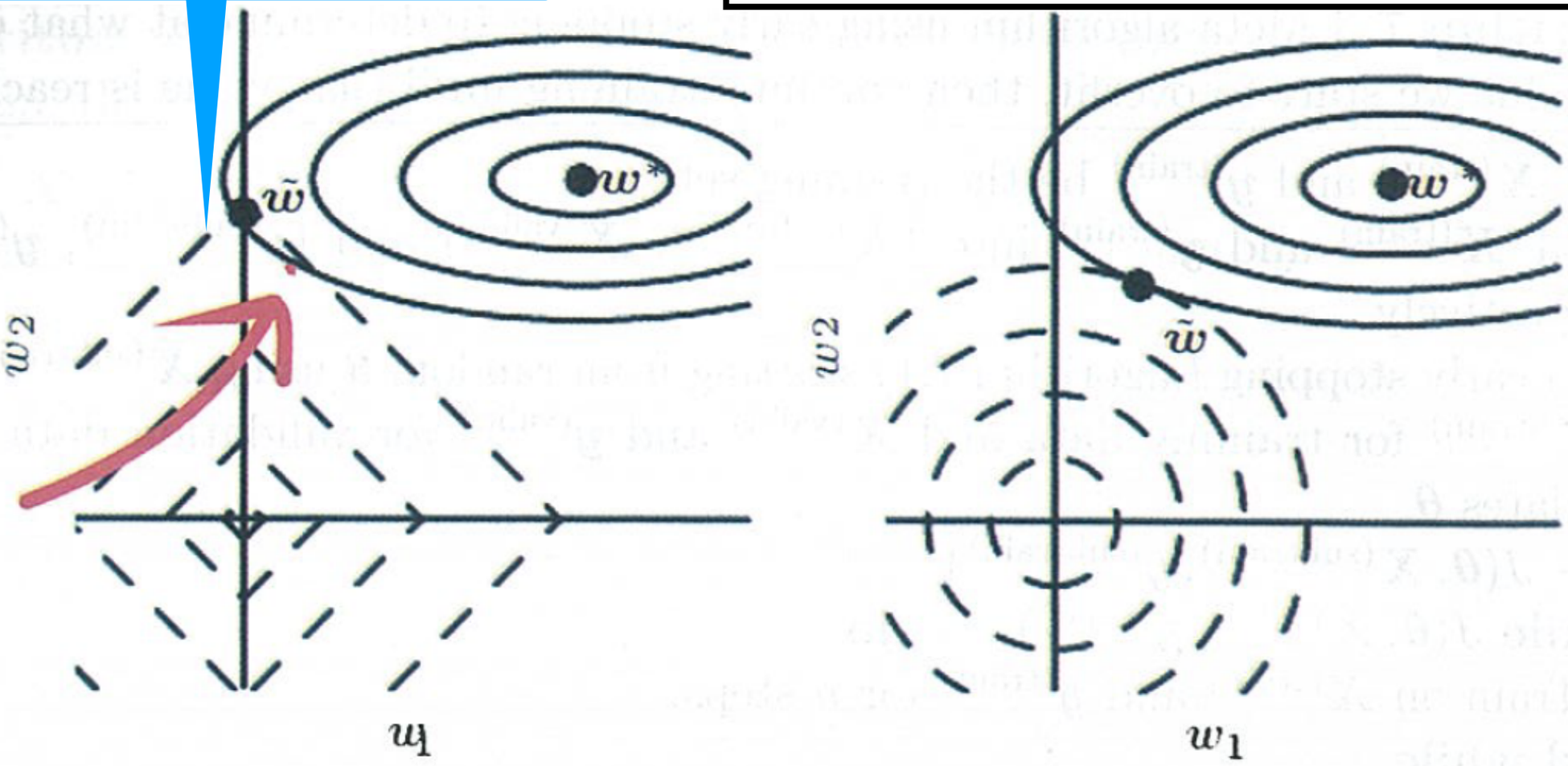


Figure 7.4: An illustration of the effect of early stopping. *(Left)*The solid contour lines indicate the contours of the negative log-likelihood. The dashed line indicates the trajectory taken by SGD beginning from the origin. Rather than stopping at the point $\boldsymbol{w}^*$ that minimizes the cost, early stopping results in the trajectory stopping at an earlier point $\tilde{\boldsymbol{w}}$. *(Right)*An illustration of the effect of L^2 regularization for comparison. The dashed circles indicate the contours of the L^2 penalty, which causes the minimum of the total cost to lie nearer the origin than the minimum of the unregularized cost.

p245

Parameter Tying and Parameter Sharing

- 考え方：モデルパラメータ間に依存（dependency）があるはず
- Parameter sharingの最も有名な例はCNN（convolutional neural networks、3回目勉強会紹介する）
- 画像にたくさん移転に不変なプロパーティーがある。The same feature (a hidden unit with the same weights) is computed over different locations in the input.
-

Sparse Representations

$y \in \mathbb{R}^m$ $A \in \mathbb{R}^{m \times n}$ $x \in \mathbb{R}^n$

$$\begin{bmatrix} 18 \\ 5 \\ 15 \\ -9 \\ -3 \end{bmatrix} = \begin{bmatrix} 4 & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & -1 & 0 & 3 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & -4 \\ 1 & 0 & 0 & 0 & -5 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ -2 \\ -5 \\ 1 \\ 4 \end{bmatrix}$$

Sparse Parametrization、モデルパラメータへの罰。（前に紹介した重みへのL1 penalization）

$y \in \mathbb{R}^m$ $B \in \mathbb{R}^{m \times n}$ $h \in \mathbb{R}^n$

$$\begin{bmatrix} -14 \\ 1 \\ 19 \\ 2 \\ 23 \end{bmatrix} = \begin{bmatrix} 3 & -1 & 2 & -5 & 4 & 1 \\ 4 & 2 & -3 & -1 & 1 & 3 \\ -1 & 5 & 4 & 2 & -3 & -2 \\ 3 & 1 & 2 & -3 & 0 & -3 \\ -5 & 4 & -2 & 2 & -5 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 0 \\ 0 \\ -3 \\ 0 \end{bmatrix}$$

Sparse Representation、活性化（activation）への罰。

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

hは、まばらにxを表現する関数。
h is a function of x that represents the information present in x, but does so with a sparse vector.

Bagging and Other Ensemble Methods

- 考え方：複数のモデルを別々にトレーニングして、テストする時、モデルらのアウトプットを合わせる。

$$\mathbb{E}[\epsilon_i^2] = v$$

$$\mathbb{E}[\epsilon_i \epsilon_j] = c$$

エラー分布のvarianceとcovariance

kモデルの平均エラー予測器

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

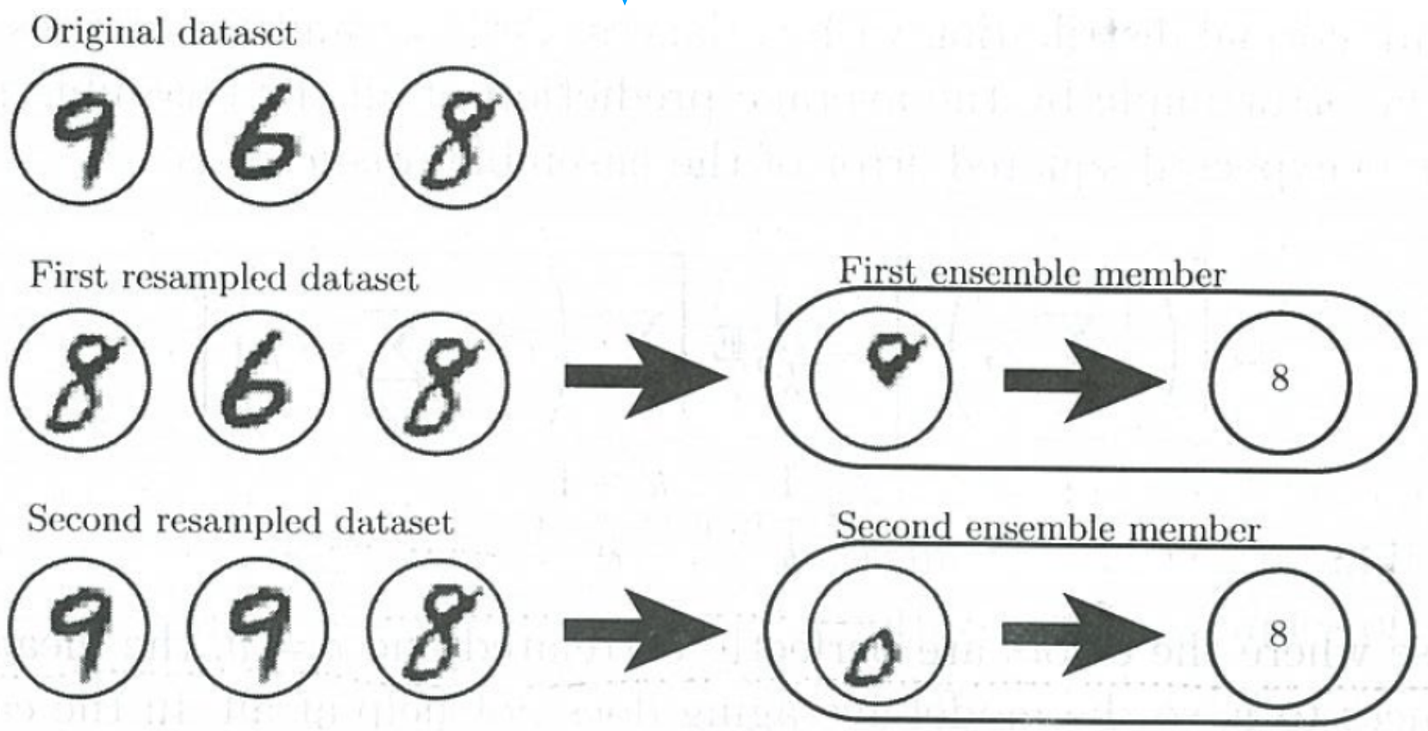
$$= \frac{1}{k}v + \frac{k-1}{k}c$$

最高：エラーが完全に無相関だと、c=0。二乗誤差がv/kになる。
最低：エラーが完全に相関だと、c=v。二乗誤差がvのまま。

-

８の検出器をトレーニングしている。

別々のモデルが弱いが、合わせると、丈夫になる。

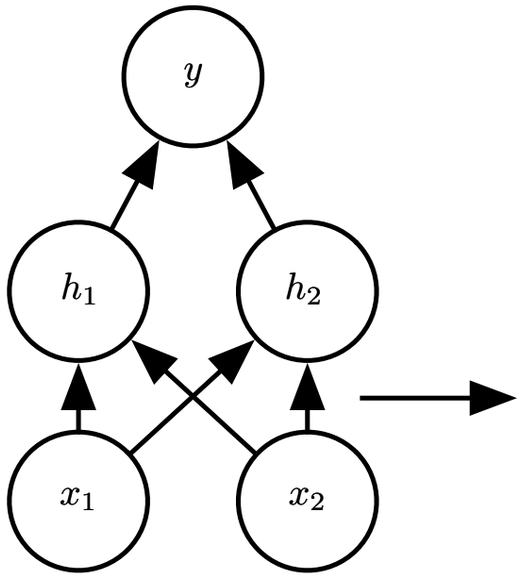


一つ目のモデルは、８の上半分ありの画像を８と判定する。

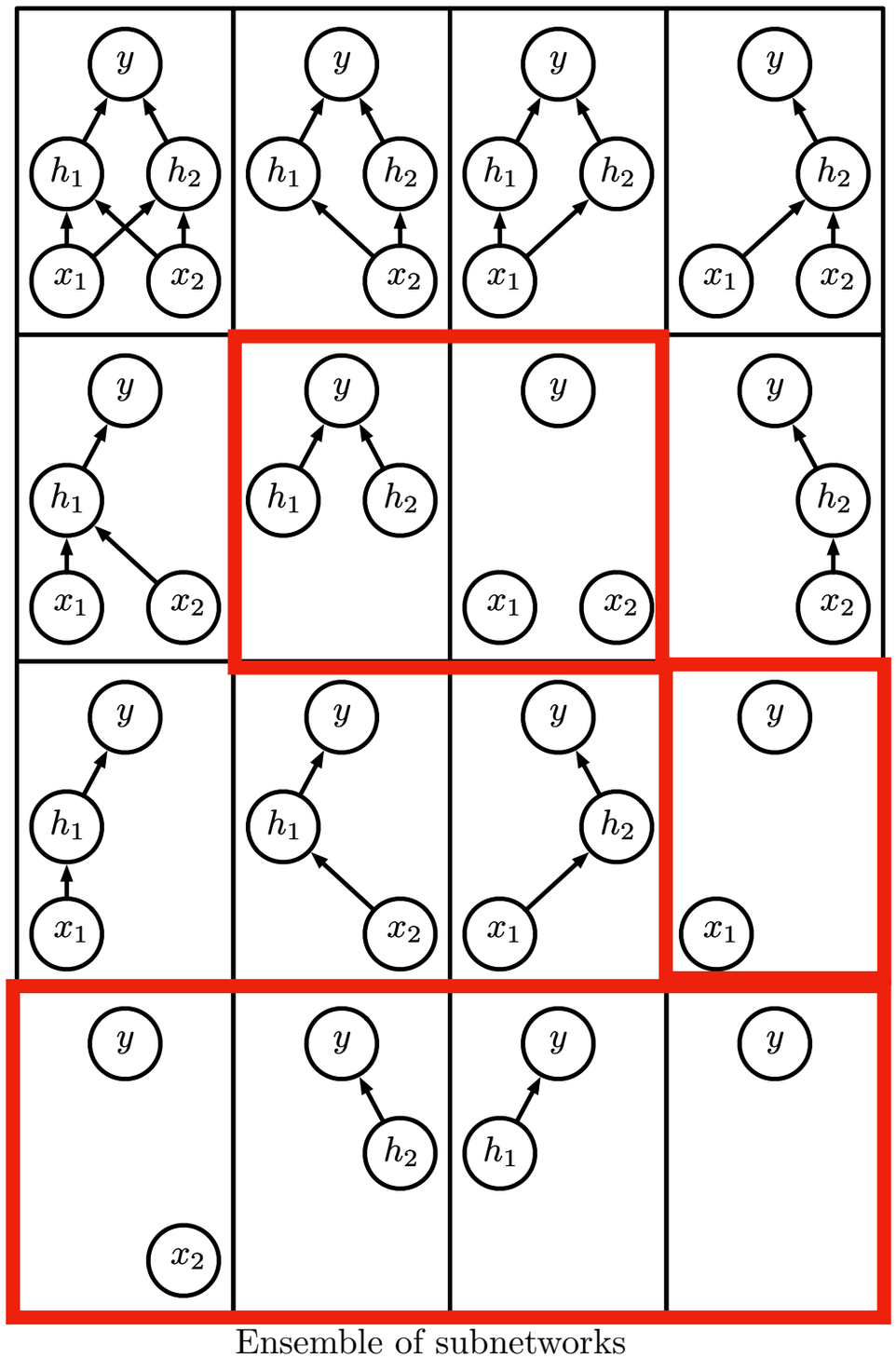
二つ目のモデルは、８の下半分ありの画像を８と判定する。

Figure 7.5: A cartoon depiction of how bagging works. Suppose we train an 8 detector on the dataset depicted above, containing an 8, a 6 and a 9. Suppose we make two different resampled datasets. The bagging training procedure is to construct each of these datasets by sampling with replacement. The first dataset omits the 9 and repeats the 8. On this dataset, the detector learns that a loop on top of the digit corresponds to an 8. On the second dataset, we repeat the 9 and omit the 6. In this case, the detector learns that a loop on the bottom of the digit corresponds to an 8. Each of these individual classification rules is brittle, but if we average their output, then the detector is robust, achieving maximal confidence only when both loops of the 8 are present.

Dropout①baggingしたい



Base network



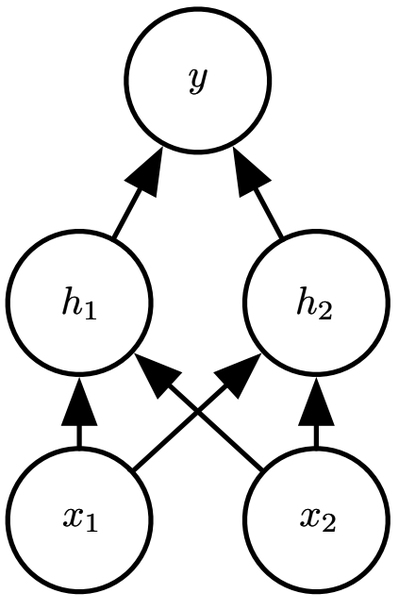
Ensemble of subnetworks

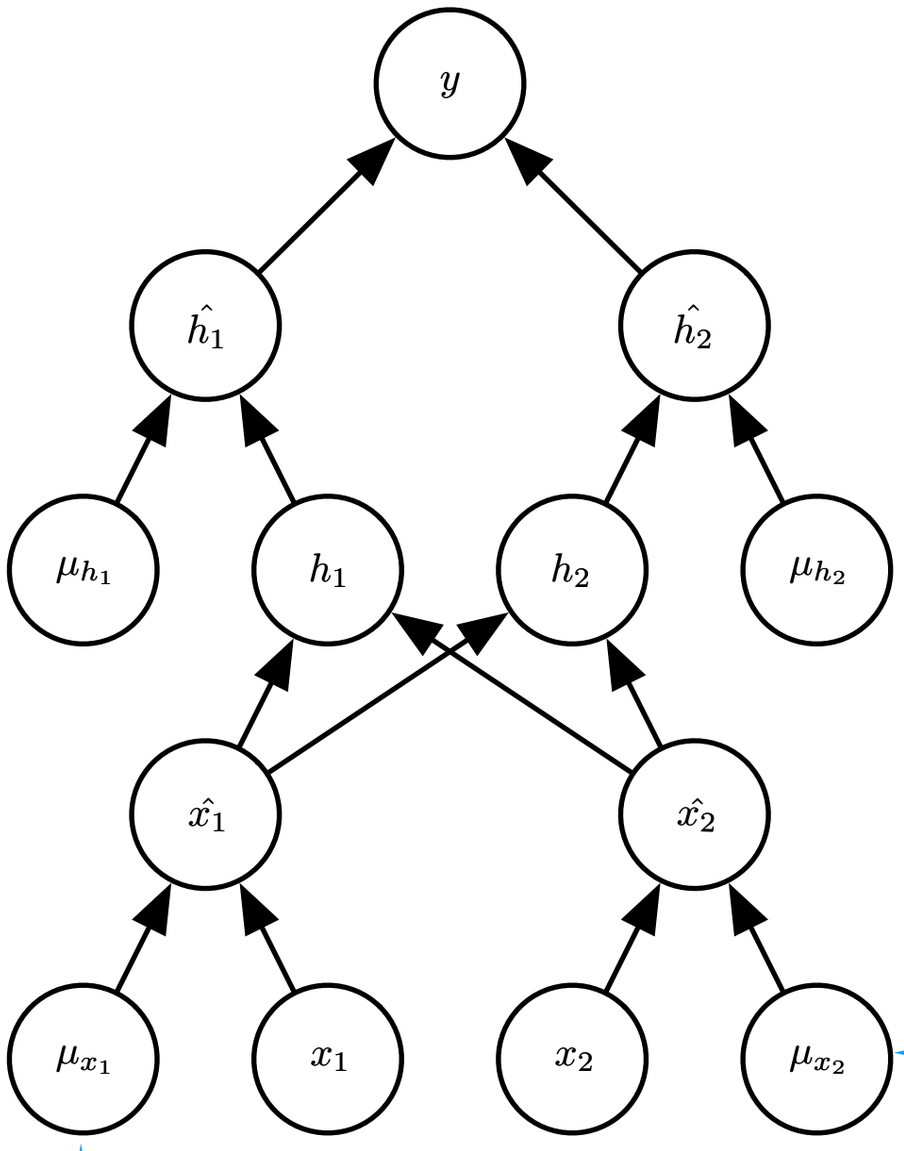
①Dropoutは、大きいneural networkでもbagging方法が現実的にできる方法。
②Dropoutがnonoutput units（y以外）を除く（０を掛ける）subnetworksをトレーニングする。
③baggingと大きい違う所：baggingのモデルらが独立。dropoutのsubnetworkらはパラメータ共有している。

Typically, an input unit is included with probability 0.8, and a hidden unit is included with probability 0.5.

inputがなかったり、pathがなかったり、赤枠のsubnetworkがいらない。広い層のnetworkでは、全部のinputもしくは全部のpathが除く可能性が低いので、問題にならない。

Dropout②Training中maskを掛ける





①step毎に、uをサンプリングして（前ページのsubnetworksから一つを選ぶと等価）、一つのsubnetworkをトレーニングする。
②勿論全部のsubnetworksまでサンプリングできないので、一部のsubnetworksしかがトレーニングされていない。
③だが、parameter sharingのため、残るsubnetworksにとっても良いパラメータ筈だ。

Typically, an input unit is included with probability 0.8, and a hidden unit is included with probability 0.5.

p(u=1)は hyperparameter

1 or 0

Dropout③Training後networkでどうやって予測する（p(y|x)の計算）

arithmetic mean（算術平均）：

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

uの可能性が無限なので、この式で計算できない

dは除けるunit数

geometric mean（幾何平均）：

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

renormalize（再標準化）：

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

weight scaling inference ruleで予測できる。
①モデルの全てのunitsがある
②unit iから出た重みは、unit iの含まれる確率を掛けられる
例えば、もし全てのunitの含まれる確率が0.5だったら、training完了後、モデルの重みに0.5を掛ければ、この後通常で使える。

説明①：linearモデルにとっては、weight scale ruleが正確。しかし、non-linearityがあるモデルにとっては、weight scale ruleが近似値しか言えない。
説明②：だが、経験的に、weight scale ruleが良くしている。

p255-256

Dropout④weight scaling ruleの証明（softmaxを例として）

geometric mean（幾何平均）：

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

dはmask。
p(d=1)=p(d=0)
=1/2

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

$$= \frac{\sqrt[2^d]{\prod_{d \in \{0,1\}^n} exp(W^T_{y,:}(d \odot v) + b)_y}}{\sqrt[2^d]{\prod_{d \in \{0,1\}^n} \Sigma_{y'} exp(W^T_{y',:}(d \odot v) + b)_{y'}}}$$

標準化があるから、yにとって定数である分母を気にしなくて良い

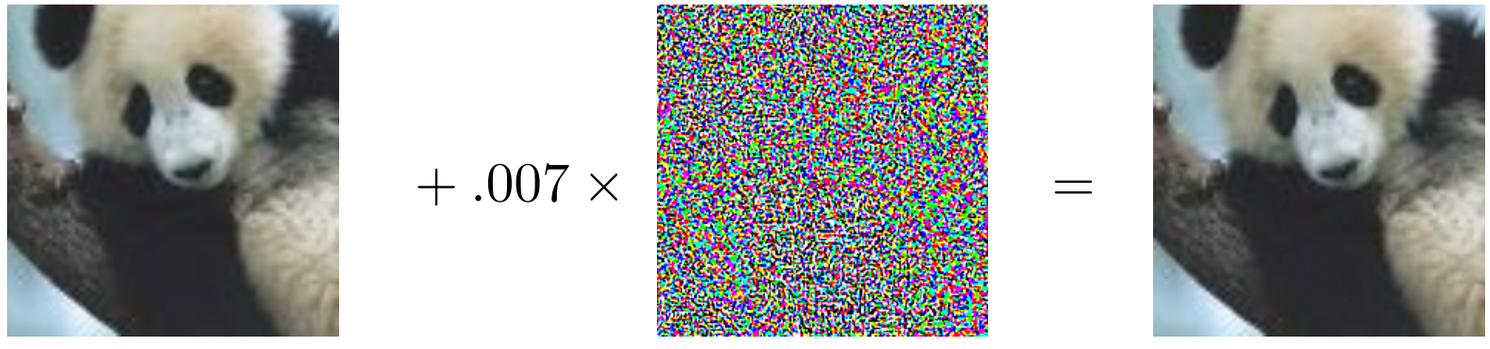
2^nの元の総和。
d*p(d=1)=1/2.

$$\tilde{p}_{ensemble}(y \mid v) \propto \sqrt[2^d]{\prod_{d \in \{0,1\}^n} exp(W^T_{y,:}(d \odot v) + b)_y}$$

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

1/2Wのsoftmaxになった。

Adversarial（敵対） Training：背景や原因、対策



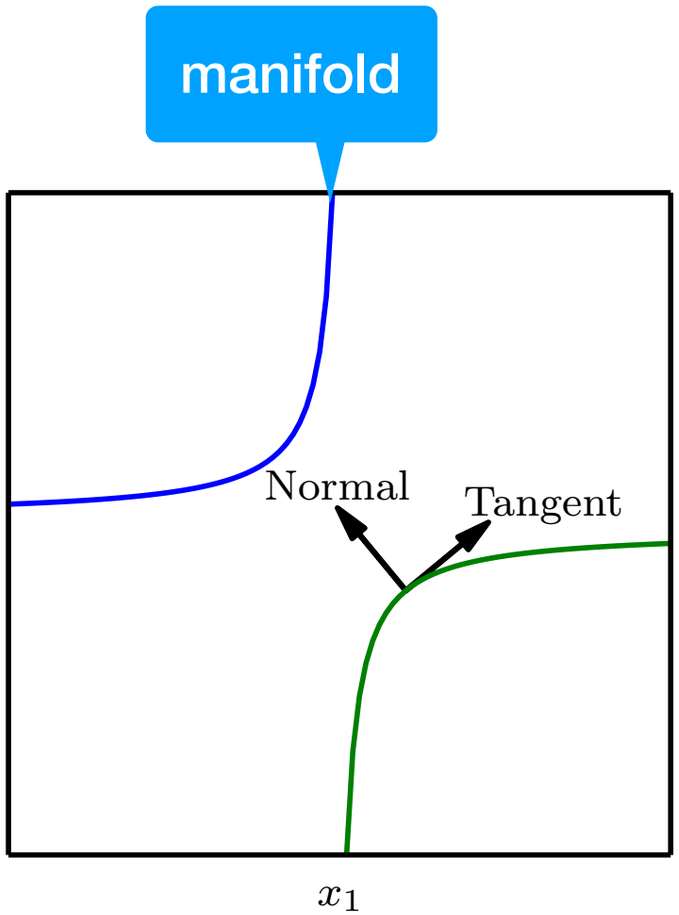
$\boldsymbol{x}$	$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$	$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
$y =$“panda”	“nematode”	“gibbon”
w/ 57.7%	w/ 8.2%	w/ 99.3 %
confidence	confidence	confidence

data point xに気づかれないほどの小さいベクトルを入れても、分類器の分類結果が結構変わる。

原因：excessive linearity（過大線型性）。（インプットが小さく変わっても、アウトプットが大きく変わる）
対策：敵対学習はモデルを学習データの近くのlocal constancyに働きかける。

因みに、inputの小さい変わりによってoutputがどのくらい変わるかを評価するパラメータが、条件数（condition number）という。excessive linearityだと、条件数が悪い。

Tangent（接線の） Prop and Manifold Tangent Classifier



①Tangent PropagationとManifold Tangent Classifierは、f(x)がTangent方向でほぼ変わらない、Normal方向で早く変わるように正則化している。
②Manifold Tangent Classifierは、autoencoderをトレーニングして、manifold tangent方向を予測する。（Autoencoderは、Chapter 14で紹介する）

本当のmanifoldは、頭の2種類回転の角度です。two dimensional manifold corresponding to two angles of rotation.