Regularization.

Key problem in ML: How to generalize on unseen data?

Demo.

One technique: Regularization — any modification to learning algorithm that is intended to reduce its generalization (test) error but not its training error.

There are many ways: Restrict the model- hyper parameters: no of nodes in a decision tree.

add extra terms in the objective fn itself.

Combine multiple hypotheses - Ensemble methods.

Restrict the parameter space. - most common.

SVM - $\frac{1}{2} w^T w + C \sum_i \xi_i$

We want — a good trade where bias is increased for reduced variance.

Bayesian viewpoint — Adding prior to the estimation: eg: L_2 reg. for linear regression is a MAP estimate with Normal prior on W.

In the context of DL, we regularize to generalize over true data-generating distributions that are often very complicated

eg: Image/speech classification task — true data generating distro is the entire universe(?)

Only weights are regularized - not the biases.

reasons — 1) Biases only control single variable (threshold).

(weights quantify the interactions b/w features)

2) Regularizing bias induces overfitting

∵ intuitively thr for neuron firing should not be restricted.

Parameter norm penalties:

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

Bias-variance Dilemma:

Error of a function approximator,

$$E = \lim_{N\to\infty} \frac{1}{2N} \sum_{i=1}^{N} \sum_{K} \left\{ f_K(x^i; W) - y_K^i \right\}.$$

$f_K \rightarrow$ estimate for K^{th} output variable

$y \in \mathbb{R}^K$

$$= \frac{1}{2} \sum_K \int\int \left\{ f_K(x; W) - y_K \right\} P(y_K, x)\, dy_K\, dx$$

Define $\langle y_K | x \rangle = \int y_K P(y_K|x)\, dy_K$

$$\langle (y_K)^2 | x \rangle = \int (y_K)^2 P(y_K|x)\, dy_K$$

Can show $E = \frac{1}{2} \sum_K \int \left\{ f_K(x; W) - \langle y_K | x \rangle \right\}^2 P(x)\, dx$ ← Tunable through learning

$$+ \frac{1}{2} \sum_K \int \left\langle y_K^2 | x \right\rangle - \langle y_K | x \rangle^2 \Big\} P(x)\, dx$$

↑ I/d of approximator

noise in data.

Can't change this.

Minimal Error occurs when.

$$f_K(x_K; W) = \langle y_K | x \rangle$$

However, since we don't know $\langle y_k|x\rangle$ we have to tweak ws in f to make error as small as possible. $\left(\hat{y}(x) \equiv f(x,w)\right)$.

Simply put, $$E = \frac{1}{2}\int \{\hat{y}(x) - \langle y|x\rangle\}^2 p(x)\,dx$$

$$+ \frac{1}{2}\int \{\langle y^2|x\rangle - \langle y|x\rangle\}^2 p(x)\,dx.$$

A measure of how close the approximator to the desired one is given by

$$\{\hat{y}(x) - \langle y|x\rangle\}^2 \quad \text{---} (1)$$

Particular

This will depend on the dataset where the network was trained.

To eliminate this dependence, Lets average (1) over all possible datasets.

measure of error $\hat{E}$ $= \mathcal{E}_D\left[\{\hat{y}(x) - \langle y|x\rangle\}^2\right]$ (f of x).

$\mathcal{E}_D \rightarrow$ Expectation over all datasets.

A perfect predictor will make it zero.

$$\left[\hat{y}(x) - \langle y|x\rangle\right] = \left\{\hat{y}(x) - \mathcal{E}_D\left[\hat{y}(x)\right]\right\} + \mathcal{E}_D\left[\hat{y}(x) - \langle y|x\rangle\right]$$

$$\hat{\mathcal{E}} = \mathcal{E}_D\left[\left\{\hat{y}(x) - \langle y|x\rangle\right\}^2\right] = \text{(can be shown to be equal to below)}$$

$$\left\{\mathcal{E}_D\left[\hat{y}(x) - \langle y|x\rangle\right]\right\}^2 + \mathcal{E}_D\left[\left\{\hat{y}(x) - \mathcal{E}_D\left[\hat{y}(x)\right]\right\}^2\right]$$

$$\text{Bias}^2 \quad \text{Variance}$$

$$\left\{\mathcal{E}_D\left[\hat{y}(x) - \langle y|x\rangle\right]\right\}^2 + \mathcal{E}_D\left[\left\{\hat{y}(x) - \mathcal{E}_D\left[\hat{y}(x)\right]\right\}^2\right]$$

Total Error = Bias^2 + variance.

Bias – The extent to which the average (over all data sets) of the network f^n differs from the desired function.

Variance — measures the extent to which the network f^n $\hat{y}(x)$ is sensitive to the choice of a particular dataset D.

(3)

Have to reduce both bias & variance

62

Now, Bias & Variance are fns of x.

Lets look at Bias_{avg}, Variance_{avg} by integrating out the x.

$$\therefore \quad \text{Bias}_{avg}^2 = \frac{1}{2} \int \left\{ \mathcal{E}_D [\bar{y}(x) - \langle y|x \rangle \right\}^2 P(x)\, dx$$

$$\text{Variance}_{avg} = \frac{1}{2} \int \mathcal{E}_D \left[\left\{ \bar{y}(x) - \mathcal{E}_D [\bar{y}(x)] \right\}^2 \right] P(x)\, dx.$$

Lets study the behaviour of B & V.

Consider a regression problem with the target function being a smooth parabola.

Data is generated by adding a small noise to $h(x^n)$.

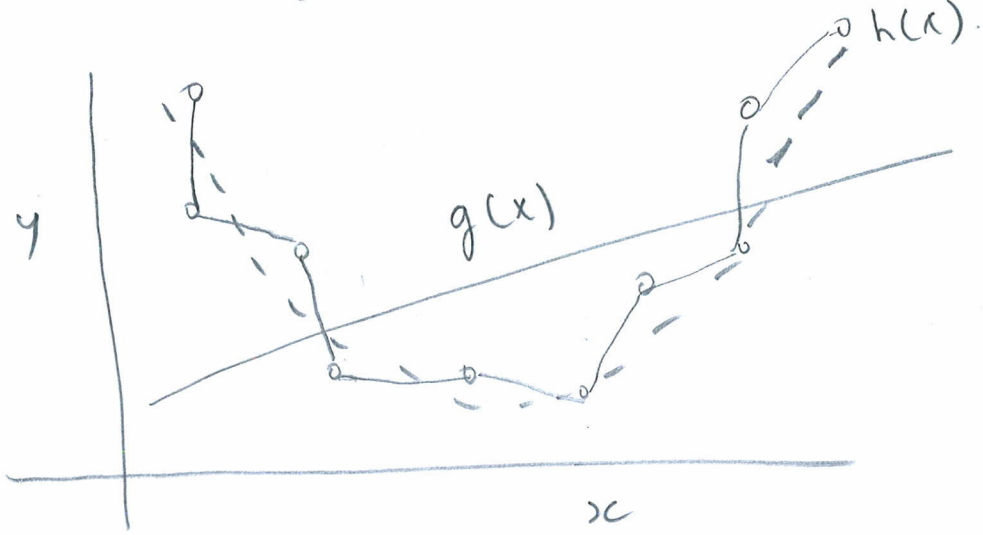
$$y^n = h(x^n) + \epsilon^n$$

Optimal mapping function $\langle y|x \rangle = h(x)$.

Now, Lets choose $\bar{y}(x)$ to be some $g(x)$ which is completely i/d of data.

In this case the variance term in E_m vanishes because $\mathcal{E}_D [\bar{y}(x)] = g(x) = \bar{y}(x)$ [I/d of data].

However the bias term will be typically high unless we have some prior knowledge that helps us to choose $g(x)$



The opposite extreme is to choose

$$\tilde{y}(x^n) = h(x^n) + \epsilon \quad \& \text{ interpolate}$$

In this case, Bias = 0 $\therefore \mathcal{E}_D[\tilde{y}(x)] = \mathcal{E}_D[h(x) + \epsilon]$

$$= h(x) = \langle y | x \rangle.$$

However, variance

$$= \mathcal{E}_D\left[\{\tilde{y}(x) - \mathcal{E}_D(\tilde{y}(x))\}^2\right]$$

$$= \mathcal{E}_D\left[\{\tilde{y}(x) - h(x)\}^2\right] = \mathcal{E}_D[\epsilon^2]$$

= variance of noise (high).

Therefore, there is a trade-off b/w B & V.

f that closely fits the data — low bias / high variance

[overfitting] => large E_{out}.

variance can be decreased by smoothing the f

But taken too far — bias becomes too large

=> large E_{out}. [underfitting]

Question: How to min both B & V?

One answer: Have a large (infinite) dataset.

=> use complex models & reduce bias.

see enough samples to reduce variance.

One reason why DNN works well on large-scale data sets.

Question: How do you do that on limited data.

[which is often the case in practice].

Another way: Use some prior knowledge that you have about unknown $h(x)$ → Example of sq waves.

Now, use that knowledge to constrain $\vec{y}(x)$ so that bias is not increased too much. (Biased minds make better decisions).

This is known as regularization: Any modification to learning algo so that generalization is improved.

Regularization: Several strategies. (Go to page 57)

Strategy 1: Less is more! [Parameter norm Penalties].

Data that we get with AI tasks are often sparse. [Demo of images from white noise].

When data is sparse, the dominant component of error.

Cognetive systems performs well with sparse data.

Combating bias → require rich class of models

~~Go~~ V → placing restrictions on these models

Examples of simple models (Heuristics) performing remarkably better (Harry Markowitz - mean-variance (Portfolio) vs $1/N$ portfolio allocation.

$\Rightarrow$ Simpler models (High bias) are better for sparse data.

Summary: Look for simpler models to improve $\hat{E}_m$.

Parameter norm penalties:

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

L Add a penalty term constraining the parameter space to the objective f^n.

[DNN - biases are not regularized, Go to Pg. 57].

L^2 regularization (Weight decay, ridge, Tikhonov).

$\Omega(\theta) = \frac{1}{2}\|w\|_2^2$ → Drives weight closer to origin (can be anything but origin is choosen ∵ +ve & -ve unknown)

decay factor ↓

$\hat{J} = \frac{\alpha}{2} w^T w + J$

$\nabla_w \hat{J} = \alpha w + \nabla_w J$ Forces some neurons to behave linearly.

$$w^{k+1} = w^k - \epsilon(\alpha w^k + \nabla_w J)$$

$$= (1 - \epsilon\alpha) w - \epsilon \nabla_w J.$$

Shrink the weight by a cnst factor on each step before gradiant update

Let's study what L^2 does in the entire course of training.

P.T.O.

Let $w^* = \arg\min_w J(w)$ → minimizer for un-regularized cost.

As before, Lets make a quadratic approximation of $\hat{J}$ around w^*

$$\hat{J}(\theta) = J(w^*) + \frac{1}{2}(w-w^*)^T H (w-w^*)$$

Gradiant term is absent ∵ w^* is a minimizer.

$$\nabla_w \hat{J}(w) = H(w-w^*)$$

minima occurs when $\nabla_w \hat{J}(w) = 0$.

with regularization,

$$\alpha \tilde{w} + H(\tilde{w} - w^*) = 0 \qquad \tilde{w} - \text{minima with regularized weight.}$$

$$(H + \alpha I)\tilde{w} = Hw^*$$

$$\tilde{w} = (H + \alpha I)^{-1} H w^*$$

as $\alpha \to 0$, $\tilde{w} \to w^*$ (obvious)

But what happens with growing α?

As usual, since H is real & symmetric,

$$\exists\ Q, \Lambda, \text{ s.t. } H = Q \Lambda Q^T \quad (\text{E.V.D.})$$

$$\therefore \quad \hat{w} = \left(Q \Lambda Q^T + \alpha I\right)^{-1} Q \Lambda Q^T w^*$$

$$= \left[Q\left(\Lambda + \alpha I\right) Q^T\right]^{-1} Q \Lambda Q^T w^*$$

$$= Q \left(\Lambda + \alpha I\right)^{-1} \Lambda Q^T w^*.$$

⇒ weight decay rescales w^* along the axes defined by Evecs of H.

Specifically, $\hat{w}$ alligned with i^{th} Evec, is rescaled by a factor of $\frac{d_i}{d_i + \alpha}$.

⇒ Along the directions where H has relatively higher Evals, $d_i \gg \alpha$, the effect of L^2 is small. when $d_i \ll \alpha$, weights gets shrunk (Demo).

Example with linear regression.

$$J = \| w^T x - y \|_2^2 + \frac{1}{2} \alpha \| w \|_2^2$$

$$\Rightarrow \quad w^* = (X^T X)^{-1} X^T y$$

$$\hat{w} = (X^T X + \alpha I)^{-1} X^T y.$$

70

Now, $X^T X$ → Co-variance matrix.

* L^2 increases the "perceived" variance.

⇒ weights on features whose covariance with the target output is low a (w.r.t to the new variance) are shrinked.

L^2 is also equivalent to MAP estimation. with gaussian priors

---

L^2 is not sparse - it still makes weights to have no-zero values.

Desirable to have zero-values for weights that are useless.

Solution: L^1 regularization.

$$\Omega(\theta) = \|w\|_1 = \sum_i |w_i|$$

$$\hat{J}(w; x, y) = \alpha \|w\|_1 + J(w; x, y).$$

$$\nabla_w \tilde{J}(w; x, y) = \alpha \, \text{sign}(w) + \nabla_w J(x, y; w)$$

Again making quadratic approx,

$$\nabla_w \tilde{J}(w) = H(w - w^*)$$

Since L' does not have clean algebraic expressions in case of full H, lets further simplify by assuming H is diagonal.

i.e, $H = \text{diag}\left([H_{1,1}, \ldots H_{n,n}]\right)$, $H_{i,i} > 0$

[Easy to get with PCAed data]

Now, $\tilde{J}(w; x, y) = J(w^*; x, y) +$

$$\sum_i \left[\frac{1}{2} H_{i,i} (w_i - w_i^*)^2 + \alpha |w_i|\right]$$

—(1)

Let $\tilde{w}_i$ be the minimizer with L'.

Have to find minimizer of (1) w.r.t. w_i

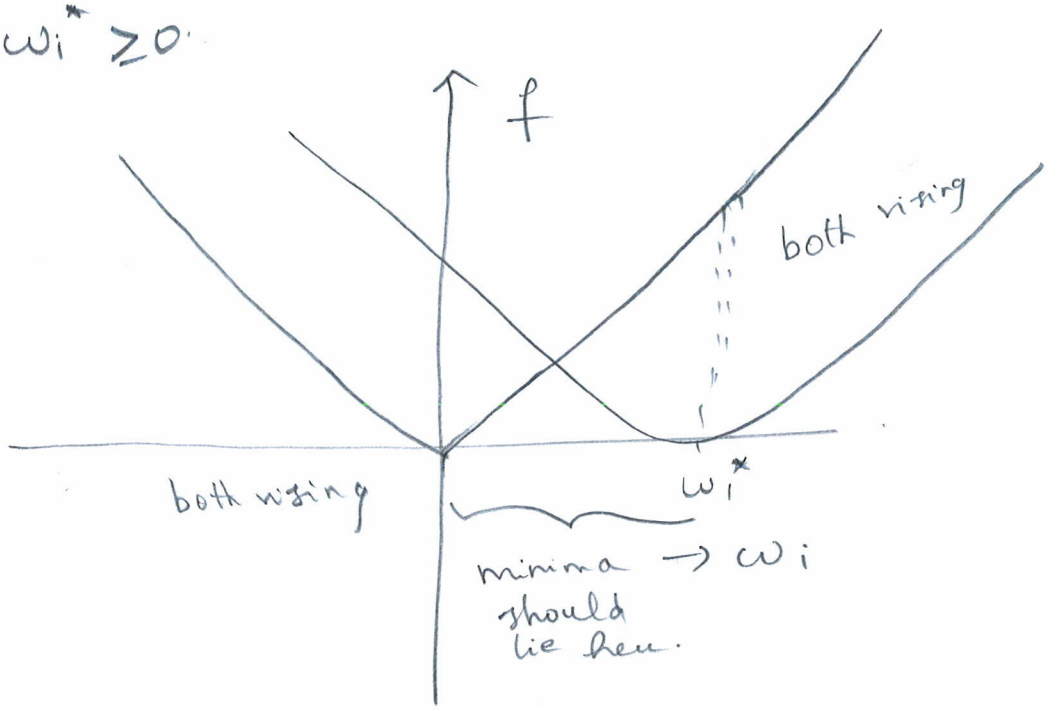
First term does not contribute to minima ∵ it is i/d of w_i. Lets Concentrate on the 2nd term.

29/1 We have to find the minima of the following. 72

$$\frac{1}{2} H_{ii} (w_i - w_i^*)^2 + \alpha |w_i| = f$$

↓ Parabola ↑ V-shaped curve.

Assume $w_i^* \geq 0$.



Also, algebraically, if $w_i^* \geq 0$,

w_i ... w_i^* ... 0 ... w_i^*

$$\nabla_{w_i} f = \underbrace{(w_i - w_i^*) H_{ii}}_{\text{-ve}} + \underbrace{\alpha \, \text{sign}(w_i)}_{\text{+ve}}$$

for critical point, $\nabla_{w_i} f = 0$.

if $w_i^* > 0$, w_i can't be negative ∴

$\nabla_{w_i} f$ should be zero.

Further $\text{sign}(w_i) = w_i \Rightarrow \hat{w}_i = w_i^* - \frac{\alpha}{H_{ii}}$

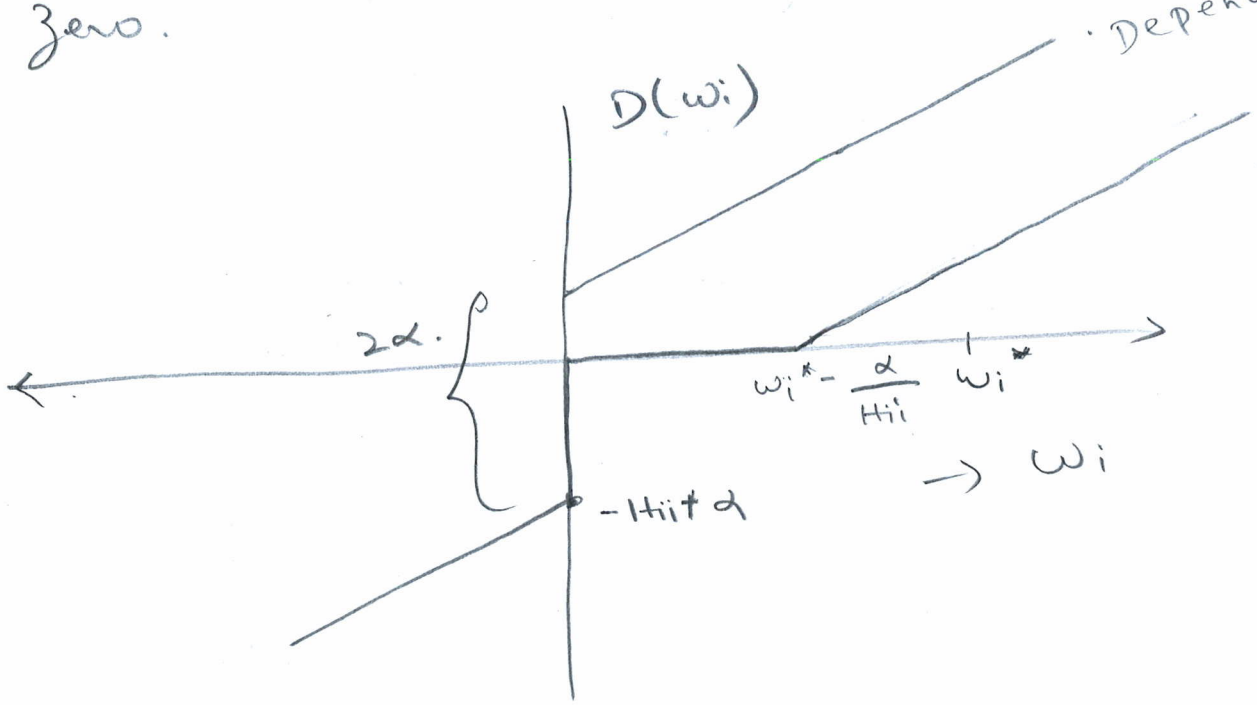
Also from above graph, $0 \leq \hat{w}_i \leq w_i^*$

$$\Rightarrow \hat{w}_i^* = \max \left\{ w_i^* - \frac{\alpha}{H_{ii}}, 0 \right\}$$

Another argument with derivative. Let $w_i \geq 0$.

$$D(w_i) = H_{ii}(w_i - w_i^*) + \alpha \underbrace{\text{sign}(w_i)}_{\text{"}\alpha w_i}$$

Now, minimum occurs when $D(w_i)$ crosses zero.



∴ Zero crossing occurs either at 0, or at $w_i^* - \frac{\alpha}{H_{ii}}$

Thus, $\hat{w}_i^* = \max\left\{ w_i^* - \frac{\alpha}{H_{ii}}, 0 \right\}$.

A similar argument can be made when $w_i < 0$.

Thus, $\hat{w}_i = \text{sign}(w_i^*) \max\left\{ |w_i^*| - \frac{\alpha}{H_{ii}}, 0 \right\}$.

74

Now, Lets interpret this.

For the case when $w_i^* > 0$, there are two cases.

i) $\alpha w_i^* \leq \frac{\alpha}{|H_{ii}|} \Rightarrow \hat{w}_i^* = 0$

$\Rightarrow$ along direction i, along which regularizer overwhelms the other.

H_{ii} is small $\Rightarrow$ f^n varies less along that direction specified by i

(Contribution of J to $\tilde{J}$ is overwhelmed by L^1)

ii) when $w_i^* > \frac{\alpha}{|H_{ii}|}$,

L^1 just shifts the minimum in the direction i by $\frac{\alpha}{|H_{ii}|}$

Summary: L^1 induces sparsity & might be better in some cases where there are too-many parameters

[Again simplifying the model to increase the Bias]. Demo.

Info: L^1 is MAP with isotropic laplacian distr.

L^2 can't select features L^1 is a feature selector

L^1 is robust to outliers (sq term in L^2 vs mod in L^1).

L^2 is comp. efficient (L^1 is not on non-sparse case)

29/1.

Norm penalties as Constrained optimization.

$$\tilde{J} = J + \alpha \Omega(\theta)$$

Equivantent to $\min J$
$s.t \quad \alpha \Omega(\theta) < k.$

$\therefore$ Lagrangian $L = J + \alpha(\Omega(\theta) - k)$

Equivalent to norm-penalties.

if Ω is L^2 → weights are Constraind to be inside the L^2-norm ball.

Similaly for L^1.

One can put norm-constraints on each column of W matrix - different layers have different weights.

Strategy 2: Noise injuction.
more data ⇒ lower bias & variance

Dataset Augmentation: Create fake data.

Example of digit recogtion: Transform to create more points.

Be careful not to mix 6 & 9, b & d.

If we know $P(x, y)$ can generate - Egg & Chiken problem.

Noise injection can also be done at the hidden-unit level.

Dropout — multiplication by noise.

Adding noise to weights is also possible stochastic implementation of Bayesian inference over weights.

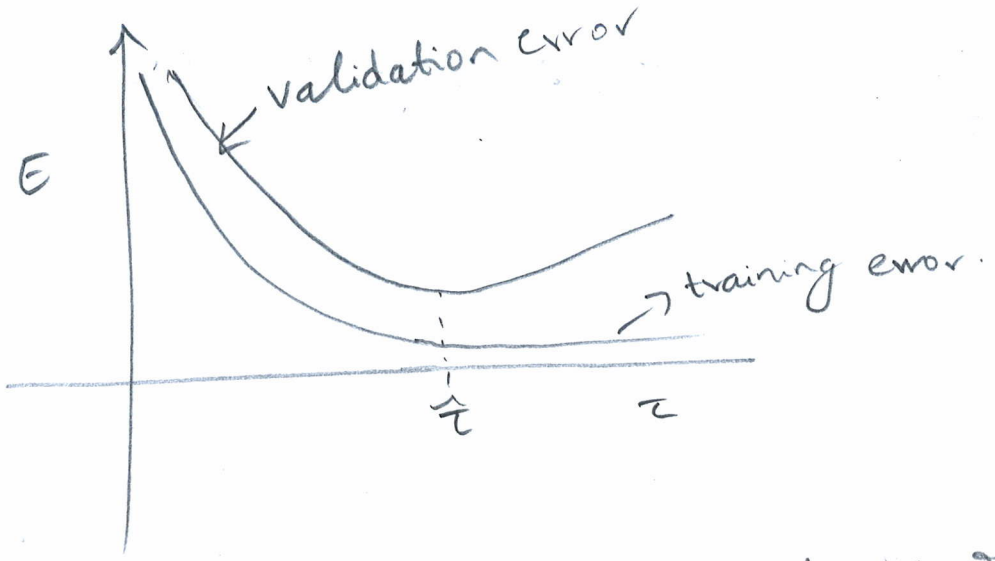
Adding noise is also equivalent to regularization. traditional.

Label-Smoothing — injecting noise at the output with $\frac{\epsilon}{k-1}$ instead of $1-\epsilon$, Eg: softmax 0.1.
Assume y is correct (18).

Regularization - Continued.

Early stopping : Training involves iterative reduction of cost.

Typical training curve:



Validation error increases beyond a point ∴ of overfitting.

Early stopping: stop training. as VE starts increasing

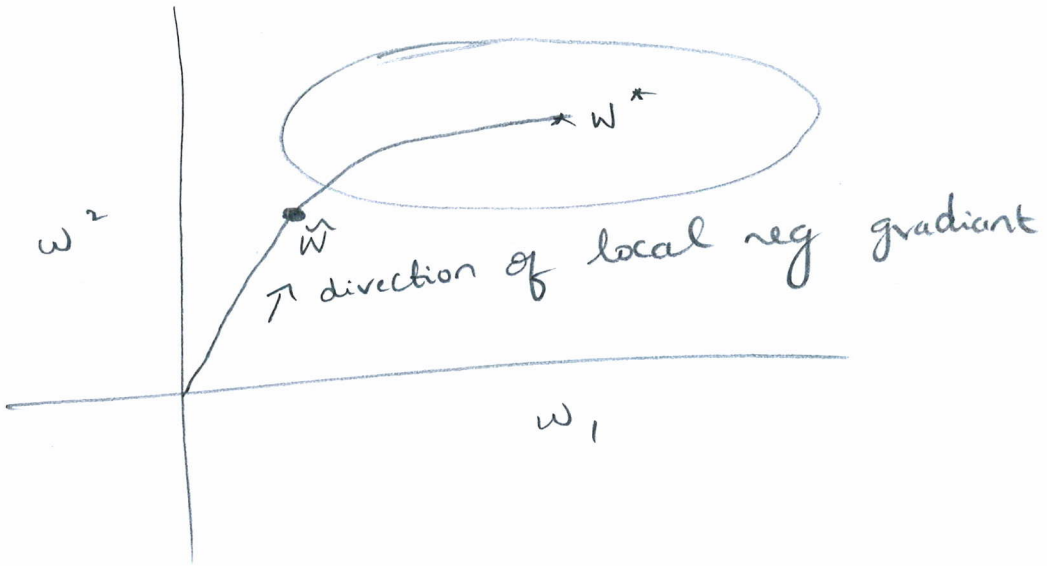
unobstrusive way of regularization (requires no change in obj. fn).

Can be used with other regularizers as well.

Can use this as init value & retrain the whole model with all the training data.

(1).

Why is it a regularizer?



E.S acts as a weight-decay. In fact the precise relationship is derived as follows.

Let w^τ be the weight-update at τ th iteration.

$$w^\tau = w^{\tau-1} - \epsilon \nabla_w \hat{J}(w^{\tau-1}). \quad -(1)$$

$$\hat{J} = J(w^*) + \frac{1}{2}(w-w^*)^T H (w-w^*).$$

$$(1) = w^{\tau-1} - \epsilon H (w^{\tau-1} - w^*) \quad -(2).$$

$$\therefore \nabla_w \hat{J} = H(w-w^*).$$

$$w^\tau - w^* = (I - \epsilon H)(w^{\tau-1} - w^*)$$

(2).

$H = \varphi \Lambda \varphi^T \quad [\varepsilon \vee D]$

$$\therefore \quad \omega^\tau - \omega^* = \left(I - \epsilon \varphi \Lambda \varphi^T\right)\left(\omega^{\tau-1} - \omega^*\right)$$

$$\varphi^T\left(\omega^\tau - \omega^*\right) = \left(I - \epsilon \Lambda\right) \varphi^T \left(\omega^{\tau-1} - \omega^*\right) \quad -(3)$$

Now, Let $\omega^0 = 0$.

Then, $\varphi^T \omega^\tau = \left[I - \left(I - \epsilon \Lambda\right)^\tau\right] \varphi^T \omega^* \quad -(1)$

for L_2,

$$\hat{\omega} = \varphi \left(\Lambda + \alpha I\right)^{-1} \Lambda \varphi^T \omega^*$$

comparing,

$$\varphi^T \hat{\omega}_{L_2} = \left(\Lambda + \alpha I\right)^{-1} \Lambda \varphi^T \omega^*$$

$$\varphi^T \hat{\omega}_{L_2} = \left[\left(I - \Lambda + \alpha I\right)^{-1} \alpha\right] \varphi^T \omega^* \quad -(2)$$

$\Rightarrow$ If $\epsilon, \alpha \neq I$ are chosen s.t.

Matrix cookbook. pg-17.

$$\left(I - \epsilon \Lambda\right)^\tau = \left(\Lambda + \alpha I\right)^{-1} \alpha.$$

then L_2 & ES are equivalent

80.

Taking log & using Taylors Series for $\log(1+x)$, if d_i are small s.t ($\epsilon \lambda_i \ll 1$ & $d_i/\alpha \ll 1$) Then

$$\tau \approx \frac{1}{\epsilon \alpha}$$

$$\text{or } \alpha \approx \frac{1}{\tau \epsilon}$$

[not clear yet

$$\tau \log(I - \epsilon \Lambda) = -1 \log(\Lambda + \alpha I) + \log \alpha.$$

$$\tau \log(1 - \epsilon d_i) = -1 \log(d + \alpha) + \log \alpha.$$

$$\tau + \frac{-1(\epsilon d)}{1 - \epsilon d} = -1 \quad \ldots \quad]$$

$\epsilon \rightarrow$ learning rate, $\tau \rightarrow$ iteration index.

$\tau\epsilon$ behaves as effective capacity.

Kind of reciprocal of weight decay

[inverse of capacity).

So far, we have seen two approaches for regularization: Parameter norm- penalties
Controlling the effective capacity

Third important approach: Confusion is good!!

Noise induction: Go to 76 for intro
after Bagging.

Another approach for regularization: Democracy.

Committees of networks or bagging: Netflix Grand prize ↓ predict movie ratings.

Demo with the example of 9, 6 vs 8.

Bagging does better than individual classifiers

Let $h(x)$ be the true fn we are seeking to approximate.

Let there be L networks learning this function.

82

Let $y_i(x) = h(x) + \epsilon_i(x)$ be the o/p of each network where $\epsilon_i(x)$ is some error fn.

Now, Avg sum-of-sq. error for $y_i(x)$ is

$$E_i = \mathcal{E}\left[\{y_i(x) - h(x)\}^2\right] = \mathcal{E}[\epsilon_i^2]$$

$$\mathcal{E}[\epsilon_i] \equiv \int \epsilon_i^2(x)\, p(x)\, dx$$

Now avg error made by all acting individually is

$$E_{AV} = \frac{1}{L}\sum_{i=1}^{L} E_i = \frac{1}{L}\sum_{i=1}^{L} \mathcal{E}[\epsilon_i^2]$$

Let's now introduce a simple bagging scheme.

$$y_{com}(x) = \frac{1}{L}\sum_{i=1}^{L} y_i(x)$$

avg of all o/ps.

Now, error due to the committee is

$$E_{COM} = \mathcal{E}\left[\left(\frac{1}{L}\sum_{i=1}^{L} y_i(x) - h(x)\right)^2\right]$$

$$= \mathcal{E}\left[\left(\frac{1}{L}\sum_{i=1}^{L} \epsilon_i\right)^2\right]$$

Assume $E[\epsilon_i(x)] = 0 \quad E[\epsilon_i \epsilon_j] = 0$

Then $E_{COM} = \frac{1}{L^2} \sum_{i=1}^{L} E[\epsilon_i^2] = \frac{1}{L} E_{AV}$

$\Rightarrow$ there is a dramatic reduction in error by a factor of L^2.

Not possible in practice $\because E[\epsilon_i \epsilon_j] \neq 0$.

However, Cauchy's inequality

In general $\Rightarrow \left(\sum_{i=1}^{L} \epsilon_i \right)^2 \leq L \sum_{i=1}^{L} \epsilon^2$

$\Rightarrow E_{COM} \leq E_{AV}$

One can also take $y_{GEN} = \sum_{i=1}^{L} \alpha_i y_i(x)$

for which it can be shown that $E_{GEN} \leq E_{COM}$

Problem: Too many networks to be trained - memory!

Another problem: redundancy.

84

In general if $E[\epsilon_i^2] = v$ & $E[\epsilon_i \epsilon_j] = c$

Then avg error $= \frac{1}{k} \sum_i \epsilon_i$

Expected sq. error

$$E\left[\frac{1}{k}\sum_i \epsilon_i\right]^2 = \frac{1}{k^2} E\left[\sum_i \left(\epsilon_i^2 + \sum_{j \neq i} \epsilon_i \epsilon_j\right)\right]$$

$$= \frac{1}{k} v + \frac{k-1}{k} c.$$

∴ if the classifiers are perfectly correlated ~~k=~~ $c = v$. then model avg doesn't help.

Perfectly un-correlated case, error $= \frac{v}{k}$

$\Rightarrow$ error $\propto \frac{1}{k}$

Bagging requires datasets sampled with replacement.

Another kind of regularization:

Parameter sharing: Other way of expressing our prior knowledge.

Creating dependencies b/w model parameters

Now, we showed, $L_2 \equiv$ MAP with Gaussian prior

$$p(w) = \frac{1}{(2\pi)^{1/2}} \exp\left(\frac{-w^2}{2}\right)$$

$$L = \prod_{i=1}^{w} p(w) = \frac{1}{(2\pi)^{w/2}} \exp\left\{\frac{-1}{2}\sum_i w_i^2\right\}$$

$\Rightarrow L_2$ is encouraging weights to be inside a circle around origin.

Now, instead of one group, ~~diff~~ weights can lie in different groups.

Eg: $p(w) = \sum_{j=1}^{M} \alpha_j \phi_j(w)$ [GMM prior].

→ known to have better ~~reg~~ effects.

Another example:

Let there be two models A & B with parameters $w^{(A)}$ & $w^{(B)}$

$$y^{(A)} = f(w^{(A)}, x) \qquad y^{(B)} = g(w^{(B)}, x)$$

86

Two different tasks with same features.

If the tasks are similarly, $w^{(A)}$ should be close to $w^{(B)}$.

New regularizer $\Omega(w^A, w^B)$.

$$= \| w^A - w^B \|_2^2.$$

One approach: Train one classifier under a supervised paradigm & close to another one trained as un-supervised.

Another powerful approach to Regularization:

Confusion is a good thing:

Noise injection: Go to page 76.

Suppose x_i's are perturbed by with a random noise. → Can show that it is similar to L_2 regularizer with L_2 defined by noise co-variance.

The Pinnacle of all regularizers:

The Dropout: ...

We have seen that Bagging always decreases the error.

However, with DNNs, averaging with many separately trained DNNs is prohibitively expensive.

Another problem with Bagging: One has to train with different subsets of data

But training many DNNs with different subsets is not possible ∵ of limited data.

Further, similar networks should share parameters.

Dropout addresses all of them.

88

Dropout: Arbitarily remove the hidden units (during training) with its i/p, o/p nodes.

Simplest case: Can be retained with a probability P.

There are 2^n possible subnetworks in a DNN.

For each presentation of a training case, a new thinned case is sampled & trained.

∴ Dropout – training a collection of 2^n thinned networks with extensive weight sharing, where each thinned network gets trained very sarely, if at all.

Testing: If a unit is retained with Pb. P, its weights are multiplied by P.
This makes the expected o/p (under dist for dropping) same for all units.

Motivation:

o/2

Motivation for dropout: Sexual reproduction
↓
½ from each parent
(still robust)

Criterion for natural selection ≠ individual fitness
but mixability of genes.

Ability of genes to work well with random set of genes makes them robust.

Another motivation: Ten successful conspiracies involving 5 people each is a better way to create havoc than one big conspiracy that requires 50 ppl all playing their parts correctly!

Formalization: Multiplication of hidden units with Bernoullie noise.

X

90

$X \in R^{N \times D}$ $y \in R^N$, $w \in \mathbb{R}^D$

$\|y - Xw\|^2$

under dropout,

X is retained with p,

which can be expressed as

$R \odot X$ where $R \in \{0,1\}^{N \times D}$

with $R_{ij} \sim$ Bernoulli (p).

$\therefore$ Obj fn: $\min_w E_{R \sim \text{Bernoulli}(p)} \left[\|y - (R \odot X)w\|^2 \right]$

$\Rightarrow \min_w \|y - pXw\|^2 + p(1-p)\|\Gamma w\|^2$

$\Gamma = \left(\text{diag}\left(X^T X\right)\right)^{1/2}$

Dropout $\equiv$ ridge with form Γ.

Γ - scales the weight cost with std dev of data.

if a ~~dim~~ data dim varies a lot, it will regularize the weight more

Another formulation.

$$\min_{w} \|y - X\tilde{w}\|^2 + \frac{1-p}{p}\|\Gamma\tilde{w}\|^2$$

$\tilde{w} = pw$, for $p = 1$, all i/p's are retained & with more dropout ($p \ll c$) regularizer to grows

---

One can also have multiplicative Gaussian noise.

$N(1,1)$ works better than Bernoulli ones.

$h_i \to h_i + h_i r$ $r \sim N(0,1)$

or $h_i r'$ where $r' \sim N(1,1)$.

in general $r' \sim N(1, \sigma^2)$ where σ is another hyperparameter.

Adv of null-noise : ReLU can't overlook noise.

Power of dropout : adaptive destruction of info is occurring at hidden units (higher level) of abstractions.

Example of multiplicative noise.
Enhancing the model capacity!!!