Neural Networks for Machine Learning

Lecture 9b
Limiting the size of the weights

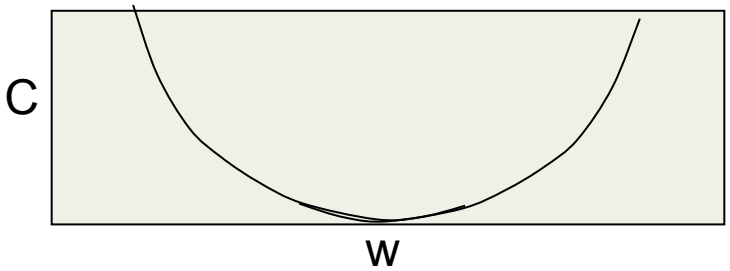
Geoffrey Hinton
Nitish Srivastava,
Kevin Swersky
Tijmen Tieleman
Abdel-rahman Mohamed

Limiting the size of the weights

- The standard L2 weight penalty involves adding an extra term to the cost function that penalizes the squared weights.
  - This keeps the weights small unless they have big error derivatives.

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

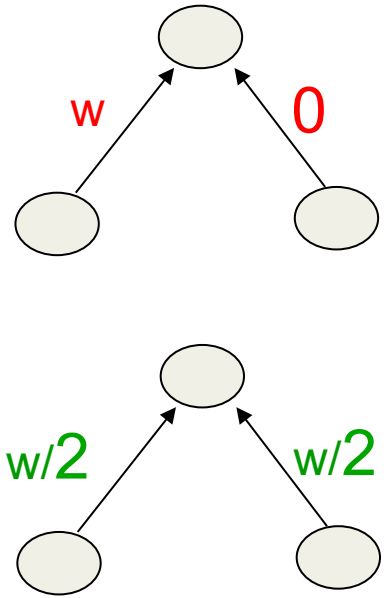
$$\frac{\partial C}{\partial w_i} = \frac{\partial E}{\partial w_i} + \lambda w_i$$



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

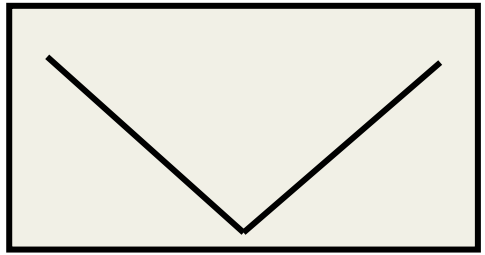
The effect of L2 weight cost

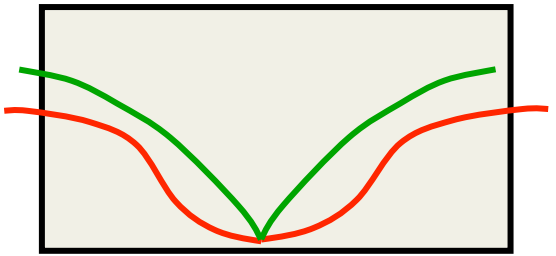
- It prevents the network from using weights that it does not need.
  - This can often improve generalization a lot because it helps to stop the network from fitting the sampling error.
  - It makes a smoother model in which the output changes more slowly as the input changes.
- If the network has two very similar inputs it prefers to put half the weight on each rather than all the weight on one.



Other kinds of weight penalty

- Sometimes it works better to penalize the absolute values of the weights.
  - This can make many weights exactly equal to zero which helps interpretation a lot.
- Sometimes it works better to use a weight penalty that has negligible effect on large weights.
  - This allows a few large weights.





Weight penalties vs weight constraints

- We usually penalize the squared value of each weight separately.
- Instead, we can put a constraint on the maximum squared length of the incoming weight vector of each unit.
  - If an update violates this constraint, we scale down the vector of incoming weights to the allowed length.
- Weight constraints have several advantages over weight penalties.
  - Its easier to set a sensible value.
  - They prevent hidden units getting stuck near zero.
  - They prevent weights exploding.
- When a unit hits it's limit, the effective weight penalty on all of it's weights is determined by the big gradients.
  - This is more effective than a fixed penalty at pushing irrelevant weights towards zero.

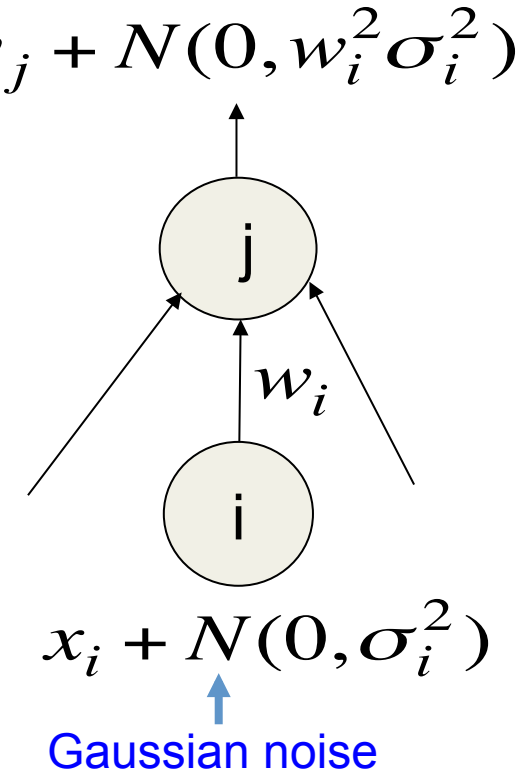
Neural Networks for Machine Learning

Lecture 9c
Using noise as a regularizer

Geoffrey Hinton
Nitish Srivastava,
Kevin Swersky
Tijmen Tieleman
Abdel-rahman Mohamed

L2 weight-decay via noisy inputs

- Suppose we add Gaussian noise to the inputs.
  - The variance of the noise is amplified by the squared weight before going into the next layer.
- In a simple net with a linear output unit directly connected to the inputs, the amplified noise gets added to the output.
- This makes an additive contribution to the squared error.
  - So minimizing the squared error tends to minimize the squared weights when the inputs are noisy.



output on one case → $y^{noisy} = \sum_i w_i x_i + \sum_i w_i \varepsilon_i$ *where* ε_i *is sampled from* $N(0, \sigma_i^2)$

$$E\left[(y^{noisy} - t)^2\right] = E\left[\left(y + \sum_i w_i \varepsilon_i - t\right)^2\right] = E\left[\left((y - t) + \sum_i w_i \varepsilon_i\right)^2\right]$$

$$= (y - t)^2 + E\left[2(y - t)\sum_i w_i \varepsilon_i\right] + E\left[\left(\sum_i w_i \varepsilon_i\right)^2\right]$$

$$= (y - t)^2 + E\left[\sum_i w_i^2 \varepsilon_i^2\right]$$

because ε_i *is independent of* ε_j *and* ε_i *is independent of* $(y - t)$

$$= (y - t)^2 + \sum_i w_i^2 \sigma_i^2$$

So σ_i^2 is equivalent to an L2 penalty

Noisy weights in more complex nets

- Adding Gaussian noise to the weights of a multilayer non-linear neural net is not exactly equivalent to using an L2 weight penalty.
  - It may work better, especially in recurrent networks.
  - Alex Graves' recurrent net that recognizes handwriting, works significantly better if noise is added to the weights.

Using noise in the activities as a regularizer

- Suppose we use backpropagation to train a multilayer neural net composed of logistic units.
  - What happens if we make the units binary and stochastic on the forward pass, but do the backward pass as if we had done the forward pass "properly"?
- It does worse on the training set and trains considerably slower.
  - But it does significantly better on the test set! (unpublished result).

$$p(s=1) = \frac{1}{1+e^{-z}}$$