CSC2515 Tutorial 4

Feb 3 2015

Presented by Ali Punjani

Outline

- Neural Networks
  - Regularization and Overfitting
  - Capacity Restriction/Pruning
  - Demo (MATLAB)

Preventing overfitting

- Use a model that has the right capacity:
  - enough to model the true regularities
  - not enough to also model the spurious regularities (assuming they are weaker)

- Standard ways to limit the capacity of a neural net:
  - Limit the number of hidden units.
  - Limit the size of the weights.
  - Stop the learning before it has time to overfit.

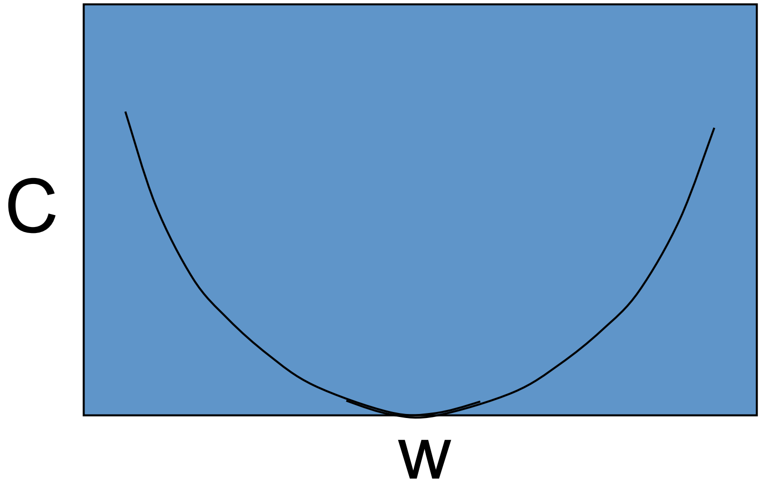
Limiting the size of the weights

Weight-decay involves adding an extra term to the cost function that penalizes the squared weights.

- Keeps weights small unless they have big error derivatives.

$$C = E + \frac{\lambda}{2} \sum_i w_i^2$$

$$\frac{\partial C}{\partial w_i} = \frac{\partial E}{\partial w_i} + \lambda w_i$$

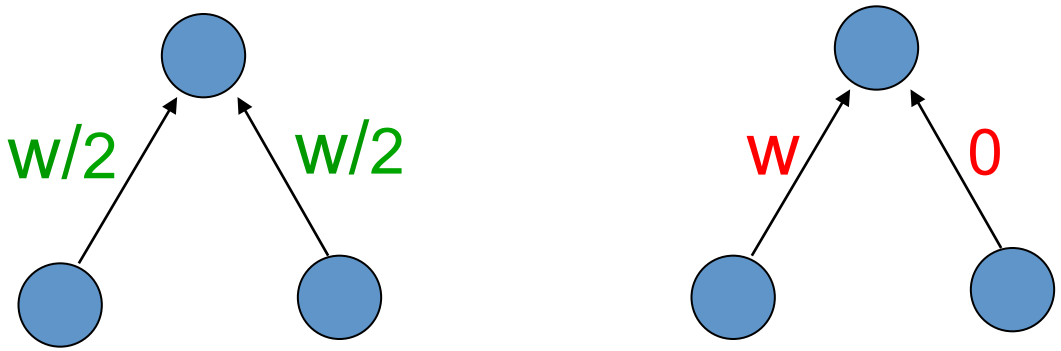


$$when \quad \frac{\partial C}{\partial w_i} = 0, \quad w_i = -\frac{1}{\lambda} \frac{\partial E}{\partial w_i}$$

The effect of weight-decay

- It prevents the network from using weights that it does not need
  - This can often improve generalization a lot.
  - It helps to stop it from fitting the sampling error.
  - It makes a smoother model in which the output changes more slowly as the input changes.

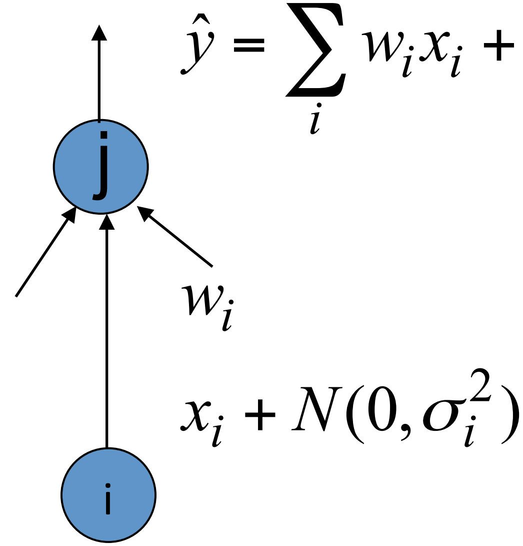
- If the network has two very similar inputs it prefers to put half the weight on each rather than all the weight on one.



Weight-decay via noisy inputs

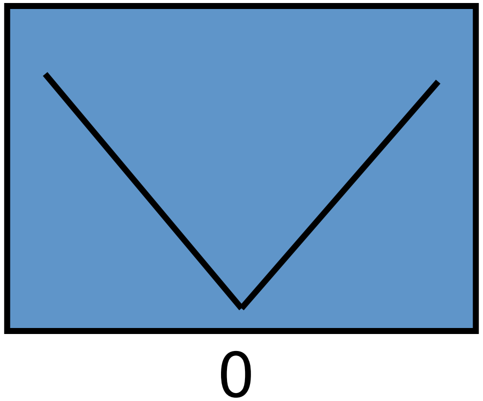
- Weight-decay reduces the effect of noise in the inputs.
  - The noise variance is amplified by the squared weight
- The amplified noise makes an additive contribution to the squared error.
  - So minimizing the squared error tends to minimize the squared weights when the inputs are noisy.
- It gets more complicated for non-linear networks.

$$\hat{y} = \sum_i w_i x_i + \sum_i N(0, w_i^2 \sigma_i^2)$$

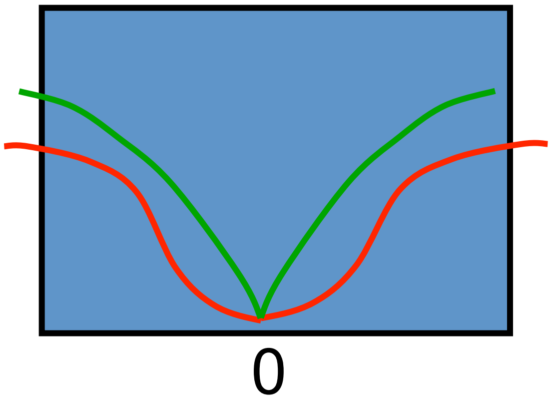


Other kinds of weight penalty

- Sometimes it works better to penalize the absolute values of the weights
  - This makes some weights equal to zero which helps interpretation



- Sometimes it works better to use a weight penalty that has negligible effect on large weights.



Pruning network weights II: Optimal brain damage

Weight saliency: analytical prediction of effectiveness of particular parameter wrt objective

Use Taylor series approximation to predict effect of perturbing some parameter (under approx)

$$\delta E = \frac{1}{2} \sum_{i} \frac{\partial^2 E}{\partial w_{ij}^2} \delta w_{ij}$$

Algorithm:

- Train network to local minimum
- Use back-prop to compute diagonal second derivatives
- Delete some parameters with low saliency (little effect of perturbing it on *E*)

Deciding how much to restrict the capacity

- How do we decide which limit to use and how strong to make the limit?
  - If we use the test data we get an unfair prediction of the error rate we would get on new test data.
  - Suppose we compared a set of models that gave random results, the best one on a particular dataset would do better than chance. But it won't do better than chance on another test set.

- So use a separate validation set to do model selection.

Using a validation set

- Divide the total dataset into three subsets:
  - Training data is used for learning the parameters of the model.
  - Validation data is not used of learning but is used for deciding what type of model and what amount of regularization works best
  - Test data is used to get a final, unbiased estimate of how well the network works. We expect this estimate to be worse than on the validation data

- We could then re-divide the total dataset to get another unbiased estimate of the true error rate.

Preventing overfitting by early stopping

- If we have lots of data and a big model, its very expensive to keep re-training it with different amounts of weight decay

- It is much cheaper to start with very small weights and let them grow until the performance on the validation set starts getting worse (but don't get fooled by noise!)

- The capacity of the model is limited because the weights have not had time to grow big.

Why early stopping works

- When the weights are very small, every hidden unit is in its linear range.
  - So a net with a large layer of hidden units is linear.
  - It has no more capacity than a linear net in which the inputs are directly connected to the outputs!
- As the weights grow, the hidden units start using their non-linear ranges so the capacity grows.