Regularization strategies for Deep Learning

Weight Decay, Augmentation, Early Stopping, Dropout & more

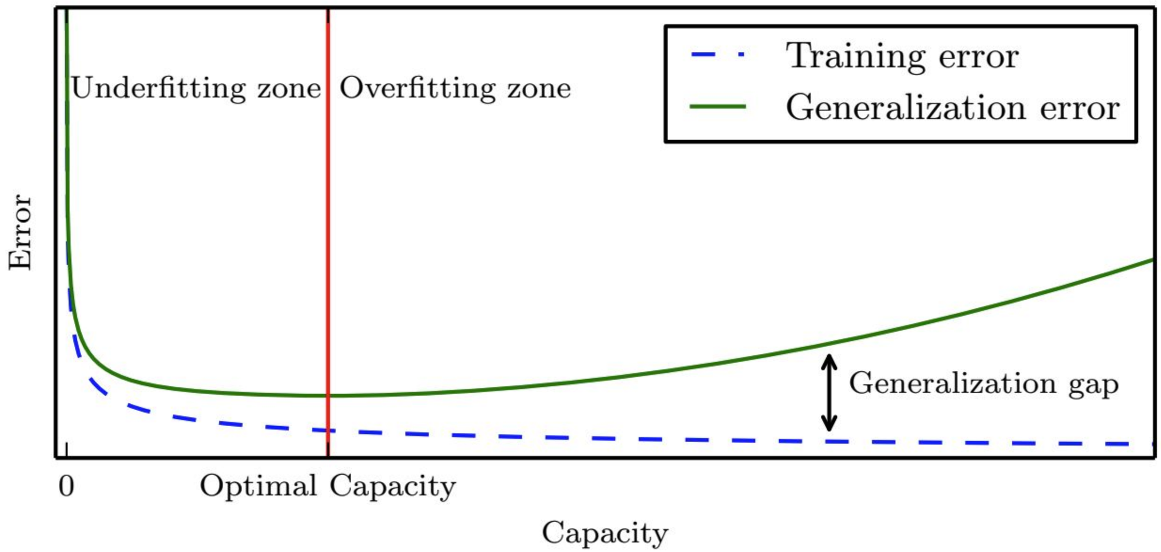
N. Rich Nguyen, PhD

SYS 6016

DATA SCIENCE

Review: Basic Concepts

- The ability to perform well on *unobserved inputs* is called **generalization.**
- **Underfitting**: when the model is not able to obtain sufficient training error.
- **Overfitting**: when the gap between training and generalization error too large.
- **Bias** and **Variance** Tradeoff: per your discussion on Piazza.

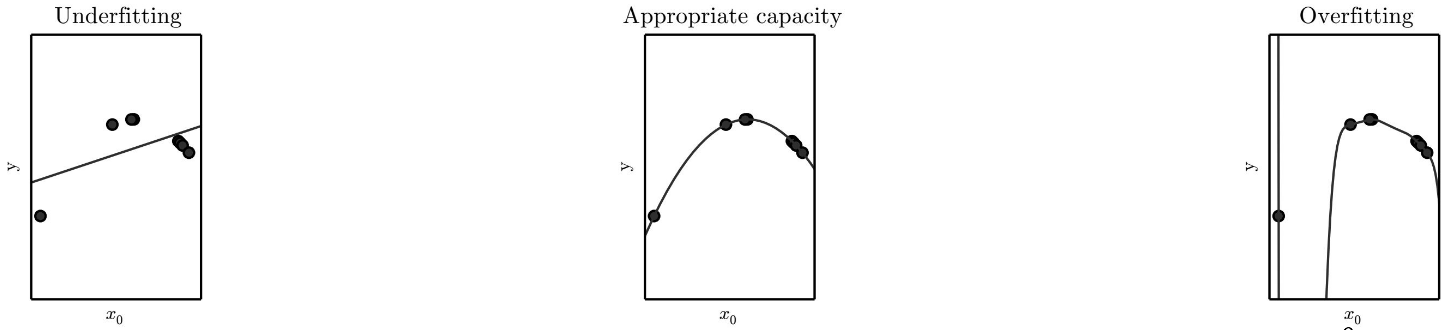


Review: Model Capacity

A model capacity is its ability to fit a wide variety of functions:

- Models with **low capacity** may struggle to fit the training set (Underfitting)
- Models with **high capacity** can overfit by memorizing properties of the training set that do not serve them well on the test set (Overfitting)

One way to control model capacity is by choosing its **hypothesis space**, the set of functions that the learning algorithms is allowed to select as being the solution.



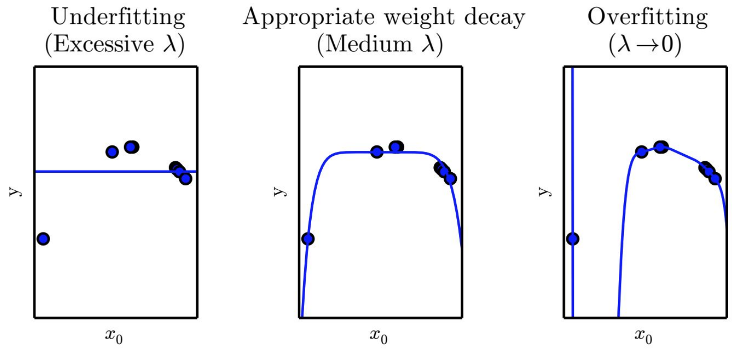
$$\hat{y} = \theta_0 + \theta_1 x_0 \qquad \hat{y} = \theta_0 + \theta_1 x_0 + \theta_2 x_0^2 \qquad \hat{y} = \theta_0 + \sum_{i=1}^{9} \theta_i x^i$$

Regularization

Regularization is the modification made to a learning algorithm that is intended to reduce its generalization error but not its training error.

For example, we can modify the loss function for linear regression to include a preference for the weights to have smaller L2 norm as a **regularizer**:

$$J(\mathbf{w}) = \mathcal{L}_{\text{mse}}(\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}}) + \lambda \mathbf{w}^\top \mathbf{w}$$



Overview: Regularization strategies

We will survey the general strategies used to regularize neural networks

1. Parameter Norms
2. Dataset Augmentation
3. Multi-task Learning
4. Early Stopping
5. Bagging
6. Dropout
7. Tangent Prop

1. Parameter Norms

Parameter Norm Penalties

Many regularization approaches are based on **limiting the model capacity** by adding a **parameter norm** penalty to the objective (loss) function:

$$J(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \underbrace{\boxed{\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})}}_{\text{Data Loss}} + \underbrace{\boxed{\lambda\Omega(\mathbf{w})}}_{\text{Norm Penalty}}$$

where λ is a hyperparameter that controls the relative contribution of the norm penalty term, Ω, relative to the standard data loss function $\mathcal{L}$

- Larger value of λ correspond to more regularization
- Setting λ to 0 results in no regularization
- Norm penalty Ω penalizes only the weights of the affine transformation
- Different choice of Ω can result in different solutions being preferred.

L^2 Parameter Regularization

L^2 regularization is aka **Ridge Regression** or **Tikhonov regularization**

The **L^2 norm** penalty commonly known as **weight decay**

$$J(\mathbf{w};\mathbf{X},\mathbf{y}) = \mathcal{L}(\hat{\mathbf{y}},\mathbf{y}) + \frac{\lambda}{2}||\mathbf{w}||^2$$

$$\nabla_{\mathbf{w}} J(\mathbf{w};\mathbf{X},\mathbf{y}) = \nabla_{\mathbf{w}}\mathcal{L}(\hat{\mathbf{y}},\mathbf{y}) + \lambda\mathbf{w}$$

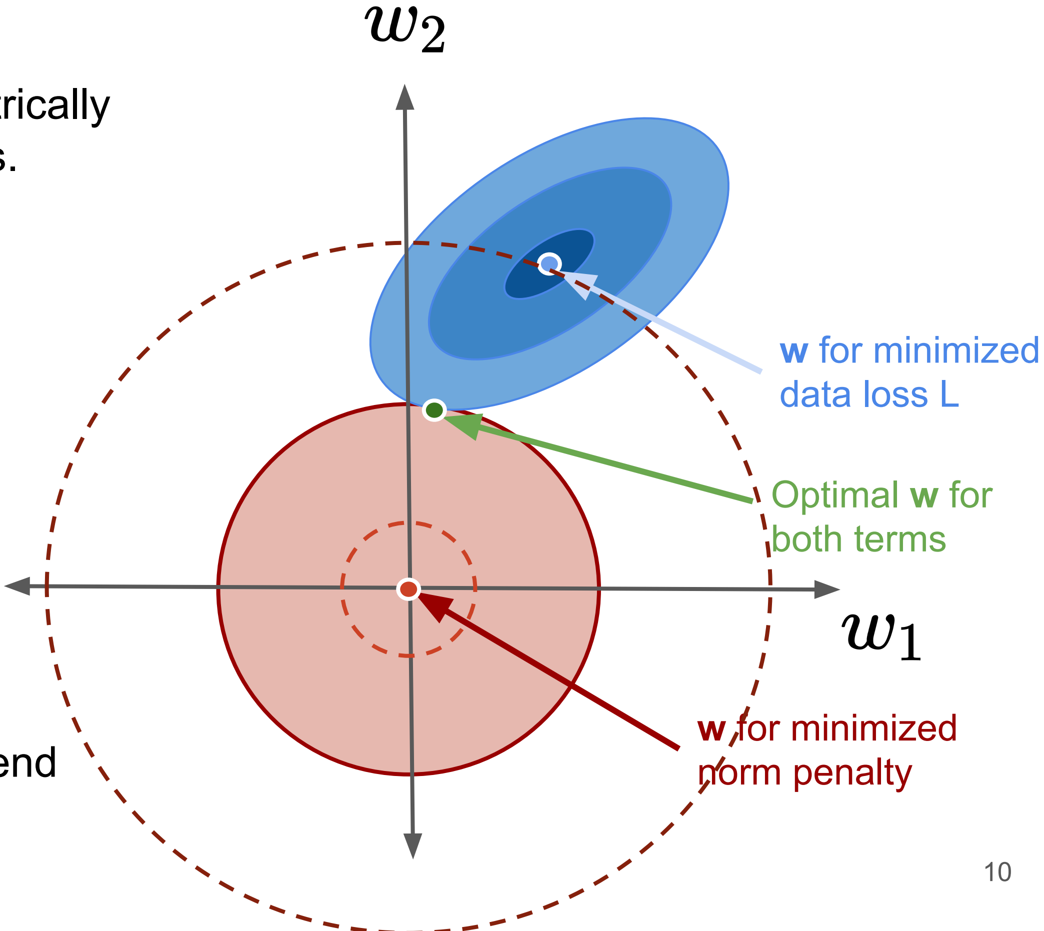
To take a single **Gradient Descent** step to update the weights:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha\nabla_{\mathbf{w}} J(\mathbf{w};\mathbf{X},\mathbf{y})$$
$$\mathbf{w} \leftarrow \mathbf{w} - \alpha(\nabla_{\mathbf{w}}\mathcal{L}(\hat{\mathbf{y}},\mathbf{y}) + \lambda\mathbf{w})$$
$$\mathbf{w} \leftarrow \boxed{\underbrace{\boxed{(1-\alpha\lambda)}}_{<1}\mathbf{w}} - \alpha\nabla_{\mathbf{w}}\mathcal{L}(\hat{\mathbf{y}},\mathbf{y})$$

Shrink the weight vector before gradient update

Geometry Interpretation

- L^2 regularizations $||\mathbf{w}||^2$ can be geometrically represented as concentric (red) circles.
- When circle is too small, the params are not useful to the model.
- When the circle region (in red) grows, due to its shape the region intersects the data contour **closer to the origin,** L^2 makes both parameters shrink and w_1 near zero.
- When the circle grows too large, you end up with a similar params as data loss.



L^1 Parameter Regularization

L^1 regularization is aka **LASSO** (least absolute shrinkage and selection operator)

L^1 norm commonly is known as the **Manhattan Distance**

$$J(\mathbf{w};\mathbf{X},\mathbf{y}) = \mathcal{L}(\hat{\mathbf{y}},\mathbf{y}) + \lambda||\mathbf{w}||_1$$

$$\nabla_{\mathbf{w}} J(\mathbf{w};\mathbf{X},\mathbf{y}) = \nabla_{\mathbf{w}}\mathcal{L}(\hat{\mathbf{y}},\mathbf{y}) + \lambda \text{sign}(\mathbf{w})$$

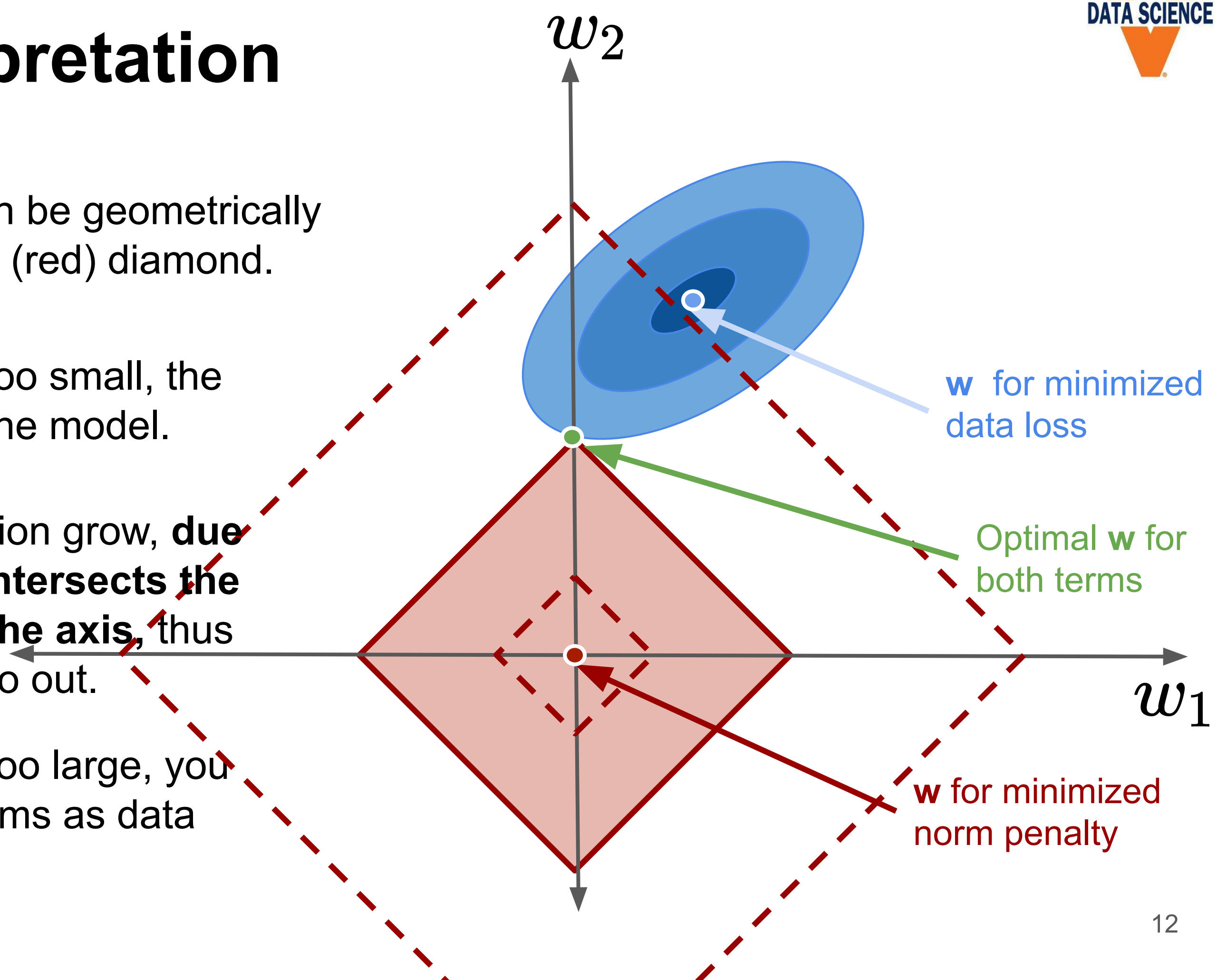
The regularization to the gradient no longer scale linearly with each w, instead it is a constant factor with a sign equal to sign(w). Thus, there is **no clean** algebraic solution to approximates ***J*** as we have just seen in L^2 regularization

L^1 norm makes the parameters to become **sparse** (contains lots of zeros)

L^1 norm tends to cause a subset of the network weights to become zero, suggesting that the corresponding signal may safely be discarded (**dead neuron**).

Geometry Interpretation

- L^1 regularizations $||\mathbf{w}||_1$ can be geometrically represented as concentric (red) diamond.
- When diamond region is too small, the params are not useful to the model.
- When as the diamond region grow, **due to its shape the region intersects the data contour on one of the axis,** thus the other parameter is zero out.
- When diamond region is too large, you end up with a similar params as data loss.



Can we combine multiple regularizations?

Elastic Net

Is a middle ground between Ridge (L2) and Lasso (L1) with a mix ratio r

$$J(\mathbf{w};\mathbf{X},\mathbf{y}) = \mathcal{L}(\hat{\mathbf{y}},\mathbf{y}) + \underbrace{r\lambda||\mathbf{w}||_1}_{\text{LASSO}} + \underbrace{(1-r)\lambda||\mathbf{w}||_2^2}_{\text{RIDGE}}$$

- Preferable to have at least a little bit of regularization, **Ridge** if a good default
- If you suspect that only a few features are actually useful, use **Lasso**
- Lasso may behave erratically when `#features>#examples`, use **Elastic Net**

Geometry Interpretation

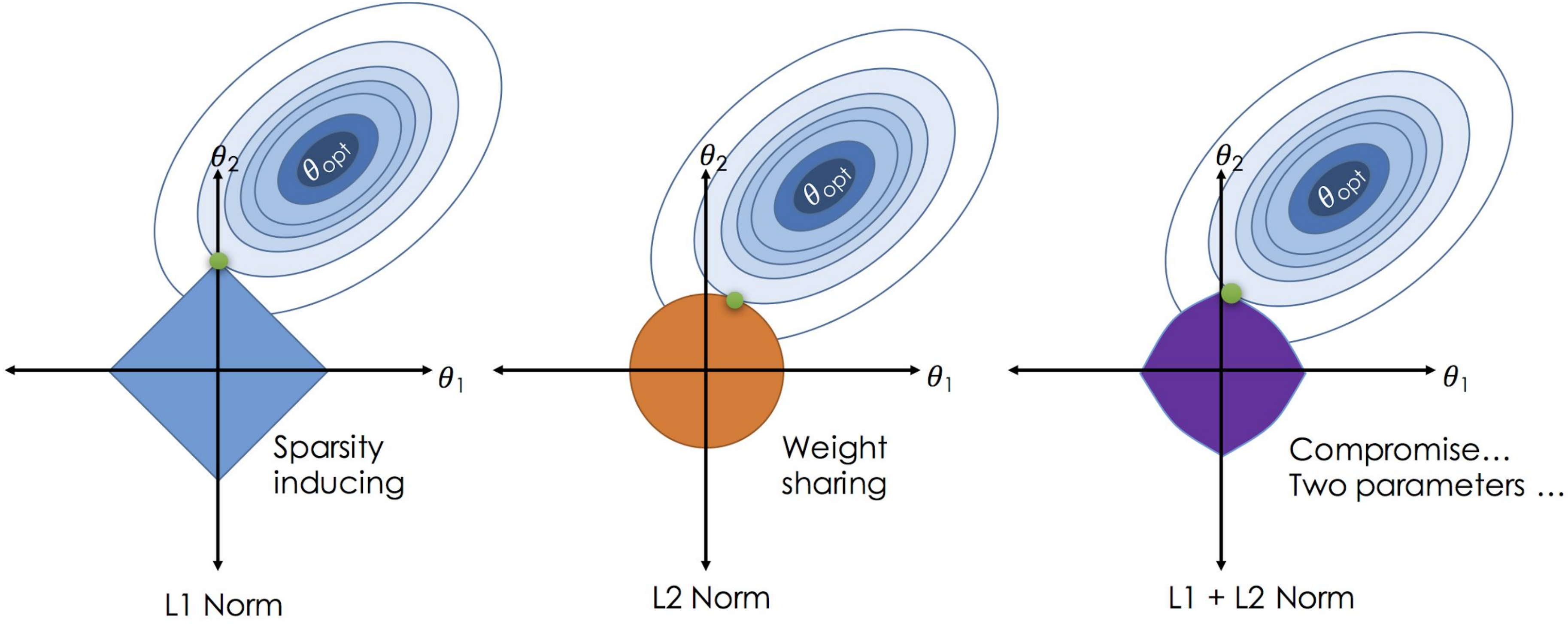
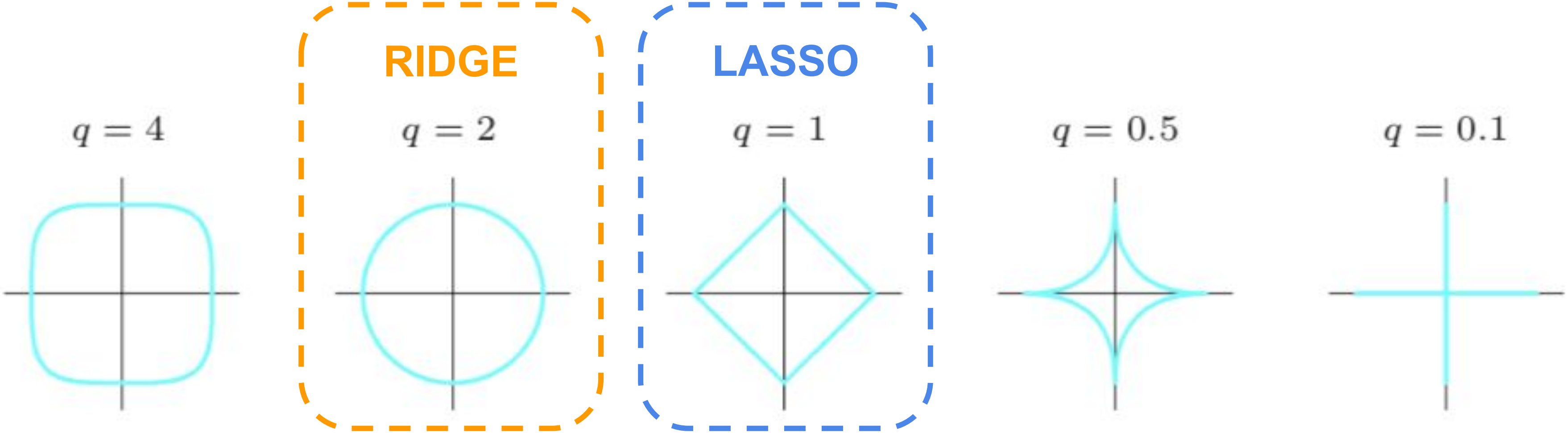


*Figure from Yin Hu, Medium

Family of Parameter Norm Models

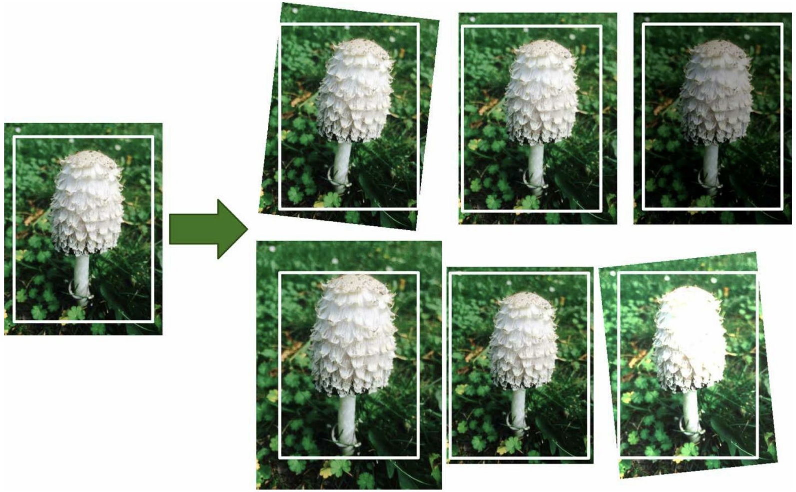
$$J(\mathbf{w}; \mathbf{X}, \mathbf{y}) = \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) + \lambda ||\mathbf{w}||_q$$



2. Dataset Augmentation

Dataset Augmentation

- Make the model generalize better by training it on **more data**
- If the amount of data available is limited, get around by creating “fake” data and augment them into the training set. How?
- Augmentation has been particularly effective for object recognition. Image transformations (ie. translating, rotating, cropping, brightness correcting, ect) often greatly improve generalization.



Injecting noise and label smoothing

- Dataset augmentation is **not** as readily applicable for many other tasks. For example: apply transformation that would change to correct class (ie. 180 rotations are not appropriate for “b” and “d”; or “6” and “9”)
- **Injecting (random) noise** in the inputs can be a form of data augmentation.
- Injecting noise to output labels? It can be harmful to maximize `log p(y|x)` when `y` is a mistake.
- Prevent this by explicitly model the noise on the label: `y` is correct with probability 1 - ε (epsilon) with small constant ε
- **Label smoothing** regularizes a model on a softmax with `k` output values by replacing hard label 0 and 1 with $\frac{\epsilon}{k-1}$ and $1-\epsilon$
- Label smoothing prevents the pursuit of hard probabilities without discouraging correct classification, and is shown prominently in modern neural nets