Mathematics of AI

Regularisation for deep learning

Bishop, *PRML* & Goodfellow *et al*, *Deep Learning*

Batch Normalization

Ian Goodfellow

Deep Learning Study Group
San Francisco
September 12, 2016

Batch Normalization

$$\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{W}$$

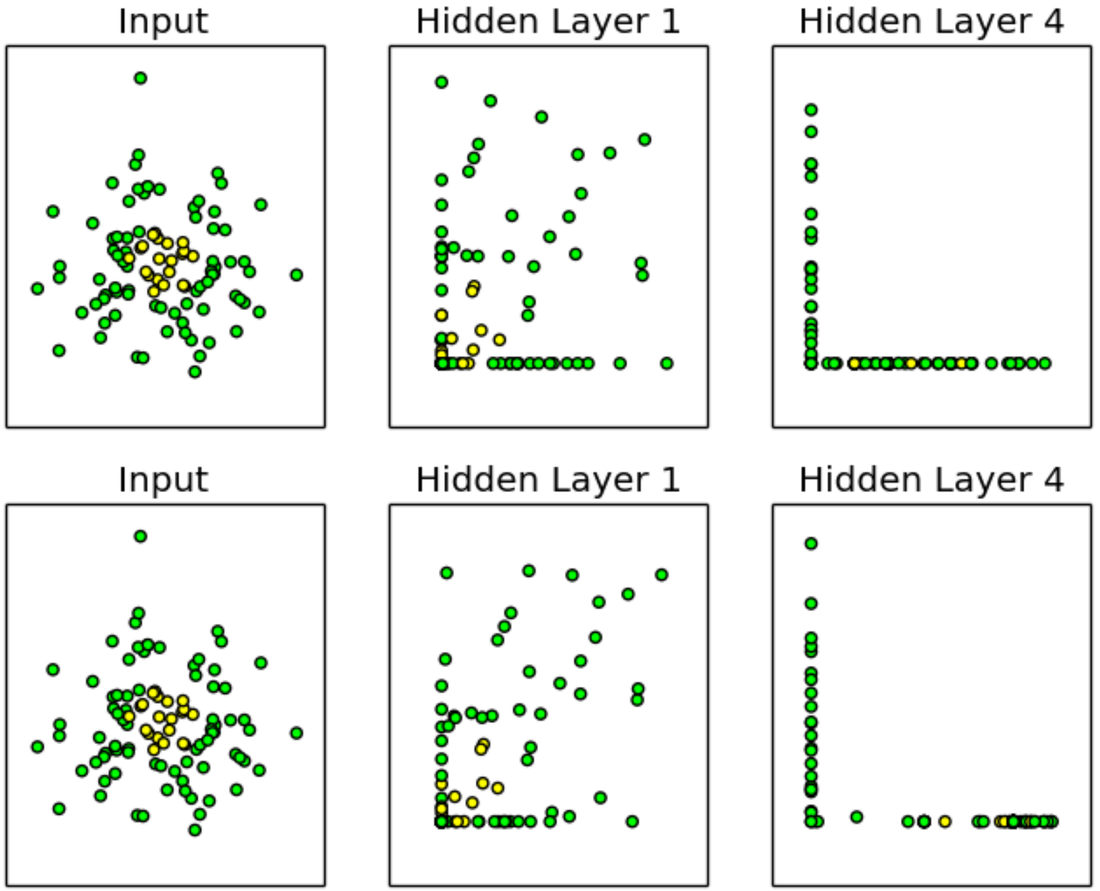
$$\tilde{\boldsymbol{Z}} = \boldsymbol{Z} - \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{Z}_{i,:}$$

$$\hat{\boldsymbol{Z}} = \frac{\tilde{\boldsymbol{Z}}}{\sqrt{\epsilon + \frac{1}{m}\sum_{i=1}^{m} \tilde{\boldsymbol{Z}}_{i,:}^{2}}}$$

$$\boldsymbol{H} = \max\{0, \boldsymbol{\gamma}\hat{Z} + \boldsymbol{\beta}\}$$

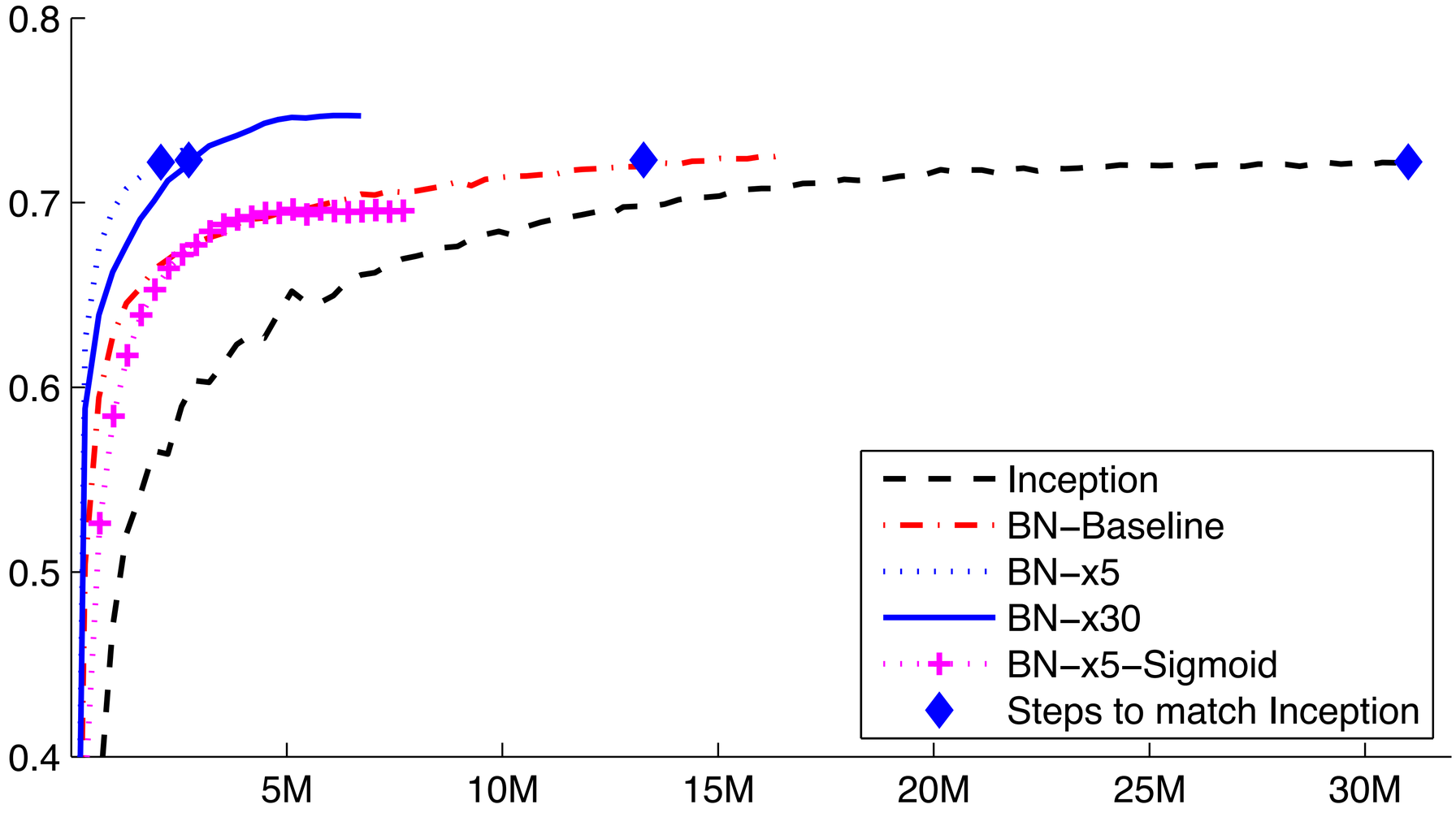
"Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," Ioffe and Szegedy 2015

Before SGD step



After SGD step

"Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," Ioffe and Szegedy 2015

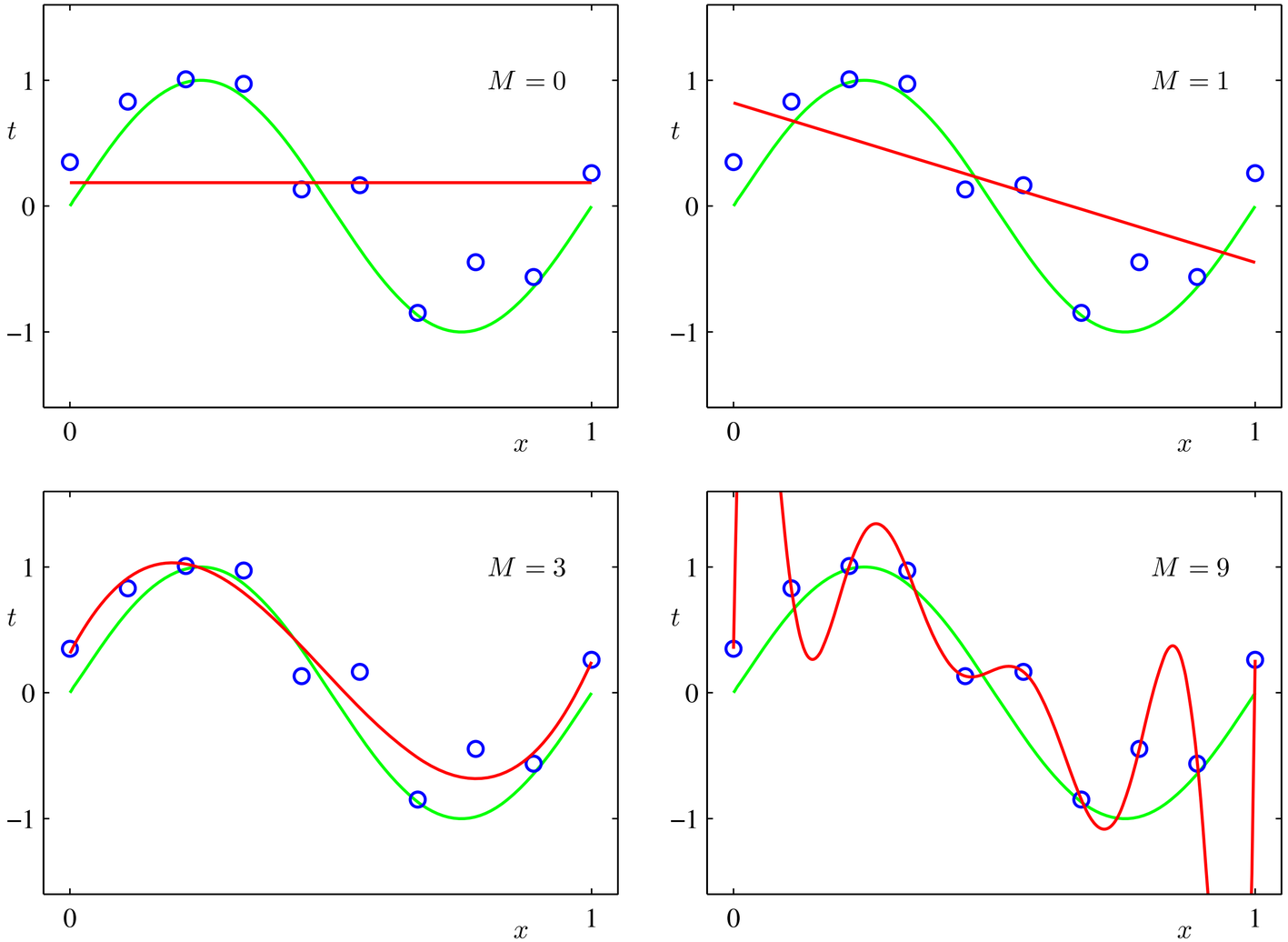


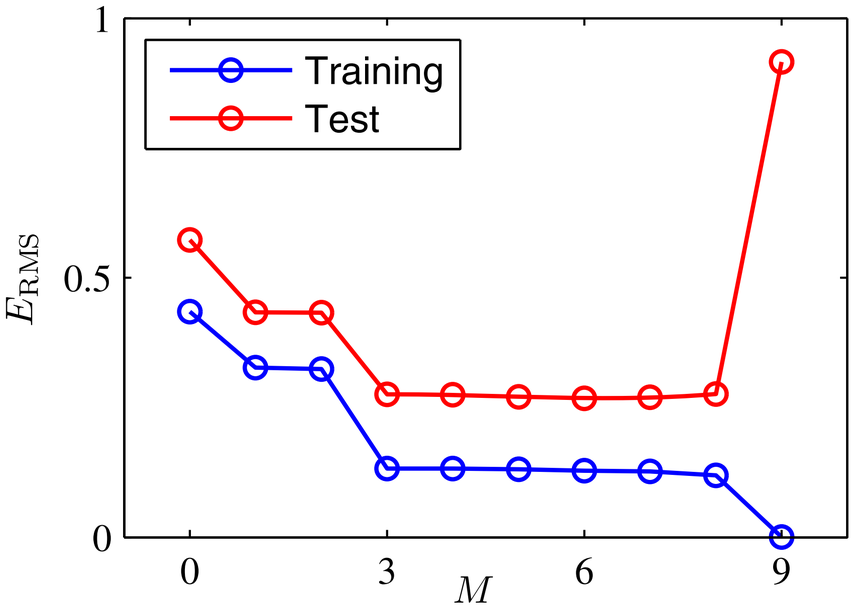
"Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," Ioffe and Szegedy 2015

Regularisation

Overfitting

When too many parameters are barely enough

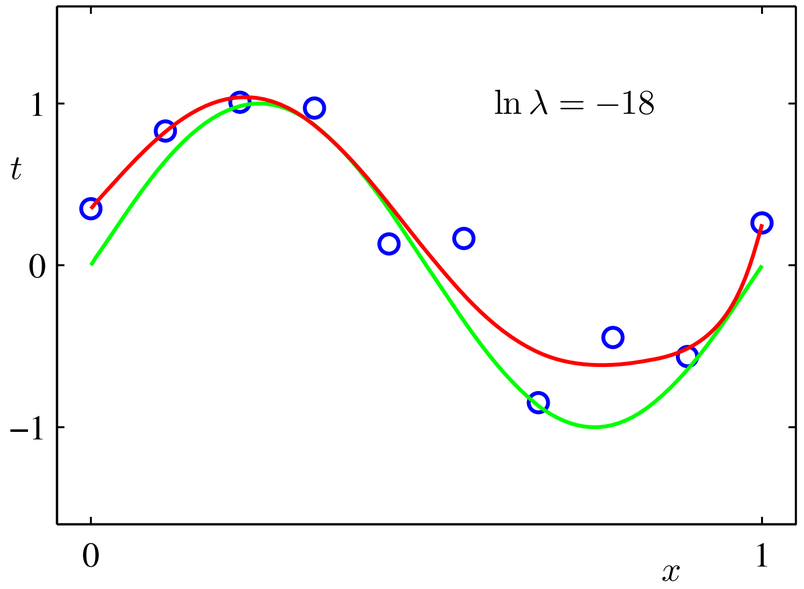


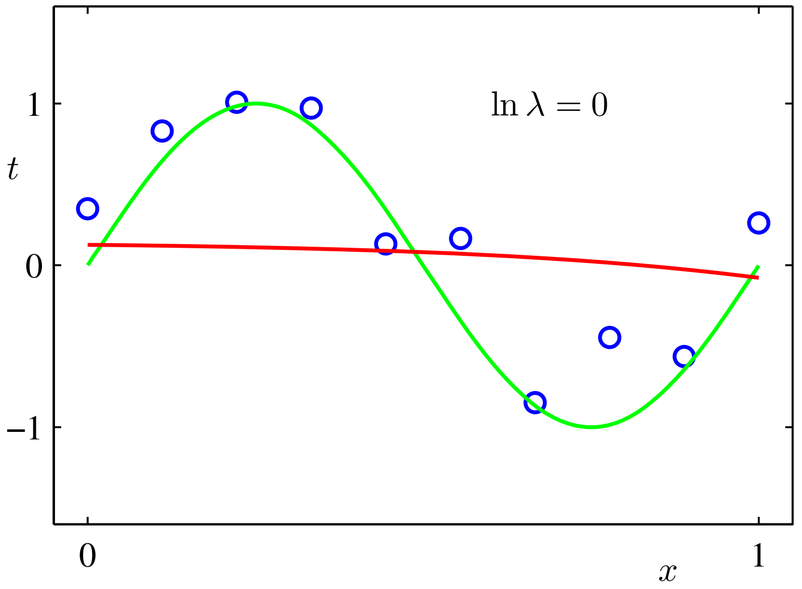


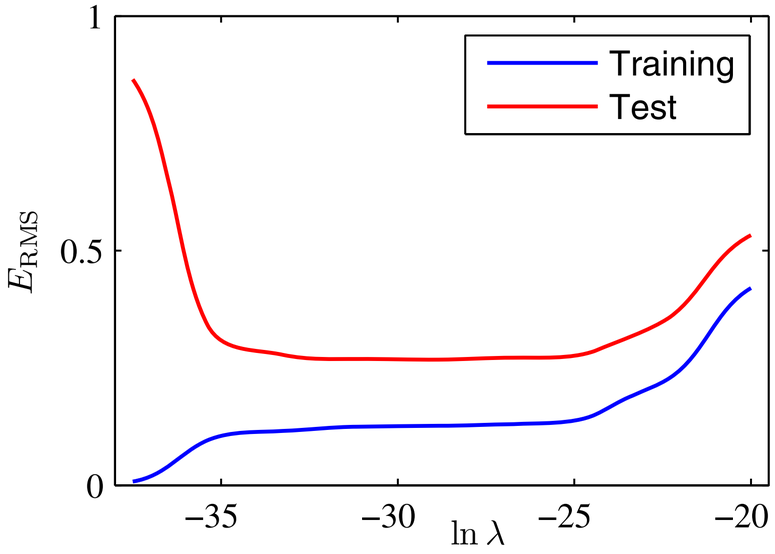
Regularisation

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

(ridge regression)







Regularisation

In general

L1 (LASSO):

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

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

(note bias-variance)

L2 (ridge):

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

Regularisation

In general

L1 (LASSO): $$\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \frac{\alpha}{2}\boldsymbol{w}^\top \boldsymbol{w} + J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}),$$ $$w_i = \operatorname{sign}(w_i^*) \max\left\{|w_i^*| - \frac{\alpha}{H_{i,i}}, 0\right\}.$$

(note bias-variance)

L2 (ridge): $$\tilde{J}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}) = \frac{\alpha}{2}\boldsymbol{w}^\top \boldsymbol{w} + J(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{y}),$$

This is constrained optimisation! (i.e., KKT)

$$\min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda}} \max_{\boldsymbol{\alpha}, \boldsymbol{\alpha} \geq 0} -f(\boldsymbol{x}) + \sum_i \lambda_i g^{(i)}(\boldsymbol{x}) + \sum_j \alpha_j h^{(j)}(\boldsymbol{x}).$$

Weight Decay as Constrained Optimization

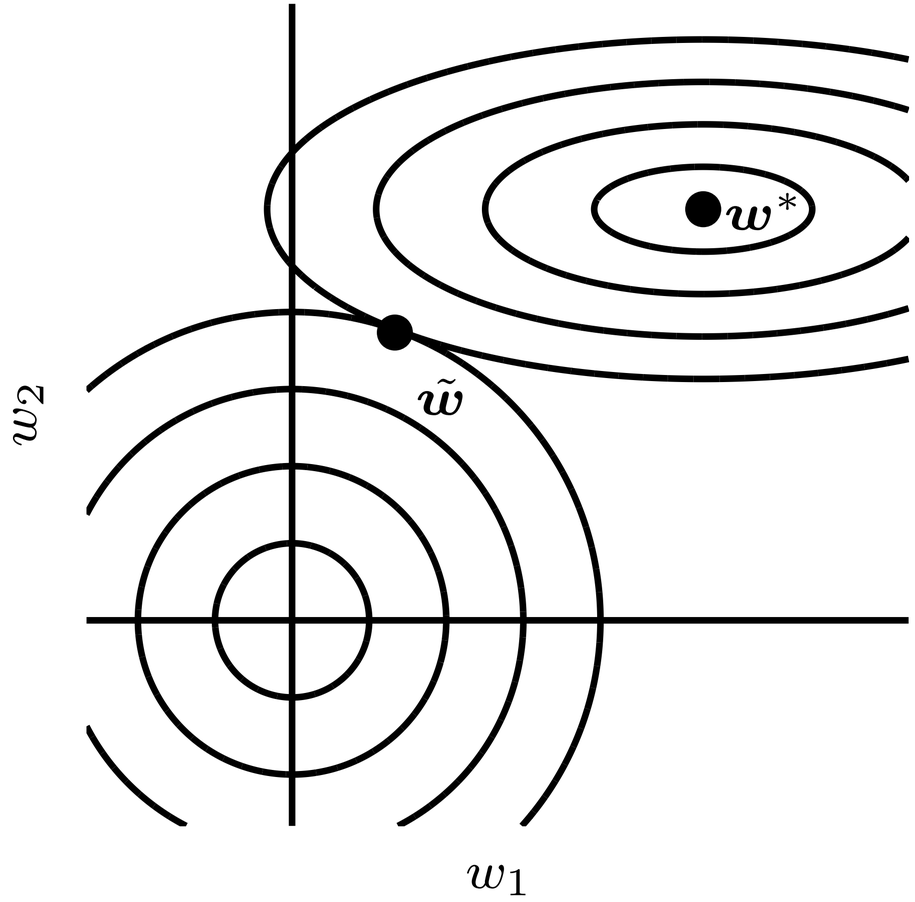


Figure 7.1

the problem is not convex (but some other regularity conditions hold), then these KKT conditions still characterise a local minima.

Theorem 4.12 (KKT: Slater's condition version)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex, continuously differentiable function on the set $\Omega = \{\boldsymbol{x} \in \mathbb{R}^n : g_i(\boldsymbol{x}) \leq 0, \text{ for all } i = 1, \dots, m\}$, where the $g_i : \mathbb{R}^n \to \mathbb{R}$ are convex and continuously differentiable for all $i = 1, \dots, m$, such that $\operatorname{int} \Omega \neq \emptyset$. A necessary and sufficient condition that $\boldsymbol{x}^* \in \Omega$ minimises f over Ω is that there exists a solution to the system*

$$\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(\boldsymbol{x}^*) = \boldsymbol{0}, \tag{4.7a}$$

$$\lambda_i g_i(\boldsymbol{x}^*) = 0 \text{ for all } i = 1, 2, \dots, m, \tag{4.7b}$$

$$\boldsymbol{g}(\boldsymbol{x}^*) \leq \boldsymbol{0}, \tag{4.7c}$$

$$\boldsymbol{\lambda} \geq \boldsymbol{0}. \tag{4.7d}$$

Section 4.2.1 proves this theorem.

The condition (4.7b) is sometimes called a complementary slackness condition because it expresses that either the constraint is active (and the Lagrange multiplier is potentially positive) or the Lagrange multiplier is zero (and potentially the constraint is not active).

The conditions (4.7) are called the **KKT conditions**. Conditions (4.7a) and (4.7b) may be equivalently replaced with

$$\nabla f(\boldsymbol{x}^*) + \sum_{i \in I(\boldsymbol{x}^*)} \lambda_i \nabla g_i(\boldsymbol{x}^*) = \boldsymbol{0}, \tag{4.8}$$

and for all $i \notin I(\boldsymbol{x}^*)$ we set $\lambda_i = 0$.

Theorem 4.12 is an extension of the Lagrange multiplier idea. That is, for the active constraints $I(\boldsymbol{x}^*)$[5] we would write a new objective function

$$h(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i \in I(\boldsymbol{x}^*)} \lambda_i g_i(\boldsymbol{x}),$$

including Lagrange multipliers as in (4.4) for the active constraints, and then ∇h is just the left-hand side of condition (4.8). So we are basically solving a problem including Lagrange multipliers to enforce the appropriate set of constraints (the active ones).

The only additional information we get from Theorem 4.12 is that the Lagrange multipliers λ_i must be positive or zero. Figures 4.15 and 4.16 illustrate the positivity requirements: the constraints $I(\boldsymbol{x}^*)$ are active, so the point $\boldsymbol{x}^*$ is in the intersection of the curves $g_i(\boldsymbol{x}) = 0$, for $i \in I(\boldsymbol{x}^*)$. We know that the feasible region is inside the set Ω, that is that $g_i(\boldsymbol{x}) \leq 0$, and also that $g_i(\boldsymbol{x})$ is increasing in the direction $\nabla g_i(\boldsymbol{x})$. Since $g_i(\boldsymbol{x}^*) = 0$, then $\nabla g_i(\boldsymbol{x}^*)$ points outside the set Ω, equivalently $-\nabla g_i(\boldsymbol{x}^*)$ points inside. Likewise, if $\boldsymbol{x}^*$ is a minimum, the objective function must increase in any direction pointing into the region. However, what the theorem says is even stronger: it says that

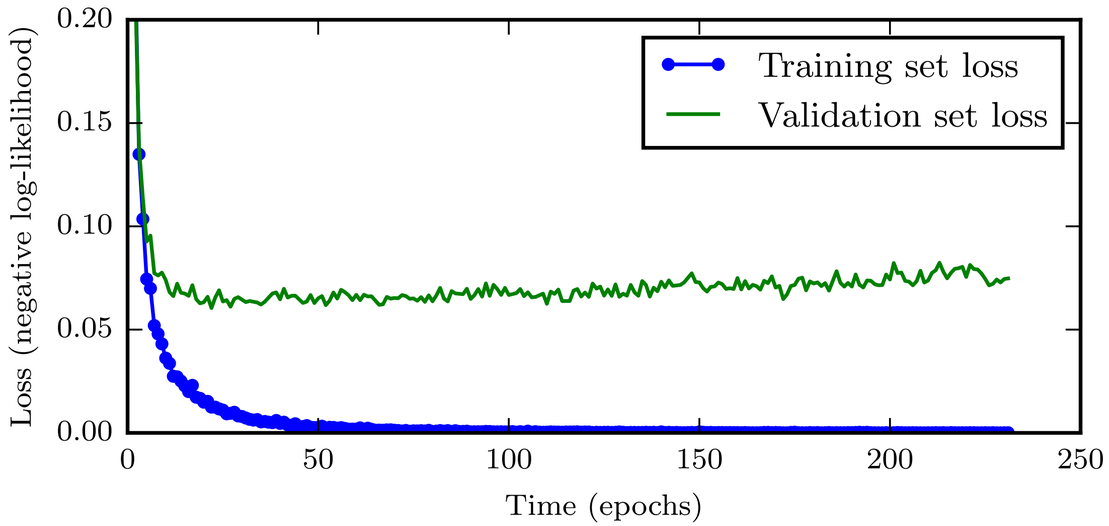
$$\nabla f(\boldsymbol{x}^*) = -\sum_{i \in I(\boldsymbol{x}^*)} \lambda_i \nabla g_i(\boldsymbol{x}^*),$$

[5] If $I(\boldsymbol{x}^*) = \emptyset$ then $\boldsymbol{x}^*$ is not on any boundary, and the KKT condition just reverts to the familiar condition for an unconstrained problem; that is, $\nabla f(\boldsymbol{x}^*) = 0$.

Regularisation

In deep learning

- “Just get more data” —Guy from Google
- Early stopping, i.e., “cheating”
- … (bagging, multi-task learning, …)
- Sparse representations! (cf. compressive sensing…)
- Dropout



Dropout

Figure 7.6