Parameter Norm Penalties

Regularization Strategies

1. Parameter Norm Penalties
2. Norm Penalties as Constrained Optimization
3. Regularization and Under-constrained Problems
4. Data Set Augmentation
5. Noise Robustness
6. Semi-supervised learning
7. Multi-task learning

8. Early Stopping
6. Parameter tying and parameter sharing
7. Sparse representations
8. Bagging and other ensemble methods
9. Dropout
10. Adversarial training
11. Tangent methods

Topics in Parameter Norm Penalties

1. Overview (limiting model capacity)
2. L^2 parameter regularization
3. L^1 regularization

Limiting Model Capacity

- Regularization has been used for decades prior to advent of deep learning
- Linear- and logistic-regression allow simple, straightforward and effective regularization strategies
  - Adding a parameter norm penalty $\Omega(\theta)$ to the objective function J:

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

    - where $\alpha \varepsilon [0, \theta)$ is a hyperparameter that weight the relative contribution of the norm penalty term Ω
      - Setting α to 0 results in no regularization. Larger values correspond to more regularization

Norm Penalty

- When our training algorithm minimizes the regularized objective function

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

  – it will decrease both the original objective J on the training data and some measure of the size of the parameters θ

- Different choices of the parameter norm Ω can result in different solutions preferred
  – We discuss effects of various norms

No penalty for biases

- Norm penalty Ω penalizes only weights at each layer and leaves biases unregularized
  - Biases require less data to fit than weights
  - Each weight specifies how variables interact
    - Fitting weights requires observing both variables in a variety of conditions
- Each bias controls only a single variable
  - We do not induce too much variance by leaving biases unregularized
- $\boldsymbol{w}$ indicates all weights affected by norm penalty
- $\boldsymbol{\theta}$ denotes both $\boldsymbol{w}$ and biases

Different or Same αs for layers?

- Sometimes it is desirable to use a separate penalty with a different α for each layer

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

- Invariance under linear transformation $\mathbf{T}$ is one case
  - i.e., we want neural net to perform the same when the inputs are transformed

- But this creates too many hyperparameters
  - Search space reduced by using same hyperparameters

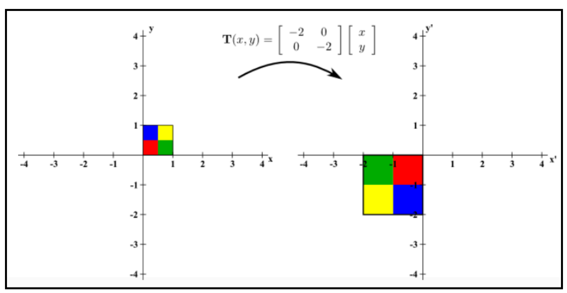
Linear Transformation T

- Consider a simple linear transformation of the input

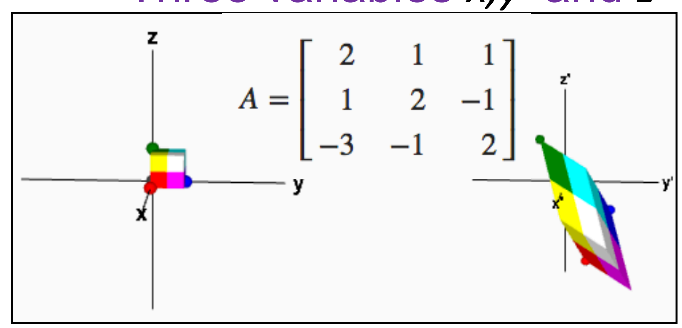
Two-variables
x and y

$$\mathbf{T}(x, y) = (ax + by, cx + dy) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\boxed{\mathbf{T}(\mathbf{x}) = A\mathbf{x}}$$



Three variables *x,y* and *z*



Weight decay and invariance

- Suppose we train two 2-layer networks
  - *First network*: trained using original data: $\boldsymbol{x}=\{x_i\}$, $\boldsymbol{y}=\{y_k\}$
  - *Second network*: input and/or target variables are transformed by one of the linear transformations

$$\boxed{x_i \rightarrow \tilde{x}_i = a x_i + b} \qquad \boxed{y_k \rightarrow \tilde{y}_k = c y_k + d}$$

- Consistency requires that we should obtain equivalent networks that differ only by linear transformation of the weights

For first layer: And/or or second layer:

$\boxed{w_{ji} \rightarrow \frac{1}{a} w_{ji}}$ and/or $\boxed{w_{kj} \rightarrow c w_{kj}}$

Simple weight decay fails invariance

- Simple weight decay $\boxed{\tilde{E}(\boldsymbol{w}) = E(\boldsymbol{w}) + \frac{\alpha}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}}$
- Treats all weights and biases on equal footing
  - While resulting w_{ji} and w_{kj} should be treated differently
  - Consequently networks will have different weights and violate invariance
- We therefore look for a regularizer invariant under the linear transformations
  – Such a regularizer is $\boxed{\frac{\alpha_1}{2}\sum_{w \in W_1} w^2 + \frac{\alpha_2}{2}\sum_{w \in W_2} w^2}$
  - where W_1 are weights of first layer and
  - W_2 are the set of weights in the second layer
    – This regularizer remains unchanged under the weight transformations provided the parameters are rescaled using

$$\lambda_1 \to a^{1/2}\lambda_1 \quad \text{and} \quad \lambda_2 \to c^{-1/2}\lambda_2$$

Weight decay used in practice

- Because it can be expensive to search for the correct value of multiple hyperparameters, it is still reasonable to use same weight decay at all layers to reduce search space

L^2 parameter Regularization

- Simplest and most common kind
- Called *Weight decay*
- Drives weights closer to the origin
  - by adding a regularization term to the objectve function

$$\boxed{\Omega(\theta) = \frac{1}{2} \| w \|_2^2}$$

- In other communities also known as *ridge regression* or *Tikhonov regularization*

Gradient of Regularized Objective

- Objective function (with no bias parameter)

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

- Corresponding parameter gradient

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

- To perform single gradient step, perform update:

$$\boxed{\boldsymbol{w} \leftarrow \boldsymbol{w} - \varepsilon\left(\alpha \boldsymbol{w} + \nabla_w J\left(\boldsymbol{w}; X, \boldsymbol{y}\right)\right)}$$

- Written another way, the update is

$$\boxed{\boldsymbol{w} \leftarrow (1-\varepsilon\alpha)\boldsymbol{w} - \varepsilon\nabla_w J\left(\boldsymbol{w}; X, \boldsymbol{y}\right)}$$

  – We have modified learning rule to shrink $\boldsymbol{w}$ by constant factor 1-$\varepsilon\alpha$ at each step

To study effect on entire training

- Make quadratic approximation to the objective function in the neighborhood of minimal unregularized cost $\boldsymbol{w}^*=\arg\min_{\boldsymbol{w}} J(\boldsymbol{w})$
- The approximation is given by

$$J(\boldsymbol{w}^*)+\tfrac{1}{2}(\boldsymbol{w}-\boldsymbol{w}^*)^T H(\boldsymbol{w}-\boldsymbol{w}^*)$$

- Where H is the Hessian matrix of J wrt $\boldsymbol{w}$ evaluated at $\boldsymbol{w}^*$

Effect of L^2 regularization on optimal w

Objective function: $\tilde{J}(w; X, y) = \frac{\alpha}{2} w^T w + J(w; X, y)$

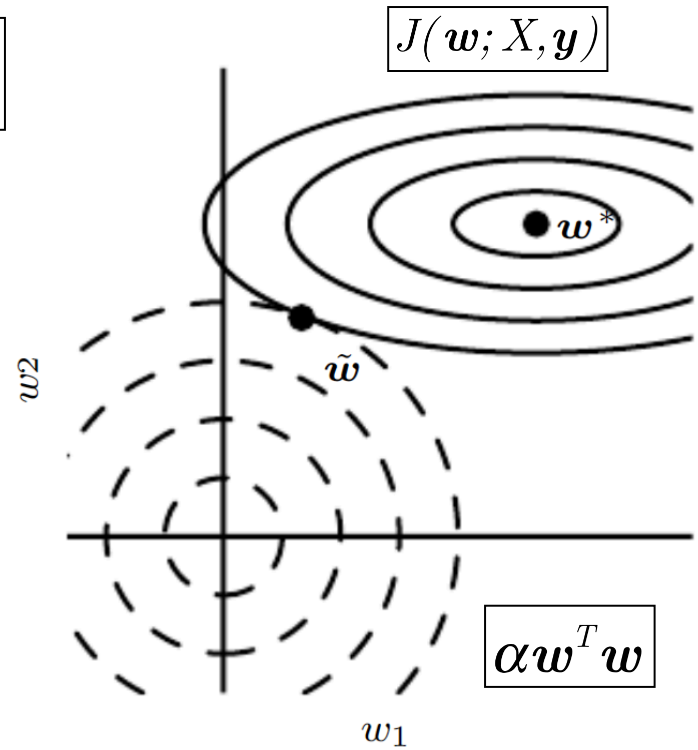
Solid ellipses:

contours of equal value of unregularized objective $J(\boldsymbol{w}; X, \boldsymbol{y})$

Dotted circles:

contours of equal value of L^2 regularizer $\alpha \boldsymbol{w}^T \boldsymbol{w}$

At point $\tilde{\boldsymbol{w}}$ competing objectives reach equilibrium



Along w_1, eigen value of Hessian of J is small. J does not increase much when moving horizontally away from $\boldsymbol{w}^*$. Because J does not have a strong preference along this direction, the regularizer has a strong effect on this axis. The regularizer pulls w_1 close to 0.

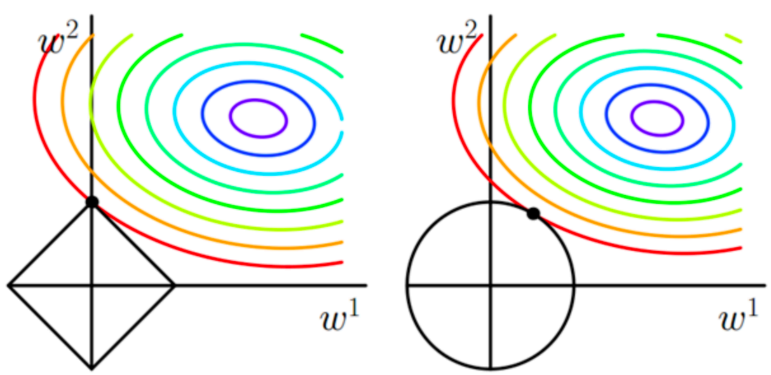
Along w_2, J is very sensitive to movements away from $\boldsymbol{w}^*$. The corresponding eigenvalue is large, indicating high curvature. As a result, weight decay affects the position of w_2 relatively little

L^1 Regularization

- L^2 weight decay is common weight decay
- Other ways to penalize model parameter size
- L^1 regularization is defined as

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

  – which sums the absolute values of parameters



(a) ℓ_1-ball meets quadratic function. ℓ_1-ball has corners. It's very likely that the meet-point is at one of the corners.

(b) ℓ_2-ball meets quadratic function. ℓ_2-ball has no corner. It is very unlikely that the meet-point is on any of axes.

Image Source: https://zhuanlan.zhihu.com/p/28023308

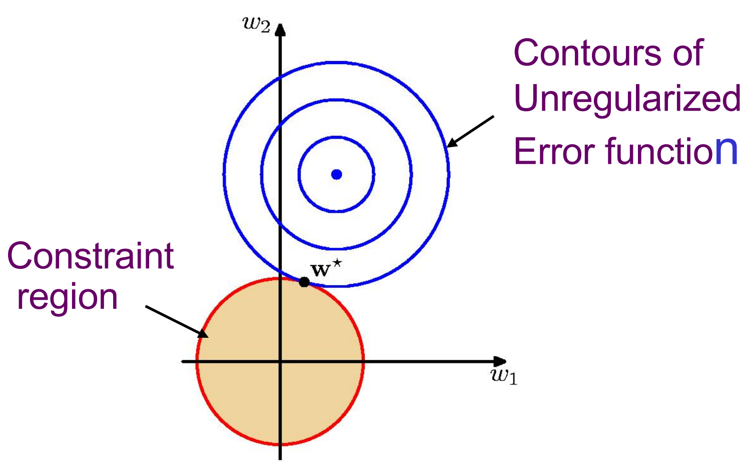
Sparsity and Feature Selection

- The sparsity property induced by L^1 regularization has been used extensively as a feature selection mechanism
  - Feature selection simplifies an ML problem by choosing subset of available features
- LASSO (Least Absolute Shrinkage and Selection Operator) integrates an L^1 penalty with a linear model and least squares cost function
  - The L^1 penalty causes a subset of the weights to become zero, suggesting that those features can be discarded

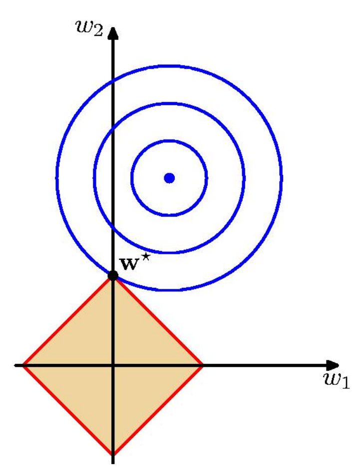
Sparsity with Lasso constraint

- With q=1 and λ is sufficiently large, some of the coefficients w_j are driven to zero
- Leads to a sparse model
  - where corresponding basis functions play no role
- Origin of sparsity is illustrated here:

Quadratic solution where w_1* and w_0* are nonzero



Minimization with Lasso Regularizer
A sparse solution with w_1*=0



Regularization with linear models

Norm Regularization:

$$Tikhonov: \quad \lambda \sum_{\forall i} w_i^2$$

$$Lasso: \quad \lambda \sum_{\forall i} |w_i|$$

$$Student-t: \quad \lambda \sum_{\forall i} \log(1+w_i^2)$$