

Deep Learning and Its Application to Signal and Image Processing and Analysis

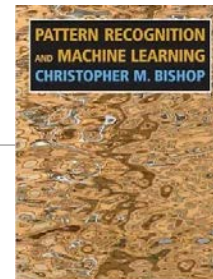
CLASS II – SPRING 2022

TAMMY RIKLIN RAVIV, ELECTRICAL AND COMPUTER ENGINEERING

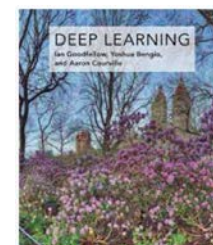
1

Today's plan

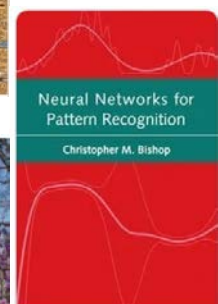
- 1) Activation functions
 - Logistic regression
 - SoftMax function
- 2) Loss functions
- 3) Optimization
- 4) Overfitting/Underfitting – Bias-Variance tradeoff
- 5) Regularization



2006



2016



1996


2

2

Topics to be covered offline

- Stochastic Gradient Descent
- Backpropagation
- Vanishing/Exploding gradients
- Optimization & Optimizers
- Training, Evaluation & Test
- Batch Normalization
- Dropout

3



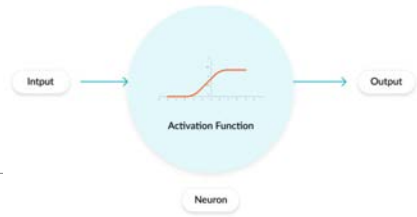
Deep Neural Network: Segmentation

Ronneberger et al 'U-Net: Convolutional networks for biomedical image segmentation', 2015

Network's performance: $w_t = w_{t-1} - \epsilon \nabla_w \mathcal{L}(w_{t-1})$

4

Activation functions



The **activation function** is a mathematical “gate” in between the input feeding the current neuron and its output going to the next layer.

Binary Step function



does not allow multi-value output

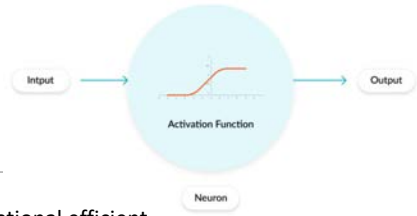
Linear function



cannot work with backpropagation (the derivative is constant)
collapses into a single layer

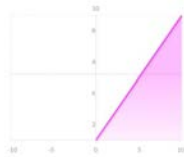
5

Activation functions



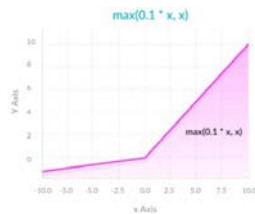
Rectified linear unit
ReLU

$$f(z) = \max(0, z)$$



Computational efficient
Non-linear
Dying ReLU problem

Leaky
ReLU



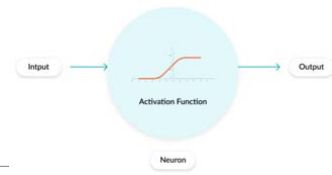
Prevents dying ReLU problem
Results are not consistent

$$f(z) = \max(\alpha z, z) \quad \alpha \in (0, 1)$$

Parametric ReLU
Allows the negative slope to be learned

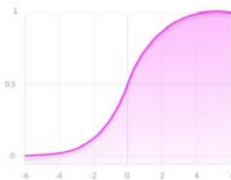
6

Activation functions



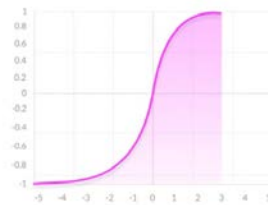
Logistic function (sigmoid)

$$\phi(\mathbf{x}) = (1 + \exp(-\mathbf{x} \cdot \mathbf{w} - b))^{-1}$$



TanH / Hyperbolic Tangent

$$\begin{aligned} \tanh(z) &= \frac{\sinh z}{\cosh z} \\ &= \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)} \end{aligned}$$



Smooth gradient

Output values bound

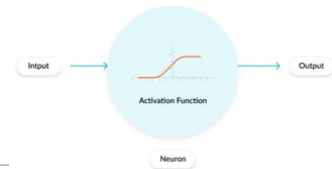
Clear predictions

Vanishing gradient

Outputs [not] zero centered.

7

Activation functions

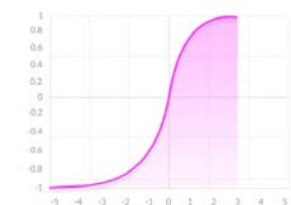
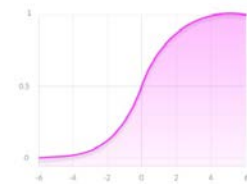


Sigmoid

$$\sigma(z) = \frac{1}{1 + \exp(-z)} = \frac{\exp(z)}{\exp(z) + 1}$$

Hyperbolic tangent

$$\begin{aligned} \tanh(z) &= \frac{\sinh z}{\cosh z} = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)} \\ &= \frac{\exp(z)}{\exp(z) + \exp(-z)} - \frac{\exp(-z)}{\exp(z) + \exp(-z)} \\ &= \frac{1}{1 + \exp(-2z)} - \frac{1}{1 + \exp(2z)} = \sigma(2z) - \sigma(-2z) \end{aligned}$$



8

Activation functions

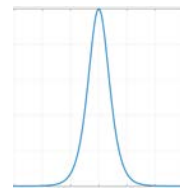
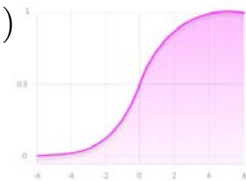
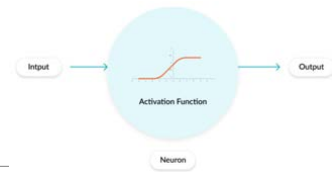
Sigmoid

$$\sigma(z) = \frac{1}{1 + \exp(-z)} = \frac{\exp(z)}{\exp(z) + 1} \quad 1 - \sigma(z) = \sigma(-z)$$

$$\sigma'(z) = \frac{d}{dz}\sigma(z) = \frac{d}{dz}(1 + \exp(-z))^{-1}$$

$$= \frac{\exp(-z)}{(1 + \exp(-z))^2} = \frac{\exp(-z)}{1 + \exp(-z)} \cdot \frac{1}{1 + \exp(-z)}$$

$$= \frac{1}{1 + \exp(z)} \cdot \frac{1}{1 + \exp(-z)} = \sigma(z)\sigma(-z)$$



9

Activation functions - Softmax

multinomial logistic regression generalizes logistic regression to a multiclass problem

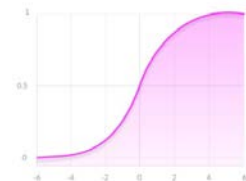
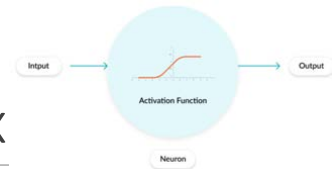
Sigmoid

$$\sigma(z) = \frac{1}{1 + \exp(-z)} = \frac{\exp(z)}{\exp(z) + 1}$$

Multinomial logistic regression function – Softmax

$$\sigma(\vec{z})_k = \frac{\exp(z_k)}{\sum_{j=1}^K \exp(z_j)} \quad \vec{z} = z_1, \dots, z_K \quad \begin{matrix} k \in \{1, 2, \dots, K\} \\ K \text{ classes} \end{matrix}$$

$$Pr(L(\mathbf{x}) = k) = \text{softmax}(k, \mathbf{w}_1 \mathbf{x}, \dots, \mathbf{w}_k \mathbf{x}, \dots, \mathbf{w}_K \mathbf{x})$$



10

The multinomial logistic function

- Multinomial logit model for K possible outcomes: $L(\mathbf{x})$
- Run $K-1$ independent binary logistic regression models - network prediction of an input \mathbf{x}
- Chose outcome K as a "pivot"
- The other $K-1$ outcomes are separately regressed against the pivot outcome

$$\ln \frac{Pr(L(\mathbf{x}) = 1)}{Pr(L(\mathbf{x}) = K)} = \mathbf{w}_1 \mathbf{x} \quad \longrightarrow \quad Pr(L(\mathbf{x}) = 1) = Pr(L(\mathbf{x}) = K) \exp(\mathbf{w}_1 \mathbf{x})$$

$$\ln \frac{Pr(L(\mathbf{x}) = 2)}{Pr(L(\mathbf{x}) = K)} = \mathbf{w}_2 \mathbf{x} \quad \longrightarrow \quad Pr(L(\mathbf{x}) = 2) = Pr(L(\mathbf{x}) = K) \exp(\mathbf{w}_2 \mathbf{x})$$

$$\ln \frac{Pr(L(\mathbf{x}) = (K-1))}{Pr(L(\mathbf{x}) = K)} = \mathbf{w}_{(K-1)} \mathbf{x} \quad \longrightarrow \quad Pr(L(\mathbf{x}) = (K-1)) = Pr(L(\mathbf{x}) = K) \exp(\mathbf{w}_{(K-1)} \mathbf{x})$$

11

The multinomial function

$$\begin{aligned} Pr(L(\mathbf{x}) = K) &= 1 - \sum_{k=1}^{K-1} Pr(L(\mathbf{x}) = k) \\ &= 1 - \sum_{k=1}^{K-1} Pr(L(\mathbf{x}) = K) \exp(\mathbf{w}_k \mathbf{x}) \end{aligned}$$

$$\longrightarrow Pr(L(\mathbf{x}) = K) = \frac{1}{1 + \sum_{j=1}^{K-1} \exp(\mathbf{w}_j \mathbf{x})}$$

$$\longrightarrow Pr(L(\mathbf{x}) = k) = \frac{\exp(\mathbf{w}_k \mathbf{x})}{1 + \sum_{j=1}^{K-1} \exp(\mathbf{w}_j \mathbf{x})}$$

12

The Softmax function - generalization

- The formulation of binary logistic regression as a log-linear model can be directly extended to multi-way regression.
- We formulate probabilities using the partition function Z :

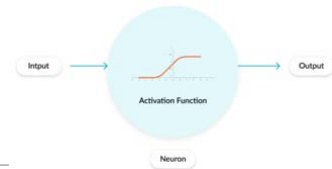
$$\sum_{k=1}^K Pr(L(\mathbf{x}) = k) = 1 \quad Pr(L(\mathbf{x}) = k) = \frac{1}{Z} \exp(\mathbf{w}_k \mathbf{x})$$

$$\sum_{k=1}^K Pr(L(\mathbf{x}) = k) = \sum_{k=1}^K \frac{1}{Z} \exp(\mathbf{w}_k \mathbf{x}) = \frac{1}{Z} \sum_{k=1}^K \exp(\mathbf{w}_k \mathbf{x}) = 1$$

$$\longrightarrow Z = \sum_{k=1}^K \exp(\mathbf{w}_k \mathbf{x}) \longrightarrow Pr(L(\mathbf{x}) = c) = \frac{\exp(\mathbf{w}_c \mathbf{x})}{\sum_{k=1}^K \exp(\mathbf{w}_k \mathbf{x})}$$

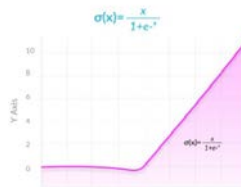
13

Activation functions- Swish



Swish: a Self-Gated Activation Function

$$f(z) = \frac{z}{1 + \exp(-z)}$$



14

Loss functions

Also termed a cost function or the objective
Measure our unhappiness with outcomes)-:

$$MSE = \frac{\sum_{i=1}^n (t_i - \hat{y}_i)^2}{n}$$

Mean squared error

$$t_i = L_{GT}(x_i)$$

$$\hat{y}_i = \hat{L}(x_i)$$

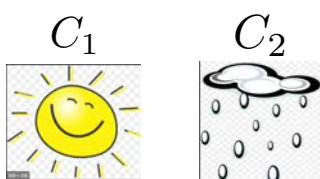
Network prediction

$$MAE = \frac{\sum_{i=1}^n |t_i - \hat{y}_i|}{n}$$

Mean absolute error

15

Loss functions: Binary Classification



$$t = 1 \text{ if } x \in C_1$$

$$t = 0 \text{ if } x \in C_2$$

$$\hat{y} = p(C_1|x)$$

$$1 - \hat{y} = p(C_2|x)$$

$$p(t|x) = p(C_1|x)^t p(C_2|x)^{(1-t)}$$

$$p(t|x) = \hat{y}^t (1 - \hat{y})^{(1-t)}$$

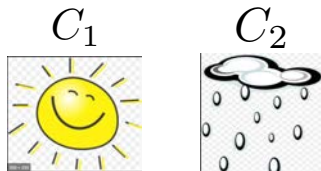
Binomial distribution – Bernoulli distribution

The likelihood of observing the training data set, assuming the data points are drawn independently from this distribution, is then given by:

$$\prod_i \hat{y}_i^{(t_i)} (1 - \hat{y}_i)^{(1-t_i)}$$

16

Loss functions: Binary Classification



$t = 1$ if $x \in C_1$
 $t = 0$ if $x \in C_2$

$\hat{y} = p(C_1|x)$
 $1 - \hat{y} = p(C_2|x)$

$$\prod_i \hat{y}_i^{(t_i)} (1 - \hat{y}_i)^{(1-t_i)}$$



$$BCE = - \sum_i t_i \ln \hat{y}_i + (1 - t_i) \ln(1 - \hat{y}_i)$$

Binary Cross Entropy function

minimize the dissimilarity between the empirical distribution defined by the training set and the model distribution

17

Information, Entropy and Cross Entropy

Information: quantifies the number of bits required to encode and transmit an event x

$$h(P) = - \log P(x)$$

Entropy: quantifies the number of bits required to transmit a randomly selected event from a probability distribution:

“Degree of surprise”

$$H(P) = - \sum_{x \in X} P(x) \log P(x)$$

Cross entropy: quantifies the average number of bits needed to encode data coming from a source with distribution P when we use model Q:

$$H(P, Q) = - \sum_{x \in X} P(x) \log Q(x)$$

18

Cross Entropy & KL Divergence

Kullback–Leibler (KL) divergence is a measure (statistical distance) of how a probability distribution Q is different from a reference probability distribution P . For discrete probability distributions it is defined by:

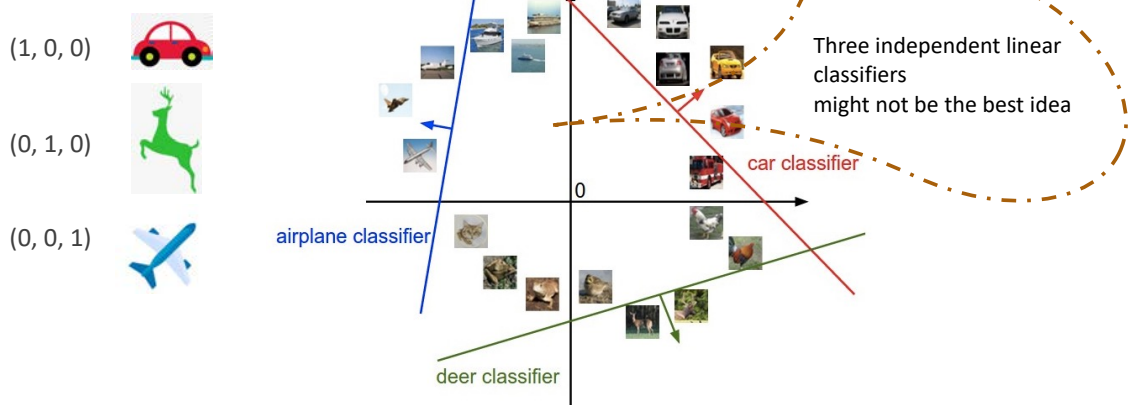
$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right) = - \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{Q(x)}{P(x)} \right)$$

KL divergence calculates the relative entropy between two probability distributions.

Cross-entropy can be thought to calculate the total entropy between the distributions.




19

Categorical cross entropy loss



20

Categorical cross entropy loss

(1, 0, 0)		$p(t_i x_i) = \prod_{k=1}^K (\hat{y}_i)_k^{(t_i)_k}$ <p style="color: red;">Categorical distribution</p>
(0, 1, 0)		
(0, 0, 1)		

$$(\hat{y}_i)_k = Pr(L(\mathbf{x}_i) = k)$$

$$CCE = -\sum_i \sum_{k=1}^K (t_i)_k \ln(\hat{y}_i)_k$$

21

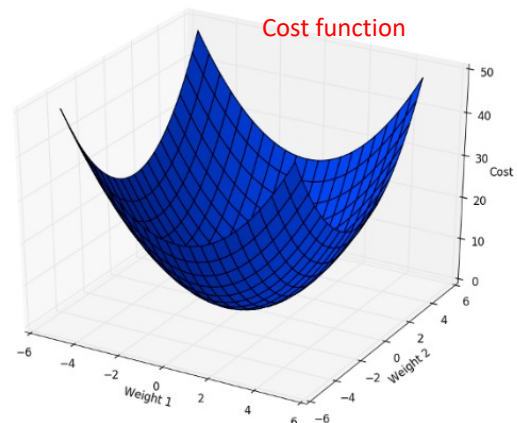
Optimization

Cost function values: the discrepancies between the outputs (NN estimations) and the training set data points.

Goal : find the set of weights for which a global minimum is obtained

the cost function is parameterized by the network's weights — we control our loss function by changing the weights.

In reality the cost function is not convex.



25

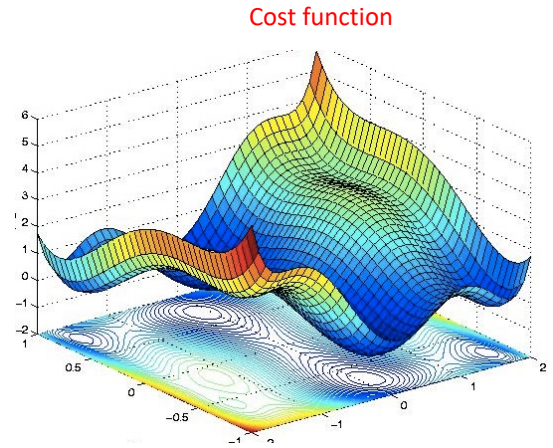
Optimization

Cost function values: the discrepancies between the outputs (NN estimations) and the training set data points.

Goal : find the set of weights for which a global minimum is obtained

the cost function is parameterized by the network's weights — we control our loss function by changing the weights.

In reality the cost function is not convex.



26

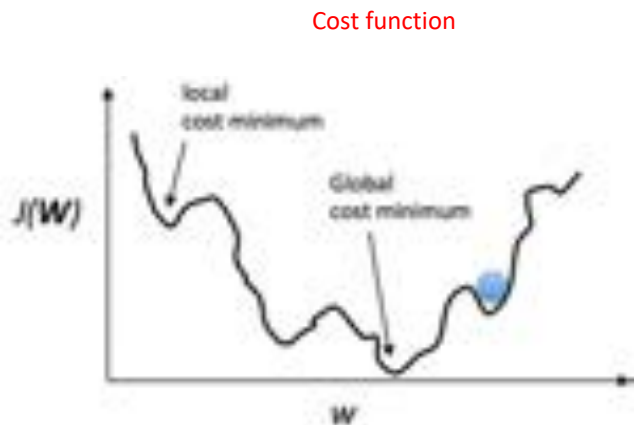
Optimization

Cost function values: the discrepancies between the outputs (NN estimations) and the training set data points.

Goal : find the set of weights for which a global minimum is obtained

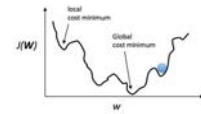
the cost function is parameterized by the network's weights — we control our loss function by changing the weights.

In reality the cost function is not convex.



27

Optimization –



Why is it difficult?

1. There is no simple equation that can be solved analytically
2. High-dimensional function
3. Function might have many local minima & maxima

Common approach:

Iterative optimization algorithms, e.g. gradient descent

28

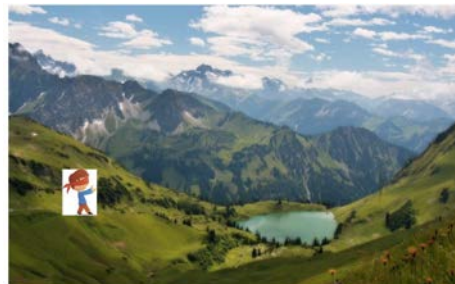
Gradients

1D

Numerical evaluation

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

- approximate
- very slow to evaluate



nD

$$\nabla f = \frac{\partial f}{\partial x_1} \mathbf{e}_1 + \dots + \frac{\partial f}{\partial x_n} \mathbf{e}_n$$

29

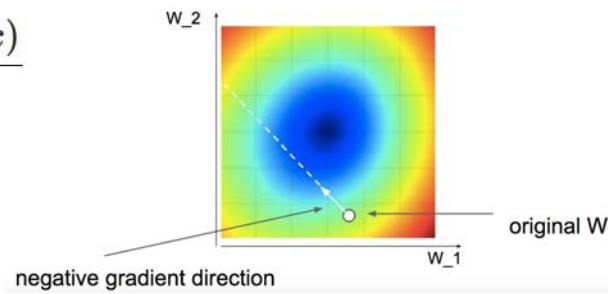
Gradients

1D

Numerical evaluation

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

- approximate
- very slow to evaluate



nD

$$\nabla f = \frac{\partial f}{\partial x_1} \mathbf{e}_1 + \cdots + \frac{\partial f}{\partial x_n} \mathbf{e}_n$$

30

Numerical vs. analytic gradient

- Numerical gradient: approximate, slow, easy to write
- Analytic gradient: exact, fast, error-prone


In practice: Always use analytic gradient, but check implementation with numerical gradient. This is called a **gradient check**.

31

Gradient descent

$$W := W - \alpha \frac{\partial L(W)}{\partial W} \quad \text{Update rule}$$

$$W := W - \alpha \nabla_W L(W)$$


 Learning rate

Learning rate: An important **hyperparameter**

too small – very slow convergence or gets stuck in local minima

Too Big – may “skip” the target minimum; may go in the wrong direction

32

Optimization and Generalization

Standard optimization – Find a model that will fit the known data

$$\frac{1}{m^{(\text{train})}} \|\mathbf{X}^{(\text{train})} \mathbf{w} - \mathbf{y}^{(\text{train})}\|_2^2$$

Machine learning - Find a model that will fit the unseen data – Generalize

$$\frac{1}{m^{(\text{test})}} \|\mathbf{X}^{(\text{test})} \mathbf{w} - \mathbf{y}^{(\text{test})}\|_2^2$$

33

Generalization

Generalization- main challenge of machine learning algorithms
perform well on new, previously unseen inputs.

i.i.d. assumption: The train dataset and the test dataset are **independent** of each other and each dataset is **identically distributed**, drawn from the same probability distribution as each other.

In theory: the expected training error of a randomly selected model should be equal to the expected test error of that model.

In practice: since we set the parameters based on the training and then use the test – the test error is higher.

34

Capacity, overfitting and underfitting

Our aims: (1) make training error small (2) make test error as small as the training.

Underfitting occurs when the model is not able to obtain a sufficiently low error value on the training set.

Overfitting occurs when the gap between the training error and test error is too large.

We can control whether a model is more likely to overfit or underfit by altering its **capacity**.

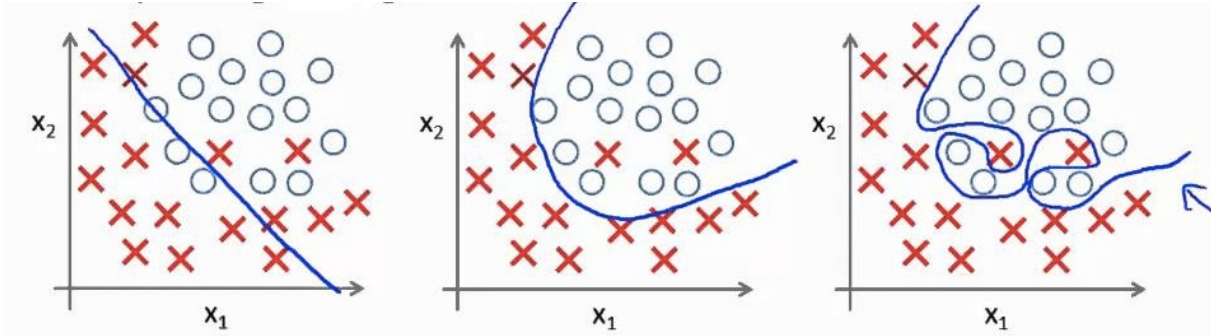
Informally, a model's capacity is its ability to fit a wide variety of functions.

35

Classification example

Back to machine learning ...

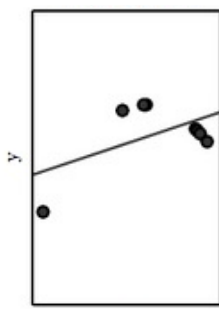
Why we don't like it?



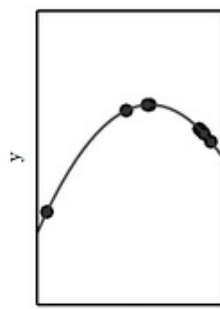
Andrew NG

36

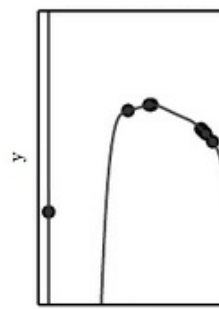
Regression example



Linear



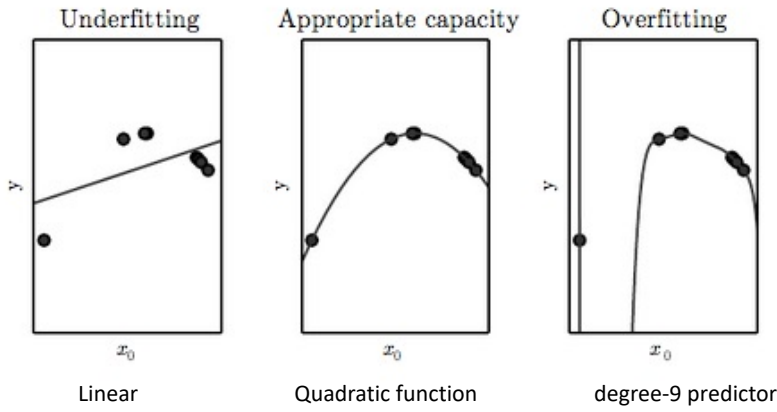
Quadratic function



degree-9 predictor

37

Regression example

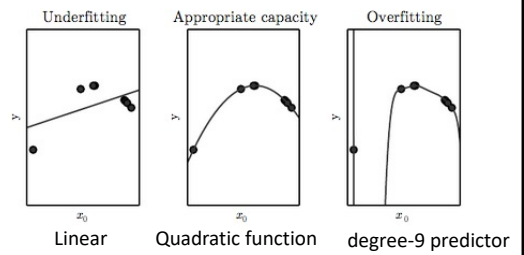


38

Capacity – an observation

finding the best function within this family is a very difficult optimization problem

In practice, the learning algorithm does not actually find the best function, but merely one that significantly reduces the training error.



39

Occam's Razor (low of parsimony)

Among competing hypotheses, the one with the fewest assumptions should be selected.

Other things being equal, simpler explanations are generally better than more complex ones.



40

Occam's Razor (low of parsimony)

Among competing hypotheses, the one with the fewest assumptions should be selected.

Other things being equal, simpler explanations are generally better than more complex ones.

Everything should be made as simple as possible, but not simpler
Albert Einstein



41

The Bias-Variance Tradeoff

The true function we want to approximate: $f = f(\mathbf{x})$

The dataset for training: $D = \{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \dots, (\mathbf{x}_N, t_N)\}$

where $t = f + \epsilon$ and $E(\epsilon) = 0$

Given D , we train an arbitrary neural network to approximate f by

$$y = g(\mathbf{x}, \mathbf{w})$$

<http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf>

42

The Bias-Variance Tradeoff

The mean-squared error of this networks is:
$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (t_i - y_i)^2$$

To assess the effectiveness of the network, we want to know the expectation of the MSE if we test the network on arbitrarily many test points drawn from the unknown function.

$$E\{\text{MSE}\} = E\left\{\frac{1}{N} \sum_{i=1}^N (t_i - y_i)^2\right\}$$

<http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf>

43

The Bias-Variance Tradeoff

$$\begin{aligned}
 E\{(t_i - y_i)^2\} &= E\{(t_i - f_i + f_i - y_i)^2\} \\
 &= E\{(t_i - f_i)^2\} + E\{(f_i - y_i)^2\} + 2E\{(f_i - y_i)(t_i - f_i)\} \\
 &= E\{\varepsilon^2\} + E\{(f_i - y_i)^2\} + 2(E\{f_i t_i\} - E\{f_i^2\} - E\{y_i t_i\} + E\{y_i f_i\})
 \end{aligned}$$

Note: $E\{f_i t_i\} = f_i^2$ since f is deterministic and $E\{t_i\} = f_i$

: $E\{f_i^2\} = f_i^2$ since f is deterministic

: $E\{y_i t_i\} = E\{y_i(f_i + \varepsilon)\} = E\{y_i f_i + y_i \varepsilon\} = E\{y_i f_i\} + 0$

$$E\{(t_i - y_i)^2\} = E\{\varepsilon^2\} + E\{(f_i - y_i)^2\}$$

44

The Bias-Variance Tradeoff

$$E\{(t_i - y_i)^2\} = E\{\varepsilon^2\} + E\{(f_i - y_i)^2\}$$

The MSE can be decomposed in expectation into the variance of the noise and the MSE between the true function and the predicted values.

<http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf>

45

The Bias-Variance Tradeoff

$$\begin{aligned}
 E\{(t_i - y_i)^2\} &= E\{\epsilon^2\} + E\{(f_i - y_i)^2\} \\
 E\{(f_i - y_i)^2\} &= E\{(f_i - E\{y_i\} + E\{y_i\} - y_i)^2\} \\
 &= E\{(f_i - E\{y_i\})^2\} + E\{(E\{y_i\} - y_i)^2\} + 2E\{(E\{y_i\} - y_i)(f_i - E\{y_i\})\} \\
 &= \text{bias}^2 + \text{Var}\{y_i\} + 2\{E\{f_i E\{y_i\}\} - E\{E\{y_i\}^2\} - E\{y_i f\}_i + E\{y_i E\{y_i\}\}
 \end{aligned}$$

Note: $E\{f_i E\{y_i\}\} = f_i E\{y_i\}$ since f is deterministic and $E\{E\{z\}\} = z$

$$: E\{E\{y_i\}^2\} = E\{y_i\}^2 \text{ since } E\{E\{z\}\} = z$$

$$: E\{y_i f\}_i = f_i E\{y_i\}$$

$$: E\{y_i E\{y_i\}\} = E\{y_i\}^2$$

46

The Bias-Variance Tradeoff

$$\begin{aligned}
 E\{(t_i - y_i)^2\} &= E\{\epsilon^2\} + E\{(f_i - y_i)^2\} \\
 E\{(f_i - y_i)^2\} &= E\{(f_i - E\{y_i\} + E\{y_i\} - y_i)^2\} \\
 &= E\{(f_i - E\{y_i\})^2\} + E\{(E\{y_i\} - y_i)^2\} + 2E\{(E\{y_i\} - y_i)(f_i - E\{y_i\})\} \\
 &= \text{bias}^2 + \text{Var}\{y_i\} + 2\{E\{f_i E\{y_i\}\} - E\{E\{y_i\}^2\} - E\{y_i f\}_i + E\{y_i E\{y_i\}\}
 \end{aligned}$$

$$E\{(f_i - y_i)^2\} = \text{bias}^2 + \text{Var}\{y_i\}$$

$$E\{(t_i - y_i)^2\} = \text{Var}\{\text{noise}\} + \text{bias}^2 + \text{Var}\{y_i\}$$

47

The Bias-Variance Tradeoff

$$E\{(t_i - y_i)^2\} = \text{Var}\{\text{noise}\} + \text{bias}^2 + \text{Var}\{y_i\}$$

Note that the variance of the noise can not be minimized; it is independent of the neural network.

Thus in order to minimize the MSE, we need to minimize both the bias and the variance.

However, this is not trivial –

e.g. if we set the output to be constant – the variance will be zero but the bias $E\{(f_i - E\{y_i\})^2\}$ will be high.

Alternatively, we could train the network to predict the training – the bias will be zero but the variance $E\{(E\{y_i\} - y_i)^2\}$ will be equal to the variance of the noise.

48

Bias-Variance tradeoffs

Bias: how much the average model over all training sets differs from the true model.

- Error due to inaccurate assumptions/simplifications made by the model.

$$E\{(f_i - E\{y_i\})^2\}$$

Variance: how much models estimated from different training sets differ from each other.

$$E\{(E\{y_i\} - y_i)^2\}$$

Slide credit: L. Lazebnik

49

Bias-Variance Trade-off

$$E(\text{MSE}) = \text{noise}^2 + \text{bias}^2 + \text{variance}$$

Unavoidable error

Error due to incorrect assumptions

Error due to variance of training samples

<http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf>

Slide credit: D. Hoiem

50

No Free Lunch Theorem

© Original Artist
Reproduction rights obtainable from
www.CartoonStock.com



Slide credit: D. Hoiem

51

No free lunch theorem

Averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.

52

No free lunch theorem

Averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.

This means that the goal of machine learning research is not to seek a universal learning algorithm or the absolute best learning algorithm. Instead, our goal is to understand what kinds of distributions are relevant to the “real world” that an AI agent experiences, and what kinds of machine learning algorithms perform well on data drawn from the kinds of data generating distributions we care about.

53

Underfitting and Overfitting

Underfitting: model is too “simple” to represent all the relevant class characteristics

- High bias (few degrees of freedom) and low variance
- High training error and high test error

Overfitting: model is too “complex” and fits irrelevant characteristics (noise) in the data

- Low bias (many degrees of freedom) and high variance
- Low training error and high test error

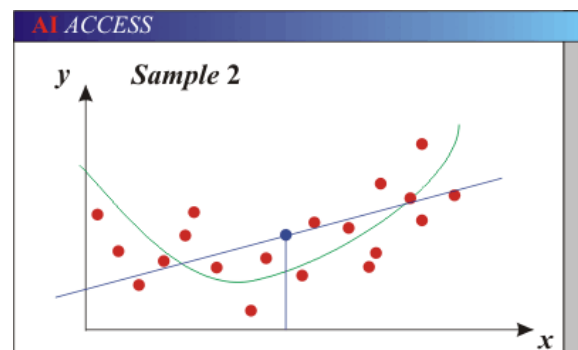
Slide credit: L. Lazebnik

54

Generalization Error Effects

Underfitting: model is too “simple” to represent all the relevant class characteristics

- High bias (few degrees of freedom) and low variance
- High training error and high test error



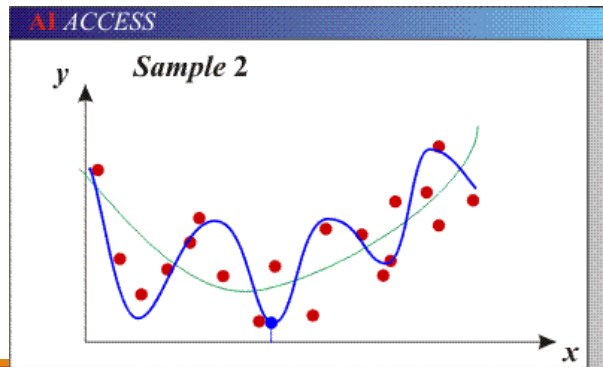
Slide credit: L. Lazebnik

55

Generalization Error Effects

Overfitting: model is too “complex” and fits irrelevant characteristics (noise) in the data

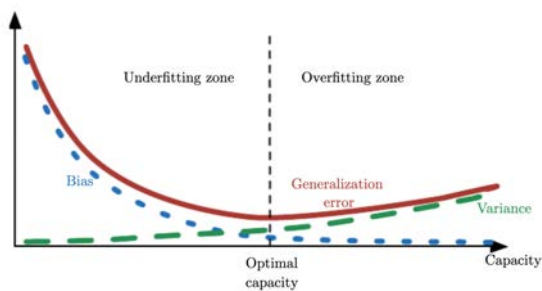
- Low bias (many degrees of freedom) and high variance
- Low training error and high test error



Slide credit: L. Lazebnik

56

Bias-Variance Trade-off



Models with too few parameters are inaccurate because of a large bias.

- Not enough flexibility!

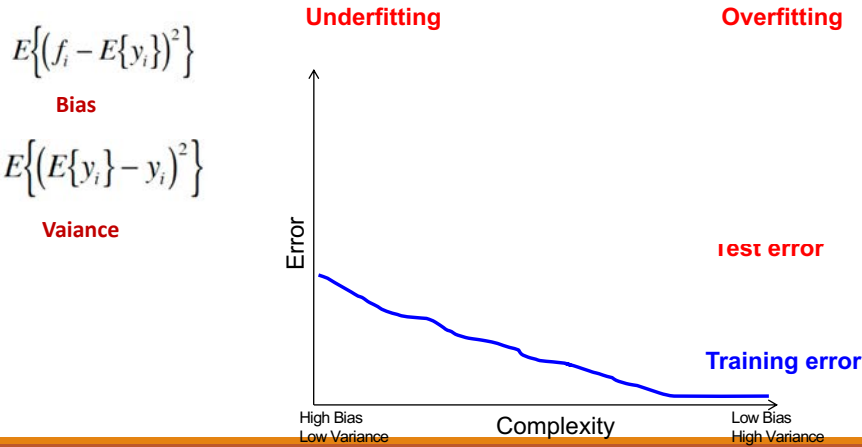
Models with too many parameters are inaccurate because of a large variance.

- Too much sensitivity to the sample.

Slide credit: D. Hoiem

57

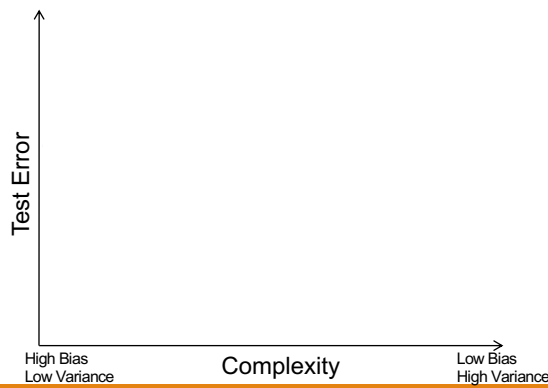
Bias-variance tradeoff



Slide credit: D. Hoiem

58

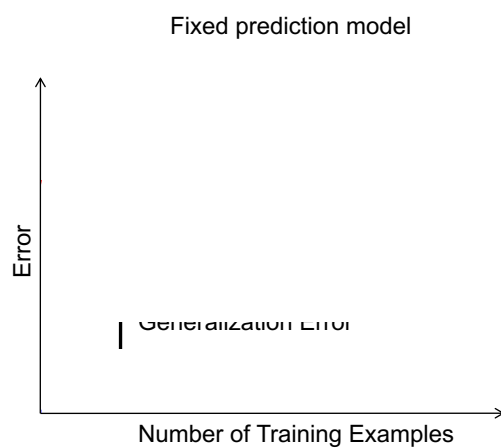
Bias-variance tradeoff



Slide credit: D. Hoiem

59

Effect of Training Size



Slide credit: D. Hoiem

60

How to reduce variance?

Choose a simpler classifier

Regularize the parameters

Get more training data

Slide credit: D. Hoiem

61

Capacity & VC dimension

Vapnik-Chervonenkis (VC) dimension is a measure of the capacity of a space of functions that can be learned by a statistical classification algorithm.

It is defined as the cardinality of the largest set of points that the algorithm can shatter.

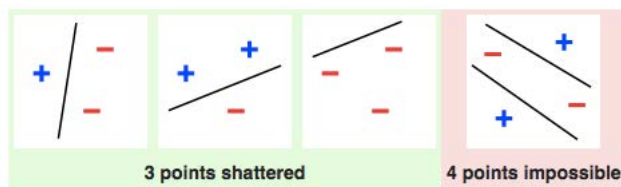
(Vapnik and Chervonenkis, 1971; Vapnik, 1982; Blumer et al., 1989; Vapnik, 1995)

62

Capacity & VC dimension

Vapnik-Chervonenkis (VC) dimension is a measure of the capacity of a space of functions that can be learned by a statistical classification algorithm.

It is defined as the cardinality of the largest set of points that the algorithm can shatter.

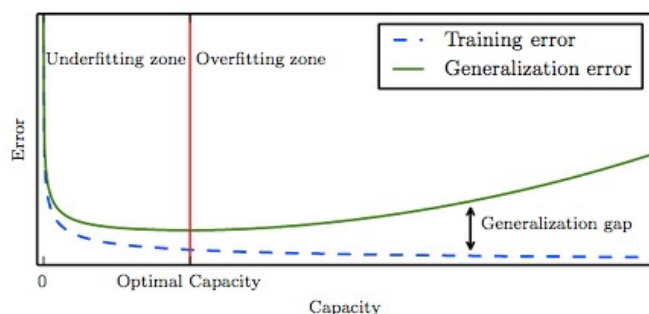


any 3 points that are not collinear can be shattered by a linear classifier (perceptron).
not all set of 4 points can be shattered

(Vapnik and Chervonenkis, 1971; Vapnik, 1982; Blumer et al., 1989; Vapnik, 1995)

63

Capacity and generalization error



The discrepancy between training error and generalization error is bounded from above by a quantity that grows as the model capacity grows but shrinks as the number of training examples increases.

(Vapnik and Chervonenkis, 1971; Vapnik, 1982; Blumer et al., 1989; Vapnik, 1995).

$$\Pr \left(\text{test error} \leq \text{training error} + \sqrt{\frac{1}{N} \left[D \left(\log \left(\frac{2N}{D} \right) + 1 \right) - \log \left(\frac{\eta}{4} \right) \right]} \right) = 1 - \eta,$$

Valid when $D - \text{VC dimension}$ is much smaller than N training examples

64

Generalization – another philosophical note

Can any Machine Learning algorithm generalize well from a finite training set of examples?

To logically infer a rule describing every member of a set,
One must have information about every member of that set.

66

Generalization – a philosophical note

Can any Machine Learning algorithm generalize well from a finite training set of examples?

Machine learning promises to find rules that are probably correct about most members of the set they concern.

67

Preference and Regularization

The no free lunch theorem implies that we must design our machine learning algorithms to perform well on a specific task. We do so by building a set of preferences into the learning algorithm. When these preferences are aligned with the learning problems we ask the algorithm to solve, it performs better.

Specifically, we can give a learning algorithm a preference for one solution in its hypothesis space to another. This means that both functions are eligible, but one is preferred. The unpreferred solution will be chosen only if it fits the training data significantly better than the preferred solution.

68

Regularization

Reduce Test error on the expense of increasing Training error.



Generalization

Many forms of regularization:

Adding extra constraints on the model

Adding extra constraints to the objective function

Encoding prior knowledge

Express preference to a simpler model

69

Regularization for Deep Learning

Regularization estimators:

Trading increased bias for reduced variance.

An effective regularizer is one that makes a profitable trade, reducing variance significantly while not overly increasing the bias.

Three regimes – where the model family being trained either

excluded the true data generating process—corresponding to underfitting and inducing bias

matched the true data generating process

included the generating process but also many other possible generating processes—the overfitting regime where variance rather than bias dominates the estimation error

70

Regularization for Deep Learning

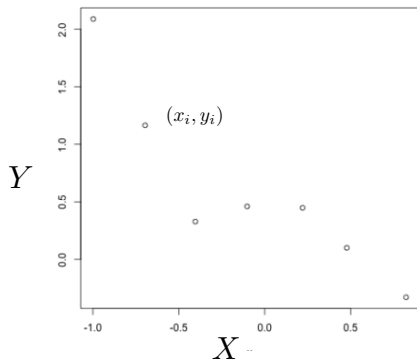
- L1 – regularization
- L2 – regularization
- Early stopping
- Bagging
- Dropout

Main resource:
 Deep learning book, chapter 7
 Goodfellow and Bengio and Courville
 MIT Press, 2016

71

Regularization

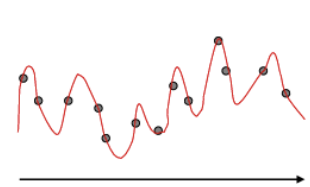
Refers to a process of introducing additional information in order to solve an ill-posed problem or to prevent overfitting.



Data $\{x_i, y_i\}$

Objective $J = \min_f \sum_i L(f(x_i), y_i)$

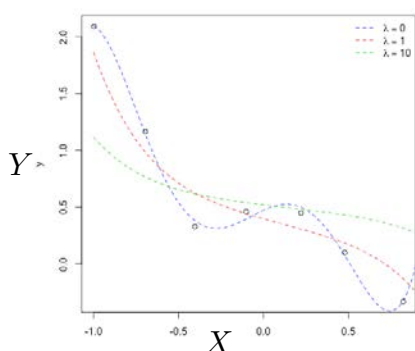
↑
Loss



72

Regularization - by norm penalty function

Refers to a process of introducing additional information in order to solve an ill-posed problem or to prevent overfitting.

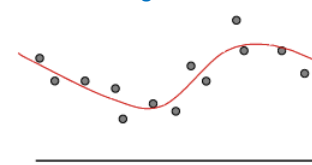


Data $\{x_i, y_i\}$

$$\text{Objective } J = \min_f \sum_i L(f(x_i), y_i) + \lambda R(f)$$

Loss

Regularization



73

Regularization –parameter norm penalties

Regularization can be explicit or implicit

Explicit: Limit the model capacity by adding a parameter norm penalty to the

objective function: $\tilde{J}(f; X, y) = J(f; X, y) + \lambda R(f)$

where $\lambda \in [0, \infty]$, is a hyperparameter that weights the relative contribution of the norm penalty term, R , relative to the standard objective function J .

In NN we usually choose R that penalizes only the weights and leaves the biases unregularized. R may be different for each layer.

74

L^2 Parameter Regularization

$$R(f) = \frac{1}{2} \|w\|_2^2$$

Commonly use

Drive the weights closer to the origin

Known as weight decay, ridge regression or Tikhonov regularization.

75

L^2 Parameter Regularization

$$R(f) = \frac{1}{2} \|w\|_2^2$$

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

with the corresponding parameter gradient

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

To take a single gradient step to update the weights, we perform this update

$$w \leftarrow w - \epsilon(\lambda w + \nabla_w J(w; X, y))$$

Written another way, the update is:

$$w \leftarrow w(1 - \epsilon\lambda) + \epsilon \nabla_w J(w; X, y)$$

Shrink

76

L^1 Parameter regularization

Formally, L^1 regularization on the model parameter w is defined as:

$$R(f) = \|w\|_1 = \sum_i |w_i|$$

The regularized objective function is given by:

$$\tilde{J}(w; X, y) = \lambda \|w\|_1 + J(w; X, y)$$

with the corresponding gradient:

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

77

L^1 and Sparsity

L^1 encourages sparsity

Sparsity in this context refers to the fact that some parameters have an optimal value of zero.

→ Feature selection mechanism

78

Bayesian point of view

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

likelihood

↓

Posterior probability ←

→ Prior



many regularization techniques correspond to imposing certain prior distributions on model parameters.

85

Bayesian point of view

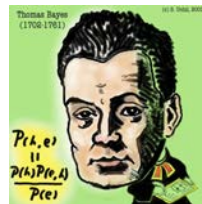
$$P(Y|X) \propto P(X|Y)P(Y)$$

likelihood

↓

Posterior probability ←

→ Prior



many regularization techniques correspond to imposing certain prior distributions on model parameters.

86

Early stopping

Early stopping can be viewed as regularization in time. Intuitively, a training procedure like gradient descent will tend to learn more and more complex functions as the number of iterations increases. By regularizing on time, the complexity of the model can be controlled, improving generalization.

In practice, early stopping is implemented by training on a training set and measuring accuracy on a statistically independent validation set. The model is trained until performance on the validation set no longer improves. The model is then tested on a testing set.

87

Bagging and Other Ensemble Methods

Bagging (bootstrap aggregating) is a technique for reducing generalization error by combining several models (Breiman, 1994). The idea is to train several different models separately, then have all of the models vote on the output for test examples.

Models using *averaging techniques* are called *ensemble methods*

The reason that model averaging works is that different models will usually not make all the same errors on the test set.

On average, the ensemble will perform at least as well as any of its members, and if the members make independent errors, the ensemble will perform significantly better than its members

88