Deep Learning: A Statistical Perspective

Myunghee Cho Paik
Guest lectures by Gisoo Kim, Yongchan Kwon, Young-geun Kim, Wonyoung Kim and Youngwon Choi

Seoul National University

March-June, 2018

Introduction

In 2009

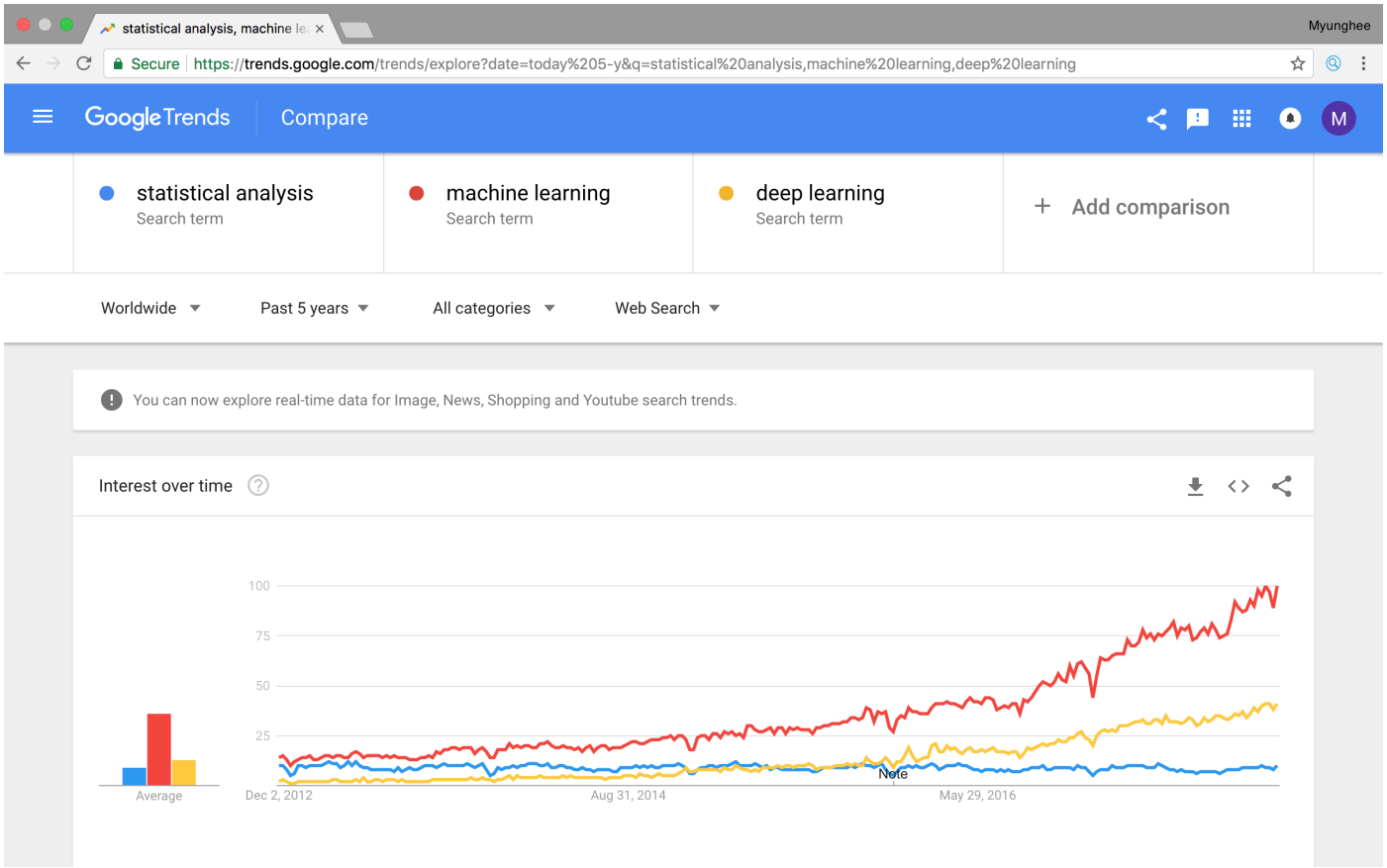
The New York Times

August 5, 2009

For Today's Graduates, Just One Word: Statistics

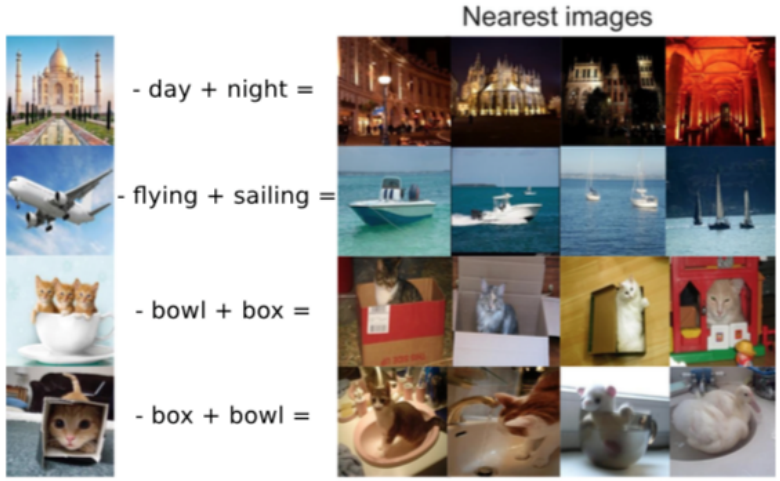
"....the sexy job in the next 10 years will be statisticians"

Attractive component of statistics



Deep impact

- Computer vision, speech recognition, natural language processing..
- Alpha go, Face recognition, autonomous driving, AI assistant...
- Combining domains...



Figures from Kiros, Salakhutdinov, and Zemel (2014)

BBC | News | Sport | Weather | Shop | Earth | Travel | More | Search

NEWS

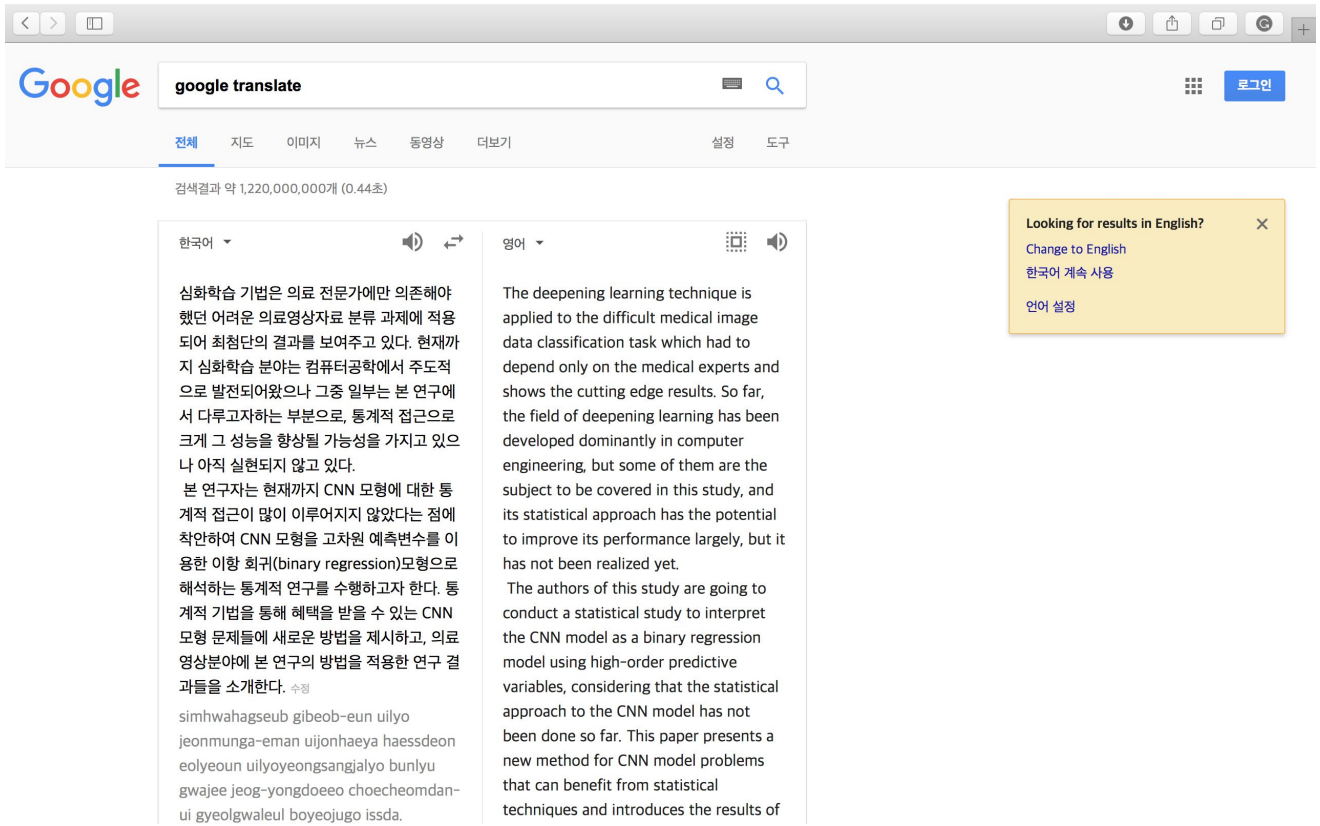
Home | Video | World | Asia | UK | Business | Tech | Science | Stories | Entertainment & Arts | More

Amazon opens a supermarket with no checkouts

Customers enter the store by scanning the Amazon Go app on their smartphone

Deep impact on education

- Medical school: Hinton said: "People should stop training radiologists now, it's just completely obvious that in five years deep learning is going to do better than radiologists, it might be ten years".
- Should we learn foreign languages?



Deep impact continues

- After training Latex files of an algebraic geometry textbook

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the compariciely in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ *(by the formal open covering* X *and a single map* $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ *over* U *compatible with the complex*

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ *is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If* T *is surjective we may assume that* T *is connected with residue fields of* S*. Moreover there exists a closed subspace* $Z \subset X$ *of* X *where* U *in* X' *is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem*

(1) f *is locally of finite type. Since* $S = \mathrm{Spec}(R)$ *and* $Y = \mathrm{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\ldots,0}$.

Lemma 0.2. *Let* X *be a locally Noetherian scheme over* S, $E = \mathcal{F}_{X/S}$. *Set* $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. *Since* $\mathcal{I}^n \subset \mathcal{I}^n$ *are nonzero over* $i_0 \leq \mathfrak{p}$ *is a subset of* $\mathcal{J}_{n,0} \circ \overline{A}_2$ *works.*

Lemma 0.3. *In Situation ??. Hence we may assume* $\mathfrak{q}' = 0$.

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

- Disruption continues

Deep learning and statistical community

- In a nutshell, a deep neural network model has a form:

$$E(y|x;\theta) = f_k(\cdots f_3(f_2(f_1(x;\theta_1);\theta_2);\theta_3)\cdots;\theta_k).$$

- Is deep learning statistics?
- Should it be taught in the department of statistics?
- Whether it is statistics or not, deep learning is changing the way of living and this disruption is likely to continue.

Deep learning: Theory vs. Practice

- 'Deep deep trouble' by Michael Elad https://sinews.siam.org/Details-Page/deep-deep-trouble-4
- At NIPS 2017, Ali Rahimi's speech: Deep learning is alchemy.
- Le Cun's facebook response: 'In the history of science and technology, the engineering artifacts have almost always preceded the theoretical understanding: the lens and the telescope preceded optic theory, the steam engine preceded thermodynamics....., the computer preceded computer science. Why? Because theorists will spontaneously study 'simple' phenomena, and will not be enticed to study complex one until there a practical importance to it'
- Donoho in his lecture: Polarization of the rationalists (Decartes, Spinoza, Leibniz) vs. the empiricalists (Locke, Berkeley, Hume). Calls for engagement of theorist in experimental studies, urges hibridization of theory and empirical science to feed each other.

Course

- We focus on deep learning that can be viewed as highdimensional compositional modeling. (We do not cover deep reinforcement learning)
- We do not cover computational issues such as parallel computing.
- Intro to python and frameworks (pytorch and tensorflow) will be provided to help get started.

Course

- The goal of the course is to (i) learn deep neural network (DNN) and (ii) solve a real-life problem including implementation (iii) gain understanding and explanation why some of deep learning techniques work.
- Evaluation will be based on final projects. No written exams.
- Final projects could include (i) implementation of a real data example (ii) numerical or theoretical experiments to gain explanations in unknown aspects of architecture or optimization (iii) critical literature review of a topic.
- One- or two-paged written proposal will be submitted presented to evaluate relevance to the course.
- Final exam constitutes a publication-worthy paper and presentation with explicit explanation of contribution of each member in a team.

Contents

- Introduction
- Pre-deep learning: Separate Feature extraction and Discrimination
- Components of well-established machine learning tools (SVM, RKHS, model complexity)
- NN, multi-layer perceptron, backprop
- CNN
- Optimization and regularization
- Visualization/Understanding
- Python/Deep learning framework
- RNN
- Generative models: Variational autoencoder, Generative adversarial network
- Detection/Segmentation/Natural Language Processing

Resources

- Lectures: Stanford: CS231: Computer vision; CS224: Natural Language Processing
  STAT385 "Theories of Deep Learning" (https://stats385.github.io/) HKUST "Deep Learning: Towards Deeper Understanding" (https://deeplearning-math.github.io/) Berkeley SAT212b "Topics course on Deep Learning" (http://joanbruna.github.io/stat212b/)
- Book: Deep Learning by Ian Goodfellow and Yoshua Bengio and Aaron Courville. www.deeplearningbook.org/
- Data: Name (input dimension, training, test data seize): MNIST (28x28, 60K, 10K), CIFAR10 (32x32, 50K, 10K), CIFAR100, ImageNet (256x256 after preprocessing, 14million)

A Brief History of Deep Learning

What triggered 'hype' of deep learning

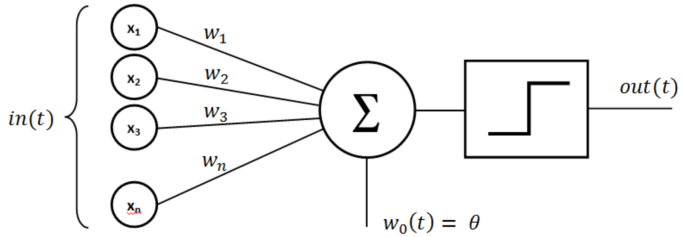
- It will take a story to explain 'what' but it is simple to pinpoint 'when': Turning point is ImageNet competition 2012.
- Some people say deep learning made 'paradigm shift'.
- A break-through model? no. New theory? no. Then what?

Biology of vision

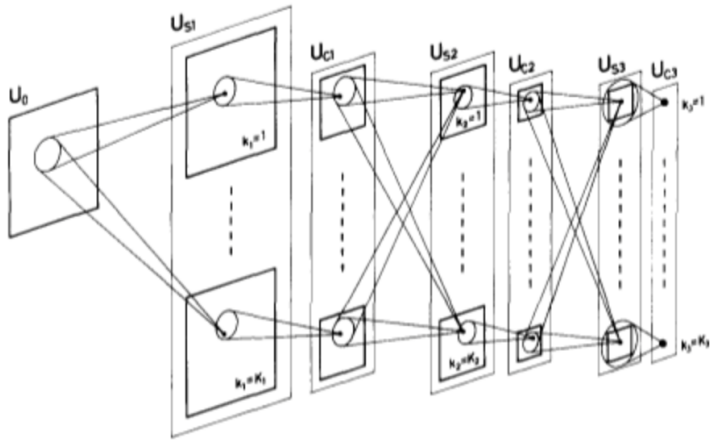
- Hubel and Wiesel (1959) inserted a microelectrode into the primary visual cortex of an anesthetized cat and found that some neurons fired rapidly when presented with lines at one angle, while others responded best to another angle. Some of these neurons responded to light patterns and dark patterns differently. Hubel and Wiesel called these neurons simple cells. Still other neurons, which they termed complex cells, detected edges and could preferentially detect motion in certain directions.

History: Emergence

- 1962 Frank Rosenblatt, 1969 Minsky & Papert: Perceptrons

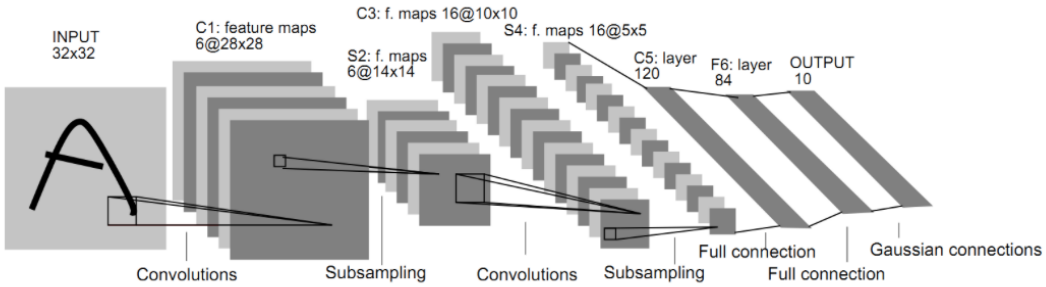


- 1980 Fukushima proposed Neurocognitron, a hierarchical, multilayered artificial neural network



History: Emergence

- 1986 Rumelhart, Hinton, & Williams, Back propagation
- 1988 LeCun, Bottou, Bengio, Haffner: LeNet-5: A deep neural network with stochastic gradient descent by Robins and Monroe (1951)



History: Decline and resurgence

- 1990-2005: Bayesian approaches for neural nets were attempted. Support Vector Machines emerged. Except handful of researchers like Hinton, LeCun, Bengio, people left neural nets. No presence in Neural Information Processing Systems (NIPS).
- 2005-2010: Attempts to resurrect neural nets with unsupervised pretraining, alternative learning method
- 2012-present Most of the alternative techniques discarded and neural nets in 1980 are revived with more data, computing power and some tricks under the new brand 'deep learning'

Massive data since 2010

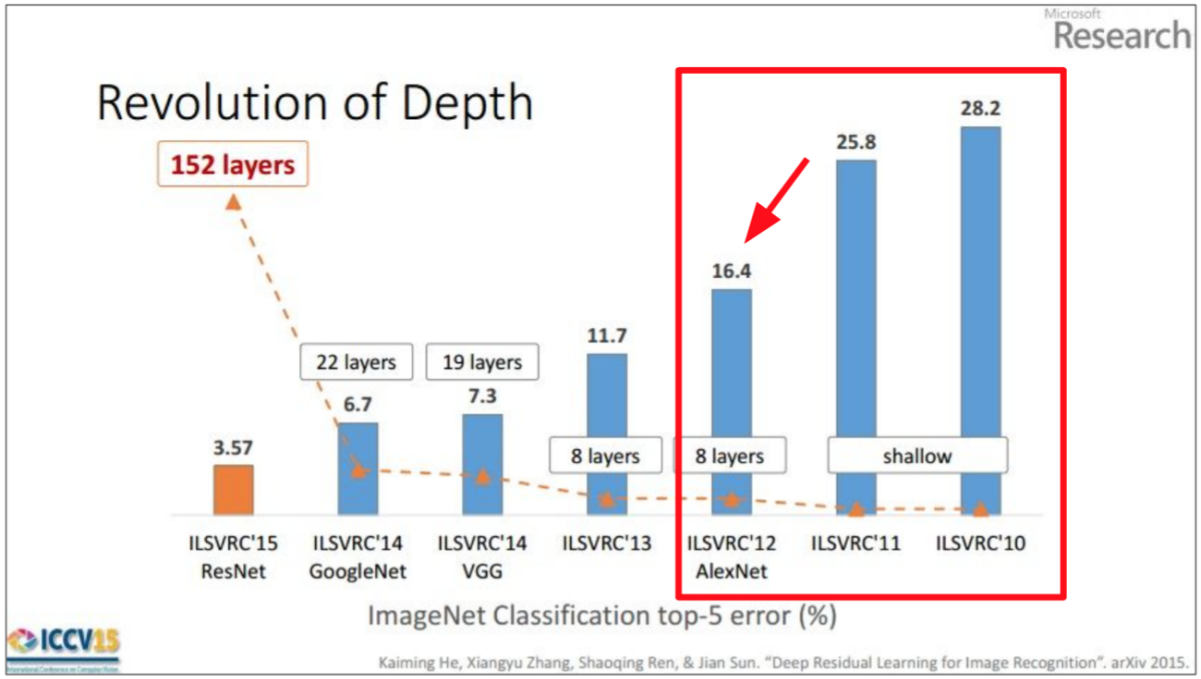
- Emergence of smartphone photos
- Generation of huge data by smartphone, sensors
- Cloud storage of data and photos
- Fei-Fei Li organizes ImageNet: 14 million+ labeled images, 21,000+ classes; Labeling took 1+ year of human labor. Starts ImageNet competition.

Computing power

- Popularity of game and emergence of GPU propelled computational power

Decades old neural network meets massive data and computing power

- Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton win the ImageNet 2012 by a big margin.
- Since then, deep learning reign has begun.



(slide from Kaiming He's recent presentation)

Practical success without theoretical understanding

Deep learning culture and practical success

- Common task framework, competition using publically available dataset, became popular: Netflix challenge (2009), Kaggle (2010)
- Competitions became important medium to foster development of deep learning.
- Models winning competitions are not necessarily better models.
- Practical success but little understanding of unreasonable effectiveness.
- Even in practical levels, only point estimation is addressed. No inferential tools.

Theoretical challenges

- Recall $E(y|x,\theta) = f_k(\cdots f_3(f_2(f_1(x;\theta_1);\theta_2);\theta_3)\cdots;\theta_k)$.
- Models are highdimensional.
- Requires non-convex optimization. Should deal with local minima, saddle points.
- Black-box model, no interpretation

'One theorem in DL': Universal approximation

- Theorem: Let $\rho(.)$ be a bounded, non-constant continuous function. Let I_m denote the m-dimensional hypercube, and $C(I_m)$ denote the space of continuous functions on I_m. Given any $f \in C(I_m)$ and $\epsilon > 0$, there exists $N > 0$ and $v_i, w_i, b_i,\ i = 1, \cdots, N$ such that

$$F(x) = \sum_{i \leq N} v_i \rho(w_i^T x + b_i)$$

  satisfies $\sup_{x \in I_m} |f(x) - F(x)| < \epsilon$.
- It guarantees that even a single hidden-layer network can represent any classification problem in which the boundary is locally linear.

Mysteries of unreasonable effectiveness of DL applications

- CNN models appear to capture high level image properties more effectively than previous models.
- Perform well in various applications: beating the curse of dimensionality?
- Which architectural components are responsible for this effectiveness mathematically?
- Which optimization tools are responsible for effective learning? (overcoming problems of local minima or saddle points. Equivalence of local solutions? role of stochastic optimization? role of normalization?)
- Universal approximation theorem does not explain why a deep model performs better in practice nor how it relate to the optimization.

Areas of theoretical studies related to DL

Stanford, Berkeley and HKUST courses cover subsets of the following topics.

- Harmonic analysis: Representation of functions
- Approximation theory: When deep networks are better than shallow ones?
- Optimization: Characterize the landscapes of loss function and find optima efficiently
- Statistics: Generalization and inference

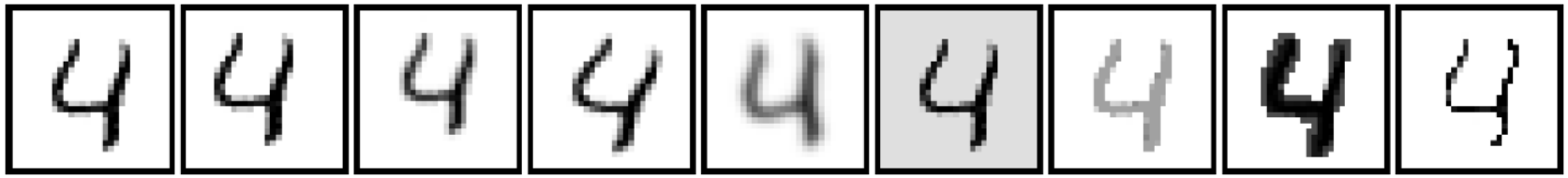
Feature extraction and Discrimination

Two parts of Deep Neural Network

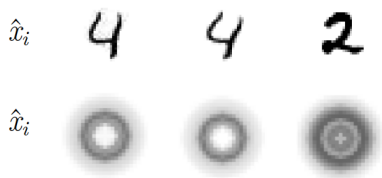
- Outcome can be binary or continuous or multivariate.
- $P(y = 1|x, \theta) = f_k(f_{k-1}(\cdots f_2(f_1(x; \theta_1); \theta_2) \cdots ; \theta_{k-1}) \cdots ; \theta_k)$.
- First part: Extracting features:
  $g(x; \Theta_{k-1}) = f_{k-1}(\cdots f_2(f_1(x; \theta_1); \theta_2) \cdots ; \theta_{k-1})$
- Second part: Classification: $P(y = 1|x, \theta) = f_k(g(x; \Theta_{k-1}); \theta_k)$,
  $f_k(x) = \exp(x)/(1 + \exp(x))$.
- Feature extraction and classification are conducted separately before 2012.
- We briefly review common approaches to feature extraction and classification separately.

Feature extraction

- Features should be invariant with respect to small scale changes, small rotations, blur, brightness, thickness, etc.



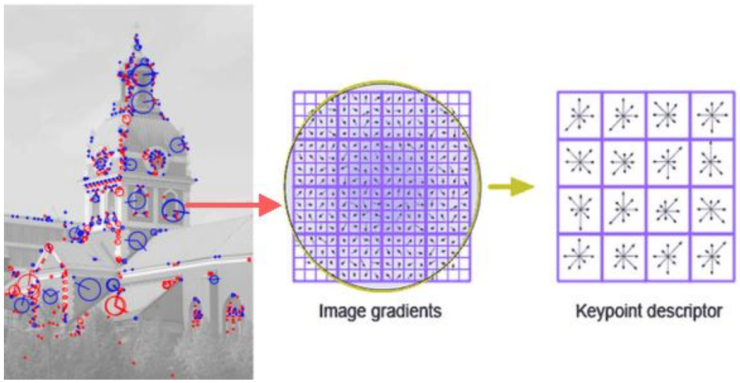
- Averaging or integration makes the images invariant.



Integration over all rotations

Feature extraction: SIFT

- The feature extraction step is performed manually using known transformations such as Scale-Invariant-Feature-Transformation (SIFT), Rotation-Invariant-FT, Histogram of oriented Gradients (HoG), etc.
- SIFT: A goal is to extract distinctive invariant features; locate the feature key point, assign orientation to the key points, describe the key point as a high dim vector



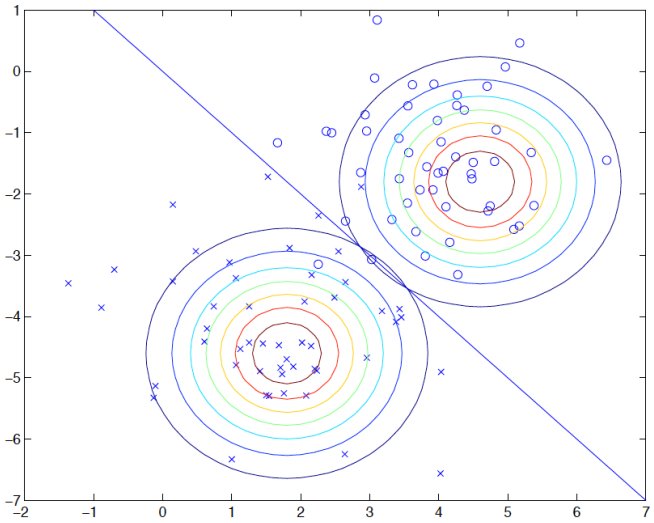
invariant to image scale and rotation; robust to affine distortion, change in 3D viewpoint locality, robust to occlusion and clutter.

Discrimination

Linear discriminant analysis

- A classification rule is a function $f : \chi \to \{1, \cdots, K\}$ where χ is the domain of X. For new X, prediction of Y is $f(X)$.
- In case of binary classification, $F(x) = w^T x + b$ such that $f(x) = I(F(x) > 0)$ is called a linear discriminant.
- The misclassification rate of f is defined $R(f) = P(Y \neq f(x))$. The rule that minimizes $R(f)$ is called Bayes rule.
- Bayes classifier $f(x)$ is $f(x) = \arg \max_{j=1,\cdots,K} P(Y = j|X = x) = \arg \max_{j=1,\cdots,K} P(X = x|Y = j)P(Y = j)$
- LDA uses Bayes rule assuming $P(X = x|Y = j) = N(\mu_j, \Sigma)$, $\mu_j \in \mathbb{R}^p$, $\Sigma \in \mathbb{R}^{p \times p}$

Linear discriminant analysis



- LDA classification rule: $\hat{w}X \geq c$, where $\hat{w} = \hat{\Sigma}^{-1}(\hat{\mu}_0 - \hat{\mu}_1)$, $c = (\hat{\mu}_0 - \hat{\mu}_1)^T \hat{\Sigma}^{-1}(\hat{\mu}_0 + \hat{\mu}_1)/2$.
- Consider $\hat{\Sigma} = UDU^T$, $\tilde{\mu}_j = D^{-\frac{1}{2}} U^T \hat{\mu}_j$, $\tilde{x} = D^{-\frac{1}{2}} U^T x$, $x \in \mathbb{R}^p$. LDA classifies $\tilde{x}$ to the nearest centroid, i.e. assigns to class j so that $\frac{1}{2}||\tilde{x} - \tilde{\mu}_j||_2^2 - \log \hat{\pi}_j$ is minimized.

Linear discriminant analysis

- Without assuming multivariate normality, $E(w^T X|Y=j) = w^T \mu_j$, $Var(w^T X|Y=j) = w^T \Sigma w$. Let

$$J(w) = \frac{\{E(w^T X|Y=0) - E(w^T X|Y=1)\}^2}{w^T \Sigma w} = \frac{w^T(\mu_0 - \mu_1)(\mu_0 - \mu_1)^T w}{w^T \Sigma w}.$$

- $\hat{w} = \hat{\Sigma}^{-1}(\hat{\mu}_0 - \hat{\mu}_1)$ minimizes $\hat{J}(w)$. Fisher linear discriminant function is $\hat{w}X$, and the classification rule $\hat{w}X \geq c$, where $c = (\hat{\mu}_0 - \hat{\mu}_1)^T \hat{\Sigma}^{-1}(\hat{\mu}_0 + \hat{\mu}_1)/2$, is the same as the LDA.

Discriminative vs. Generative

- Discriminative $P(Y = j|X)$ vs. generative models $P(X|Y = j)$.
- When $P(X = x|Y = j) = N(\mu_j, \Sigma)$, $j = 1, 2$,
  $P(Y = 1|X = x) = \exp(\beta_0 + \beta_1 x)/\{1 + \exp(\beta_0 + \beta_1 x)\}$, where
  $\beta_0 = \log \frac{\pi_1}{\pi_2} - \frac{\mu_1^T \Sigma^{-1} \mu_1}{2} + \frac{\mu_2^T \Sigma^{-1} \mu_2}{2}$ and $\beta_1 = (\mu_1 - \mu_2)^T \Sigma$.
- Logistic regression is a discriminative version of LDA.
- When $f(X|Y = j) = \prod_{i=1,\cdots,p} f_i(x_i)$, for $X \in \mathbb{R}^p$, $P(Y = 1|X)$ gives generalized additive models.
- Deep learning can be viewed as logistic regression given the features of the last layer.

Challenges of deep learning

- Since features are functions of many parameters in compositional forms, deep learning poses a high dimensional nonconvex model fitting problem.
- High dimensional problems are often handled exploiting *sparsity* or *smoothness*.
- In statistical literature high dimensional convex problems have been studied.
- High dimensional nonconvex cases have been studied (e.g. Loh and Wainwright, 2017) for global optima.

Handling high-dimensional problems using sparsity

- One overarching strategy of solving high-dimensional problem is 'bet on sparsity'.
- Intuitively, if we have sample size n and the unknown p, with $p >> n$, the number of samples n is too small to allow for accurate estimation of the parameters, unless many of p parameters are zero. If the true model is sparse, so that only $k < n$ parameters are actually nonzero, then we can estimate the parameters. (Hastie, Tibshirani and Wainwright, SLS)
- For example, for lasso, if $||\beta^*||_1 = o(\sqrt{n/\log(p)})$, lasso is known to be consistent for prediction.

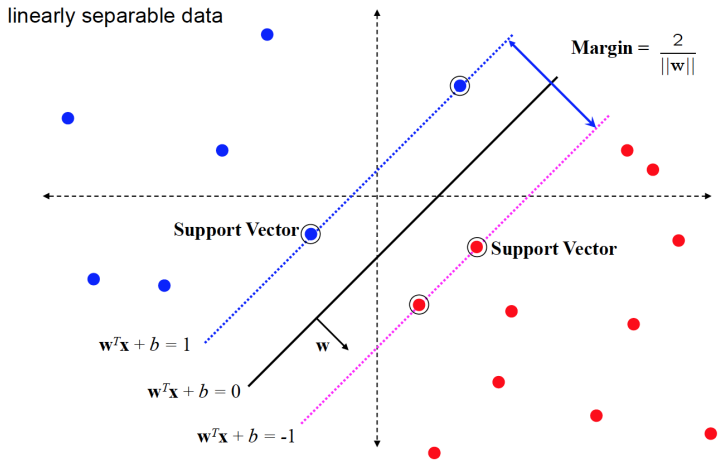
Handling high-dimensional problems using smoothness

- (Informally) Functions can be considered as infinitely long vector.
- Estimating an unknown function requires overcoming highdimensional problems.
- One popular strategy to such problem is to assume that the unknown function belongs to a restricted function class.
- Typically the function class is defined by degree of smoothness.
- SVM can be justified in two ways. One gives sparse solution and the other utilizes restricted function space called Reproducing Kernel Hilbert Space (RKHS). We review SVM in these respects.

Support Vector Machine

Support vector machine: Linearly separable case

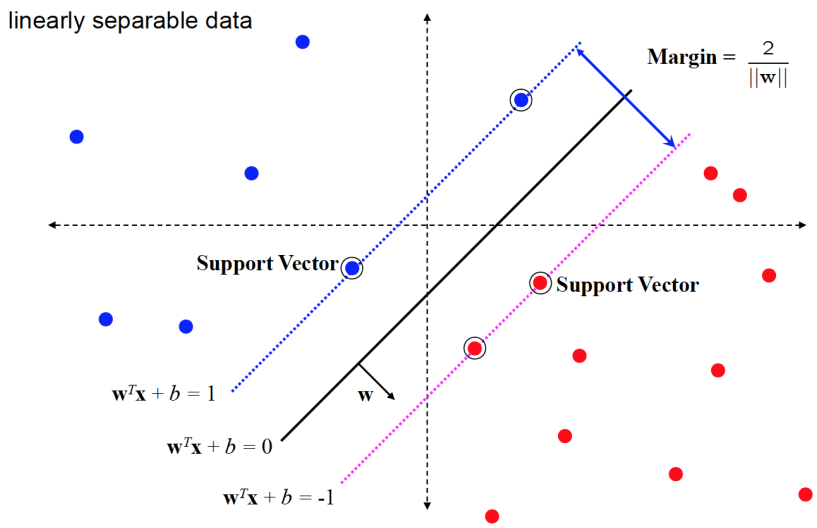
- Consider a hyperplane that can separate data. $x_i \in \mathbb{R}^d$, $i = 1, \cdots n$. $d >> n$.
- Margin=minimum distance of x_i over i from the boundary plane.
- SVM is a classifier by the hyperplane with maximum margin.



slide from Andrew Zisserman.

- Since $w^T x + b = 0$ and $c(w^T x + b) = 0$ represent the same plane, choose normalizatioin such that $w^T x + b = 1$ and $w^T x + b = -1$ for support vectors, which give margin of $2/||w||$.

Support vector machine: Linearly separable case



- w is obtained by maximizing $2/||w||$ subject to $w^T x_i + b \geq 1$ if $y_i = 1$, $w^T x_i + b \leq -1$ if $y_i = -1$ for $i = 1, \cdots, n$.
- Equivalently, $\min_w ||w||^2$ subject to $y_i(w^T x_i + b) \geq 1$ for $i = 1, \cdots, n$.

Support vector machine: Equivalent optimization problems

- (Primal) $\min_{\mathbf{w},b} \max_{\boldsymbol{\alpha}\geq 0} \frac{1}{2}||\mathbf{w}||^2 - \sum_j \alpha_j[(\mathbf{w}^T\mathbf{x}_j + b)y_j - 1]$
- (Dual) $\max_{\boldsymbol{\alpha}\geq 0} \min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 - \sum_j \alpha_j[(\mathbf{w}^T\mathbf{x}_j + b)y_j - 1]$
- Solve for dual. $\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_j \alpha_j y_j \mathbf{x}_j = 0 \quad \rightarrow \mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$
- $\frac{\partial L}{\partial b} = -\sum_j \alpha_j y_j \quad \rightarrow \sum_j \alpha_j y_j = 0.$
- Plugging back, (Dual) $\max_{\boldsymbol{\alpha}\geq 0, \sum_j \alpha_j y_j = 0} \sum_j \alpha_j - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i^T \mathbf{x}_j)$

  (solving for $\boldsymbol{\alpha}$)

Support vector machine: Dual solution

- Plugging back, (Dual) $\max_{\boldsymbol{\alpha} \geq 0, \sum_j \alpha_j y_j = 0} \sum_j \alpha_j - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i^T \mathbf{x}_j)$

  (solving for $\boldsymbol{\alpha}$)
- $\mathbf{w}$ and b are computed from α.
- $y \leftarrow \text{sign}[\sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b]$.
- The solution depends on x only through $x^T x$.
- Using the dual form, the dimension is reduced.

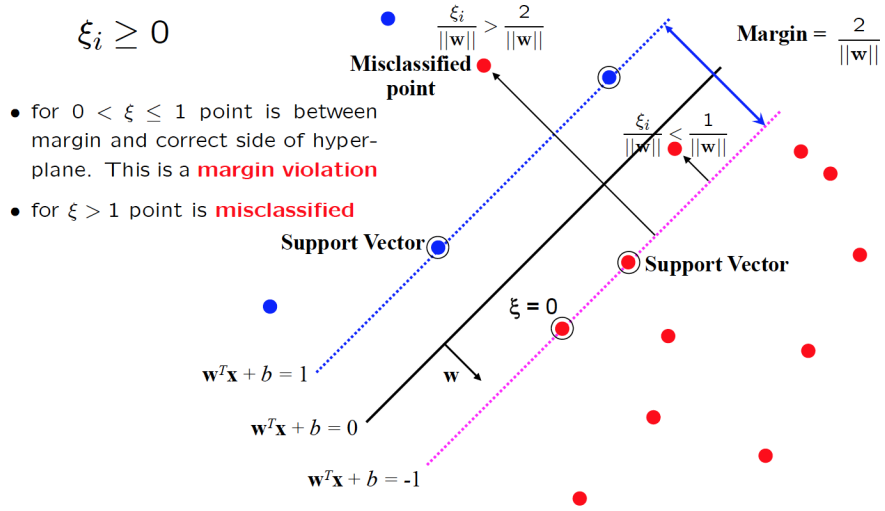
Support vector machine: Sparse solution

- Convexity of the objective function and constraint allows to solve dual problem. Moreover the solution satisfies Karush-Kuhn-Tucker complementary slackness condition.
- (Sparsity) The solution satisfies the KKT complemetary slackness condition,

$$\alpha_i\{y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1\} = 0.$$

That is, for the support vector, $\alpha_i > 0$, for other points the constraint is inactive, $\alpha_i = 0$.

Support vector machine: Linearly nonseparable case



slide from Andrew Zisserman.

$$\min_{\mathbf{w}\in\mathbb{R}^d, \xi_i \in \mathbb{R}^+} ||w||^2 + C \sum_{i=1}^{n} \xi_i$$

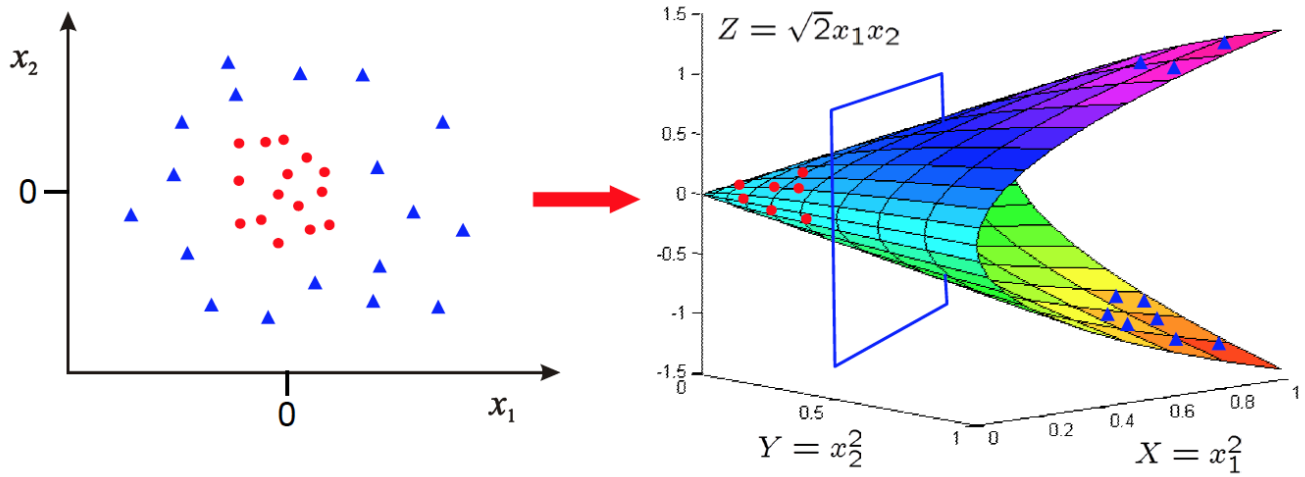
subject to $y_i(w^T x_i + b) \geq 1 - \xi_i$ for $i = 1, \cdots, n$.

Support vector machine: Linearly nonseparable case

- $\min_{\mathbf{w}\in\mathbb{R}^d,\xi_i\in\mathbb{R}^+} ||w||^2 + C\sum_{i=1}^n \xi_i$ subject to $y_i(w^T x_i + b) \geq 1 - \xi_i$ for $i = 1, \cdots, n$. Margin is allowed to be less than 1, $1 - \xi$, but price is paid by increasing objective function by $C\xi$.
- Going through similar derivation of dual form as in separable case,
  $\max_{\boldsymbol{\alpha}} \sum_j \alpha_j - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j (\mathbf{x}_i^T \mathbf{x}_j)$,
  $0 \leq \boldsymbol{\alpha} \leq C$, $\sum_j \alpha_j y_j = 0$.
- $y \leftarrow \text{sign}[\sum_i \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b]$.
- C is a regulization parameter. small C = soft margin. large C = hard margin.

Classification in transformed space

Note that optimization depends on $\mathbf{x}$ only through $\mathbf{x}^T\mathbf{x}$. Even if we use $\mathbf{z}$ with higher dimension than $\mathbf{x}$, the solution is in the same dimension. Linearly nonseparable data can become separable in a higher dimension.



$$\phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$$

Transforming to a higher dimension

- $\phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 x_2 \end{pmatrix}$
- Data may become linearly separable in transformed space.
- We can replace ${x_i}^T x_j$ with $\phi(x_i)^T \phi(x_j)$ in the algorithm.
- The classifier is $f(x) = \mathbf{w}^T \phi(x) + b = \sum_{i=1}^{n} \alpha_i y_i \phi(x_i)^T \phi(x) + b$.
- Define kernel function $k(x, z) = \phi(x)^T \phi(z)$. The classifier is $f(x) = \sum_{i=1}^{n} \alpha_i y_i k(x_i, x) + b$.

Kernel trick

- Define kernel function $k(x, z) = \phi(x)^T \phi(z)$. The classifier is $f(x) = \sum_{i=1}^{n} \alpha_i y_i k(x_i, x) + b$.
- Although we can compute $\phi(x)^T \phi(z)$ given $\phi(.)$, $k(x, z)$ may be easy to compute. e.g. $k(x, z) = (x^T z)^d$. Some kernels are infinite dimensional, e.g., $k(x, z) = \exp(-\frac{1}{2\sigma^2}||x - z||_2^2)$.
- If we specify $k(x, z)$, does ϕ exist? Yes, if K is valid matrix.
  Theorem (Mercer). Let $K : \chi \times \chi \to \mathbb{R}$ be given. For K to be a valid (Mercer) kernel, it is necessary and sufficient that for any $\{x^{(1)}, \cdots, x^{(m)}\}$, the corresponding kernel matrix is symmetric positive semi-definite.

Data analysis using gaussian kernel

- $f(x) = \sum_{i=1}^{n} \alpha_i y_i \exp(-||x - x_i||^2/2\sigma^2) + b.\ \sigma = 1.$

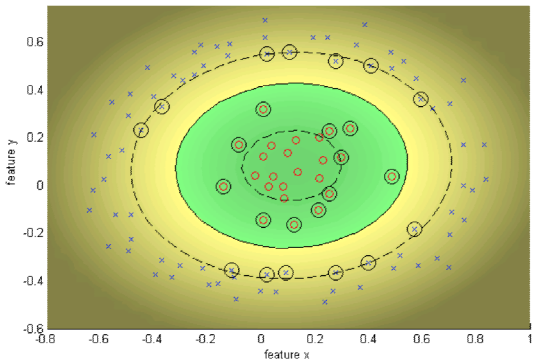


Figure: C=10

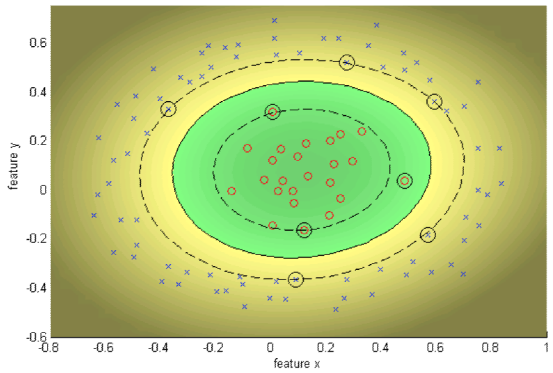


Figure: C=100

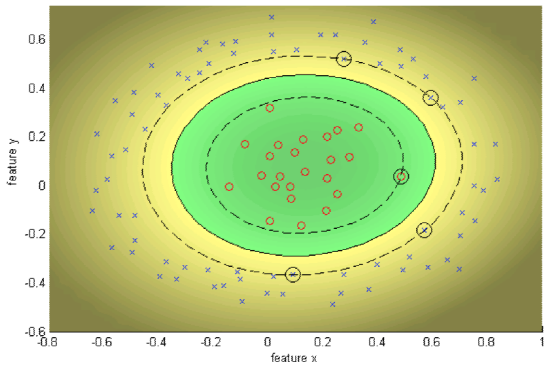


Figure: C=∞

Alternative justification of SVM

- SVM reduces the dimension of the problem via dual thanks to convexity.
- Convexity also leads to the KKT complimentary slackness condition and the sparsity of the solution.
- Alternative approach uses Reproducing Kernel Hilbert Space (RKHS) and the representer theorem which reduces function estimation to a finite linear combination of kernel products evaluated at the input value in the training dataset.
- We review RKHS and the representer theorem.

Reproducing Kernel Hilbert Space

Reproducing Kernel Hilbert Space: Kermel

- Definition (Positive definite functions) A symmetric function h: $\chi \times \chi \to \mathbb{R}$ is positive definite if $\forall n \geq 1$, $\forall (a_1, \cdots, a_n) \in \mathbb{R}^n$, $\forall (x_1, \cdots, x_n) \in \chi^n$, $\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j h(x_i, x_j) \geq 0$.
- Definition (Kernel) Let χ be a non-empty set. A function $k : \chi \times \chi \to \mathbb{R}$ is a kernel if there exist an $\mathbb{R}$-Hilbert space and a feature map $\phi : \chi \to \mathcal{H}$ such that $\forall x, x' \in \chi$, $k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$.
- Kernel, $k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$, is positive definite.

Reproducing Kernel Hilbert Space: How to generate

- We first discuss how to generate RKHS function space then present a formal definition.
- For any positive definite kernel, $k(x, .)$ is the function obtained by fixing the first corordinate at x. For the Gaussian kernel, $k(x, .)$ is a normal density function centered at x. We can generate functions $f(.) = \sum_{i=1}^{n} \alpha_i k(x_i, .)$, $g(.) = \sum_{j=1}^{m} \beta_j k(y_j, .)$. Given such generated two functions, define $\langle f, g \rangle = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \beta_j k(x_i, y_j)$. Note that $\langle k(x_i,), k(y_j, .) \rangle = \alpha_i \beta_j k(x_i, y_j)$.
- This mapping turns out to satisfy the definition of inner product.
- The function space generated by kernel $k(x, .)$ is a Hilbert space.
- The kernel has reproducing property.

Reproducing Kernel Hilbert Space

- Definition (Inner product) Let $\mathcal{F}$ be a vector space over $\mathbb{R}$. A function $\langle ., . \rangle_{\mathcal{F}} : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$ is said to be an inner product on $\mathcal{F}$ if
  1. $\langle \alpha_1 f_1 + \alpha_2 f_2, g \rangle_{\mathcal{F}} = \alpha_1 \langle f_1, g \rangle_{\mathcal{F}} + \alpha_2 \langle f_2, g \rangle_{\mathcal{F}}$
  2. $\langle f, g \rangle_{\mathcal{F}} = \langle g, f \rangle_{\mathcal{F}}$
  3. $\langle f, f \rangle_{\mathcal{F}} \geq 0$ and $\langle f, f \rangle_{\mathcal{F}} = 0$ if and only if $f = 0$.
- Definition (Complete space) A space is complete if every Cauchy sequence in that space has a limit and this limit is in that space.
- Definition (Hilbert space) Hilbert space is a complete inner product space.
- Definition (Reproducing kernel): Let $\mathcal{H}$ be a Hilbert space of $\mathbb{R}$-valued functions defined on a non-empty set χ. A function $k : \chi \times \chi \to \mathbb{R}$ is called a reproducing kernel on $\mathcal{H}$ if it satisfies
  1. $\forall x \in \chi,\ k(., x) \in \mathcal{H}$,
  2. $\forall x \in \chi,\ \forall f \in \mathcal{H},\ \langle f, k(\cdot, x) \rangle_{\mathcal{H}} = f(x)$.

Reproducing Kernel Hilbert Space

- Definition (Reproducing Kernel Hilbert Space): A Hilbert space $\mathcal{H}$ of functions $f : \chi \to \mathbb{R}$ defined on a non-empty set χ is said to be a Reproducing Kernel Hilbert Space (RKHS) if evaluation functional δ_x is continuous $\forall x \in \chi$.
- Definition (Evaluation functional): Let $\mathcal{H}$ be a Hilbert space of functions $f : \chi \to \mathbb{R}$, defined on a non-empty set χ. For a fixed point $x \in \chi$, map $\delta_x : \mathcal{H} \to \mathbb{R}$, $\delta_x : f \mapsto f(x)$ is called the evaluation functional at x.
- Definition (Continuity): A function A: $\mathcal{H} \to \mathcal{G}$, where $\mathcal{H}$, $\mathcal{G}$ are both normed linear spaces over $\mathbb{R}$, is said to be continuous at $f_0 \in \mathcal{H}$ if for every $\epsilon > 0$, there exists a $\delta = \delta(\epsilon, f_0) > 0$, s.t. $||f - f_0||_{\mathcal{H}} < \delta$ implies $||Af - Af_0||_{\mathcal{G}} < \epsilon$.
- An important property of an RKHS is that if two functions $f \in \mathcal{H}$ and $g \in \mathcal{H}$ are close in the norm of $\mathcal{H}$, then $f(x)$ and $g(x)$ are close for all $x \in \chi$. This is due to the definition of RKHS.

Reproducing Kernel Hilbert Space:

- Theorem (Existence of the reproducing kernel): $\mathcal{H}$ is a RKHS (i.e., its evaluation functionals δ_x are continuous linear operators), if and only if $\mathcal{H}$ has a reproducing kernel. Denote RKHS spanned by k by $\mathcal{H}_k$.
- (Summary) We can consider a Hilbert space where evaluation functional is continuous. This continuity allows existence of reproducing kernel. Function space constructed using these reproducing kernels is a RKHS.

Representer theorem (Kimeldorf and Wahba, 1971)

- (Representer Theorem) Let l be a loss function on $f = \beta_0 + h$ and $h \in \mathcal{H}$ and $\mathcal{H}$ be a RKHS generated by a Mercer kernel k. Let $\hat{f}$ minimize

$$C_n(f) = \sum_{i=1}^{n} l(y_i, f(x_i)) + \lambda ||h||_{\mathcal{H}}.$$

  Then

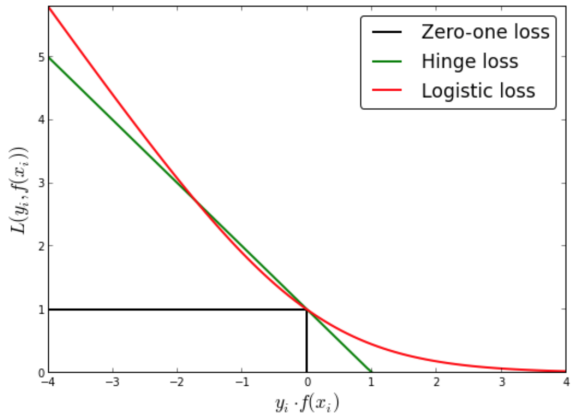
$$\hat{f}(x) = b + \sum_{i=1}^{n} \alpha_i k(x_i, x)$$

  where b and $\alpha_i \in \mathbb{R}$, $i = 1, \cdots, n$.

- Representer theorem reduces a infinite dimensional problem to a finite dimensional problem.

Alternate view of SVM

- In classification, we would like to find f to minimize misclassification rate $E_{y,x} I(y \neq f(x))$. It is hard to minimize bona fide 0-1 loss since it involves combinatorial computation. Surrogate loss functions can be used.
- In logistic regression we minimize logistic loss, $\log[1 + \exp\{-y_i f(x_i)\}]$.
- The SVM is equivalent to minimizing hinge loss, $\sum_{i=1}^{n} \max(0, 1 - y_i f(x_i))$ where $f \in \mathcal{H}$, with the regularization term $||f||_{\mathcal{H}}^2$. That is to minimize empirical loss while controlling complexity of f measured by the norm in RKHS.



SVM: Minimizing a hinge loss in a RKHS

- Consider minimizing a hinge loss, $\sum_{i=1}^{n} \max(0, 1 - y_i f(x_i))$ and f is in RKHS $\mathcal{H}$ with kernel $k(x, .)$
- Due to the Representer theorem, minimizer of

$$\sum_{i=1}^{n} \max(0, 1 - y_i f(x_i)) + \lambda ||f||_{\mathcal{H}}^2$$

  has a form $\hat{f}(.) = \sum_{i=1}^{n} \beta_i k(x_i, .)$ with $||f||_{\mathcal{H}}^2 = \sum_{i,j}^{n} \beta_i \beta_j k(x_i, x_j)$.
- Plugging in, the problem reduces to minimize over β

$$\sum_{i=1}^{n} \max(0, 1 - y_i \sum_{j=1}^{n} \beta_j k(x_j, x_i)) + \lambda \sum_{i,j}^{n} \beta_i \beta_j k(x_i, x_j)$$

Some theory of machine learning

Concentration of measures

- Empirical risk minimization: Consider (x_i, y_i), $x_i \in \mathbb{R}^d$, $y \in \{0, 1\}$. Given a classifier $f : \mathbb{R}^d \to \{0, 1\}$, the training error is $\hat{R}_n(f) = n^{-1} \sum_{i=1}^n I(Y_i \neq f(X_i))$. The true classification error is $R(f) = P(Y \neq f(X))$. We would like to know how close $\hat{R}(f)$ is to $R(f)$ with high probability.
- Given f, $P(|\hat{R}(f) - R(f)| > \epsilon) < \delta$
- Uniform bounds: $P(\sup_{f \in \mathcal{F}} |\hat{R}(f) - R(f)| > \epsilon) < \delta$ over a class of functions $\mathcal{F}$.

Hoeffding's Inequality

- Hoeffding's Inequality: If $Z_1, \cdots, Z_n$ are independent with $P(a_i \leq Z_i \leq b_i) = 1$, then for any $t > 0$,

$$P(|\bar{Z}_n - \mu| > \epsilon) < 2e^{-2n\epsilon^2/c},$$

  where $c = n^{-1}\sum_{i=1}^{n}(b_i - a_i)^2$ and $\bar{Z}_n = n^{-1}\sum_{i=1}^{n} Z_i$.
- Hoeffding's inequality implies

$$P(|R(f) - \hat{R}_n(f)| > \epsilon) \leq 2e^{-2n\epsilon^2},$$

  and

$$E|R(f) - \hat{R}_n(f)| \leq \sqrt{\frac{1}{2n}\log(\frac{2}{\delta})}$$

Measures of complexity

- Let $|\mathcal{F}| = M$. For $m = 1, \cdots, M$, event $\mathcal{B}_m$, $|\hat{R}(f_m) - R(f_m)| > \epsilon$, $P(\cup_{i=1}^{M} \mathcal{B}_m) = P(\sup_{f \in \mathcal{F}} |\hat{R}(f_m) - R(f_m)| > \epsilon) < 2Me^{-2n\epsilon^2}$.
- If a class of function $\mathcal{F}$ is finite, the size of the set measures the complexity.
- To develop uniform bounds we need to introduce some complexity measures.
- If $\mathcal{F}$ is infinite, we need a tool to measure complexity.
- Shattering number (or growth function), Vapnik and Chervonenkis (VC) dimension, Rademacher complexity (Koltchinskii and Panchenko, 2002) are measures of complexity.

Measures of complexity: Shattering number

- Let χ be a set and let $\mathcal{F}$ be a class of binary functions on χ. For $f \in \mathcal{F}$, $f : \{z_1, \cdots, z_n\} \to \{0, +1\}$. Define $\mathcal{F}(z_1, \cdots, z_n) = \{(f(z_1), \cdots, f(z_n))\}$. $|\mathcal{F}|$ can be infinite, but $|\mathcal{F}(z_1, \cdots, z_n)| \leq 2^n$.
- e.g.1. $f(z) = I(z > a)$ with $z_1 < z_2 < z_3$. Then $\mathcal{F}(z_1, z_2, z_3) = \{(0,0,0), (0,0,1), (0,1,1), (1,1,1)\}$.
- e.g.2. $f(z) = I(a < z < b)$ with $z_1 < z_2 < z_3$. Then $\mathcal{F}(z_1, z_2, z_3) = \{(0,0,0), (1,0,0), (1,1,0), (0,0,1), (0,1,1), (1,1,1), (0,1,0)\}$.
- Shattering number, growth function (definition): number of the most possible sets of dichotomies on any n points:

$$m_{\mathcal{F}}(n) = max_{z_1, \cdots z_n} |\mathcal{F}(z_1, \cdots z_n)|.$$

Measures of complexity: Shattering number

- By definition, the shattering number satisfies $m_{\mathcal{F}}(n) \leq 2^n$.
- For e.g.1, $\mathcal{F}$ is set of $f\colon \mathbb{R} \to \{0,1\}$, $f(z) = I(z > a)$. When $n = 3$ with $z_1 < z_2 < z_3$,
  $\mathcal{F}(z_1, z_2, z_3) = \{(0,0,0), (0,0,1), (0,1,1), (1,1,1)\}$.
  $m_{\mathcal{F}}(n) = n + 1$.
- e.g.2. $\mathcal{F}$ is set of $f\colon \mathbb{R} \to \{0,1\}$, $f(z) = I(a < z < b)$. When $n = 3$ with $z_1 < z_2 < z_3$. Then $\mathcal{F}(z_1, z_2, z_3) =$
  $\{(0,0,0), (1,0,0), (1,1,0), (0,0,1), (0,1,1), (1,1,1), (0,1,0)\}$.
  $m_{\mathcal{F}}(n) = n(n+1)/2 + 1$.
- $\mathcal{F}$ is set of $f\colon \mathbb{R}^2 \to \{0,1\}$, $f(z) = 1$ is convex.
  $m_{\mathcal{F}}(n) = 2^n$.

Measures of complexity: VC dimension

- The VC dimension of $\mathcal{F}$ is the largest value of n for which $m_{\mathcal{F}}(n) = 2^n$, i.e. $d_{VC}(\mathcal{F}) = \sup\{n : m_{\mathcal{F}}(n) = 2^n\}$.
- Break point: Number of data points for which we cannot get all possible dichotomies (=VC dimension +1).

Table: VC dimensions

Function class	VC dimension
interval $[a, b]$	2
Disc in $\mathbb{R}^2$	3
half-spaces in $\mathbb{R}^d$	$d + 1$
Convex polycons in $\mathbb{R}^2$	∞

Measures of complexity: Sauer's Theorem

- VC dimension or break point regardless of $\mathcal{F}$ can give information about $m_{\mathcal{F}}(n)$. If we know the break point is k, we know that any k points of n cannot have all possible patterns. e.g. $n = 3$, $k = 2$. This observation gives a upper bound for $m_{\mathcal{F}}(n)$.
- Sauer's Theorem: Suppose that $\mathcal{F}$ has finite VC dimension d. Then

$$m_{\mathcal{F}}(n) \leq \sum_{i=0}^{d} \binom{n}{i}$$

and for all $n \geq d$,

$$m_{\mathcal{F}}(n) \leq (\frac{en}{d})^d$$

- If $m_{\mathcal{F}}(n)$ is less than 2^n, $m_{\mathcal{F}}(n)$ is polynomial, nothing in between.

General measures of complexity: Rademacher complexity

- Motivation: $y = \{-1, 1\}$. $f : \chi \to \{-1, 1\}$.
  $\hat{R}(f) = \frac{1}{m} \sum_{i=1}^{m} I(f(x_i) \neq y_i) = \frac{1}{2} - \frac{1}{2m} \sum_{i=1}^{m} y_i f(x_i)$.
- Minimizing training error is to maximize $\frac{1}{m} \sum_{i=1}^{m} y_i f(x_i)$ over h. Consider a random label instead of y. A bigger model class $\mathcal{F}$ can make $\frac{1}{m} \sum_{i=1}^{m} y_i f(x_i)$ big as well.
- Definition (Rademacher complexity): Let $\sigma_1 \cdots, \sigma_n$ be independent random variables $P(\sigma_i = 1) = P(\sigma_i = -1) = 1/2$. Rademacher complexity of $\mathcal{F}$ is

$$\mathrm{Rad}_n(\mathcal{F}) = E\left\{ \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i f(x_i) \right\}.$$

General measures of complexity: Rademacher complexity

- Empirical Rademacher complexity of $\mathcal{F}$ by

$$\mathrm{Rad}_n(\mathcal{F}, x) = E_\sigma \left\{ \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i f(x_i) \right\}.$$

- When $|\mathcal{F}| = 1$,
  $\mathrm{Rad}_n(\mathcal{F}) = E\left\{\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^n \sigma_i f(x_i)\right\} = E\frac{1}{n}\sum_{i=1}^n E\sigma_i f(x_i) = 0.$
  When $|\mathcal{F}| = 2^n$, $\mathrm{Rad}_n(\mathcal{F}) = E\left\{\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^n \sigma_i f(x_i)\right\} = 1.$
- Theorem. Let $\mathcal{F}$ be a set of binary functions. Then, for all n,

$$\mathrm{Rad}_n(\mathcal{F}) \leq \sqrt{\frac{2 \log m_{\mathcal{F}}(n)}{n}}.$$

Uniform bounds using shattering number

- (Vapnik and Chervonenkis) Let $\mathcal{F}$ be a class of binary functions. For any $t > \sqrt{2/n}$

$$P(sup_{f \in \mathcal{F}} |P_n(f) - P(f)| > t) \leq 4 m_{\mathcal{F}}(2n) e^{-nt^2/8}$$

and hence with probability at least $1 - \delta$,

$$sup_{f \in \mathcal{F}} |P_n(f) - P(f)| \leq \sqrt{\frac{8}{n} \log \left(\frac{4 m_{\mathcal{F}}(2n)}{\delta} \right)}$$

where $P_n(f) = \frac{1}{n} \sum_{i=1}^{n} f(x_i)$, $P(f) = \int f(x) dP(x)$.

Uniform bounds using VC dimension

- Recall $m_{\mathcal{F}}(n) \leq (\frac{en}{d})^d$. Replacing $m_{\mathcal{F}}(n)$ with $(\frac{en}{d})^d$ in VC inequality, we obtain for any $t > \sqrt{2/n}$

$$P(sup_{f \in \mathcal{F}} |P_n(f) - P(f)| > t) \leq 4(\frac{en}{d})^d e^{-nt^2/8}$$

and hence with probability at least $1 - \delta$,

$$\sup_{f \in \mathcal{F}} |P_n(f) - P(f)| \leq \sqrt{\frac{8}{n}\left(\log\left(\frac{4}{\delta}\right) + d\log\left(\frac{ne}{d}\right)\right)}$$

where $P_n(f) = \frac{1}{n}\sum_{i=1}^n f(x_i)$, $P(f) = \int f(x)dP(x)$.

Uniform bounds using Rademacher complexity

- With probability at least $1-\delta$,

$$\sup_{f\in\mathcal{F}} |P_n(f) - P(f)| \leq 2\mathrm{Rad}_n(\mathcal{F}) + \sqrt{\frac{1}{2n}\log(\frac{2}{\delta})}$$

- A proof requires application of theorems of (i) McDiarmid and (ii) Symmetrization. (i) Let $g_n(x) = \sup_{f\in\mathcal{F}} |P_n(f) - P(f)|$, where $x = \{x_1 \cdots, x_i, \cdots, x_n\}$. Let $x' = \{x_1 \cdots, z_i, \cdots, x_n\}$. Then $g_n(x) - g(x') \leq \frac{1}{n}$. By Mcdiarmid inequality, $P(|g(x) - Eg(x)| > \epsilon) \leq 2e^{-2n\epsilon^2}$, and with probability at least $1-\delta$, $g(x) \leq E(g(x)) + \sqrt{\frac{1}{2n}\log\frac{2}{\delta}}$. Using (ii), one can show $E(g(x)) \leq 2\mathrm{Rad}_n(\mathcal{F})$
- This applies to other than binary cases but requires boundedness.

Key theorems for uniform bounds

- (McDiamid) Let $x_1, \cdots, x_n$ be independent random variables. Suppose that $\sup_{x_1,\cdots,x_n,z_i} |f(x) - f(x')| \leq c_i$ for $i = 1, \cdots n$. Then

$$P(|f(x) - E(f(x))| \geq \epsilon) \leq 2 \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^n c_i^2}\right).$$

- (Symmetrization) Let $x_i' \cdots, x_n'$ denote a second independent sample from P and P_n' denote the empirical distribution of the second sample. For all $t > \sqrt{2/n}$,

$$P(\sup_{f \in \mathcal{F}} |(P_n(f) - P(f)| > t) \leq 2P(\sup_{f \in \mathcal{F}} |(P_n(f) - P_n'(f)| > t/2).$$

Uniform bounds using empirical Rademacher complexity

- McDiarmid's inequality implies
  $|\mathrm{Rad}_n(\mathcal{F}) - \mathrm{Rad}_n(\mathcal{F}, x)| \leq \sqrt{\frac{1}{2n}\log(\frac{2}{\delta})}$. This leads
- With probability at least $1 - \delta$,

$$\sup_{f \in \mathcal{F}} |P_n(f) - P(f)| \leq 2\mathrm{Rad}_n(\mathcal{F}, x) + \sqrt{\frac{2}{n}\log(\frac{2}{\delta})}$$

Measures of complexity: Rademacher complexity

- Recall for binary case, $\mathrm{Rad}_n(\mathcal{F}) \leq \sqrt{\frac{2 \log m_{\mathcal{F}}(n)}{n}}$
- Suppose that $\mathcal{F}$ has finite VC dimension d. There exists a constant $C > 0$ that $\mathrm{Rad}_n(\mathcal{F}) \leq C\sqrt{(d \log n)/n}$
- Replacing $C\sqrt{d/n}$ by these two inequalities,

$$\sup_{f \in \mathcal{F}} |P_n(f) - P(f)| \leq 2\sqrt{\frac{2 \log m_{\mathcal{F}}(n)}{n}} + \sqrt{\frac{1}{2n} \log(\frac{2}{\delta})}$$

$$\sup_{f \in \mathcal{F}} |P_n(f) - P(f)| \leq 2C\sqrt{d/n} + \sqrt{\frac{1}{2n} \log(\frac{2}{\delta})}$$

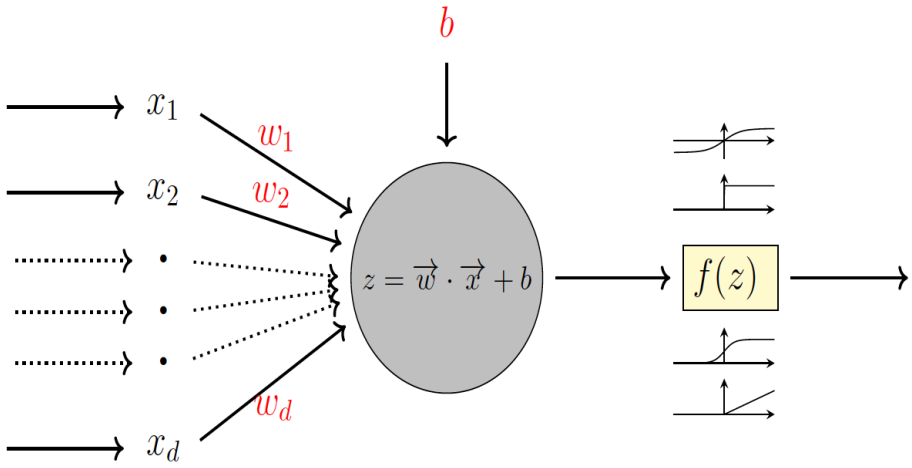
VC dimension bounds for neural network

- One can compute the complexity or capacity of Neural Network models by measuring how many configurations can be shattered (VC dimension)
- The capacity of the network, if measured by the number of pieces in a piecewise linear approximation, increases exponentially with depth [Montufar, Pascanu et al, 2014]
- These results quantify an upper bound on the empirical risk of deep neural networks
- The bounds might be very pessimistic.
- They do not explain the superior generalization properties of CNNs versus models with similar capacity

Overview of Convolutional Neural network

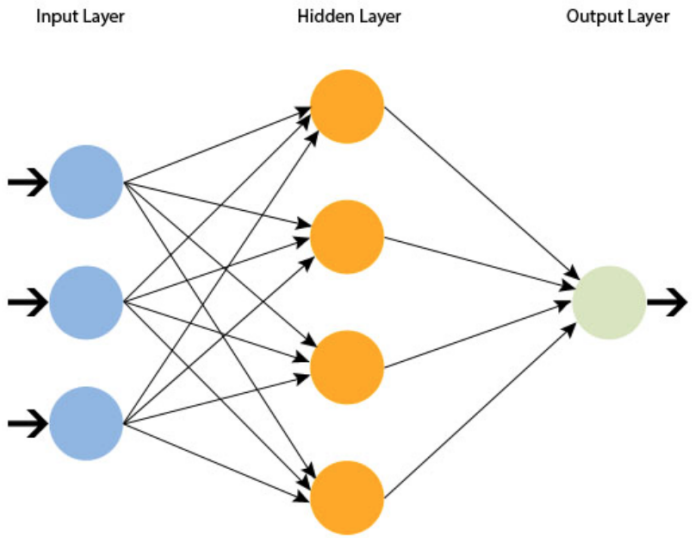
Perceptron: Building block

- Invented by Frank Rosenblatt (1957)



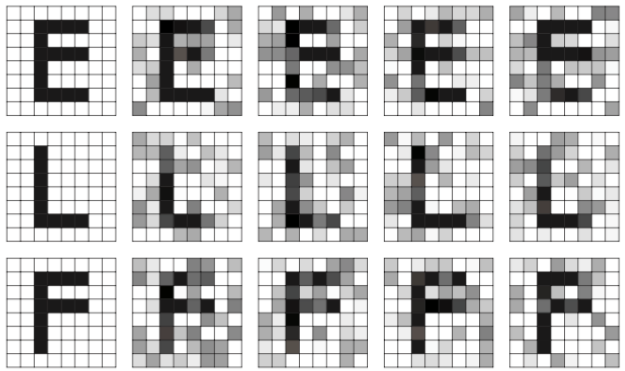
• The perceptron was intended to be a machine, rather than a program, and the perceptron machine was designed for image recognition of an array of 400 photocells.
• The perceptron is an algorithm for a binary classifier: $f(x) = 1$ if $wx + b > 0$, 0, otherwise.

Single-layered neural network

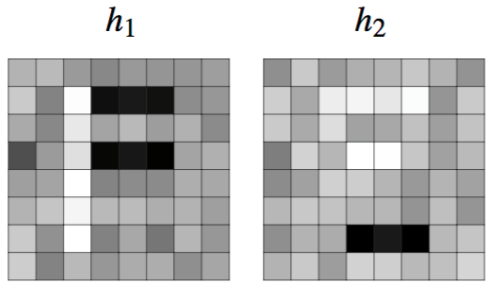


- The perceptron model is called single-layered neural network.

An example of learned filters or weights for input images



training data (with random noise)



weight values of h_1 and h_2 plotted as images

- Note that the filter size is the same as the input size.

Multi-layered feedforward neural network

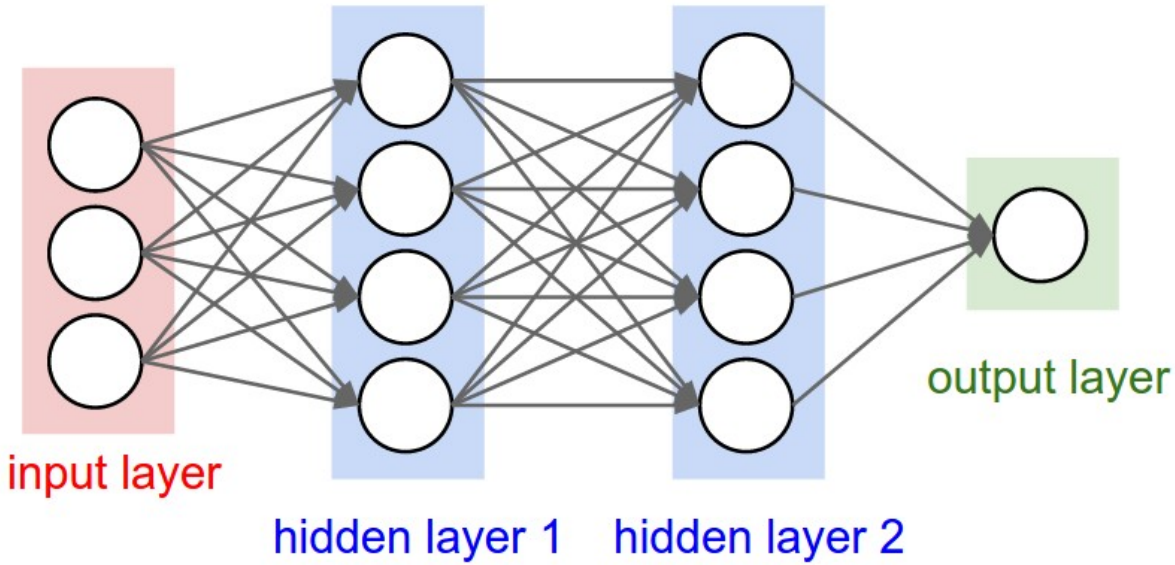


figure from slides of Andrej Karpathy

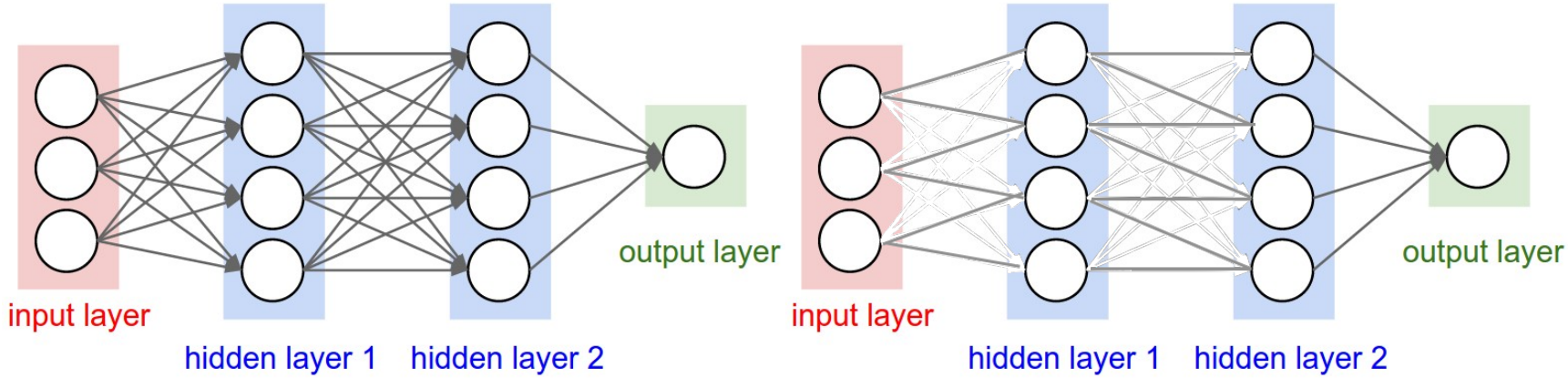
Feedforward neural networks take input x and predict

$$P(y = 1|x, \theta) = f_k(\cdots f_3(f_2(f_1(x; \theta_1); \theta_2); \theta_3) \cdots ; \theta_k).$$

Layers of Artificial Neural Network (ANN)

- $f_l(.)$ is commonly a repeated compositional function of linear and nonlinear transformation.
- Trying to estimate invariant function in a compositional manner.
- A unit of layers is composed of known and unknown transformations.
- Convolutional layer: at the l^{th} layer: $Z^l = W^l h^{l-1} + b^l$, where $h^0 = x$.
- W=filters. Z^l= neurons. W's and b's are unknown and to be estimated or trained.
- Pooling layer
- Activation layer: $h^l = g_l(Z^l)$: nonlinear transformation
- The last layer: softmax: $h_i^K = \exp(Z^i)/\sum_{l=1}^k \exp(Z^l)$.

Convolutional neural network (CNN)



- CNN is a special case of feedforward neural network with locality and sharing restriction.
- This characteristic is referred to as 'shift invariance'.
- Restriction reduces the number of parameters and helps capture local characteristics.

Convolutional layer

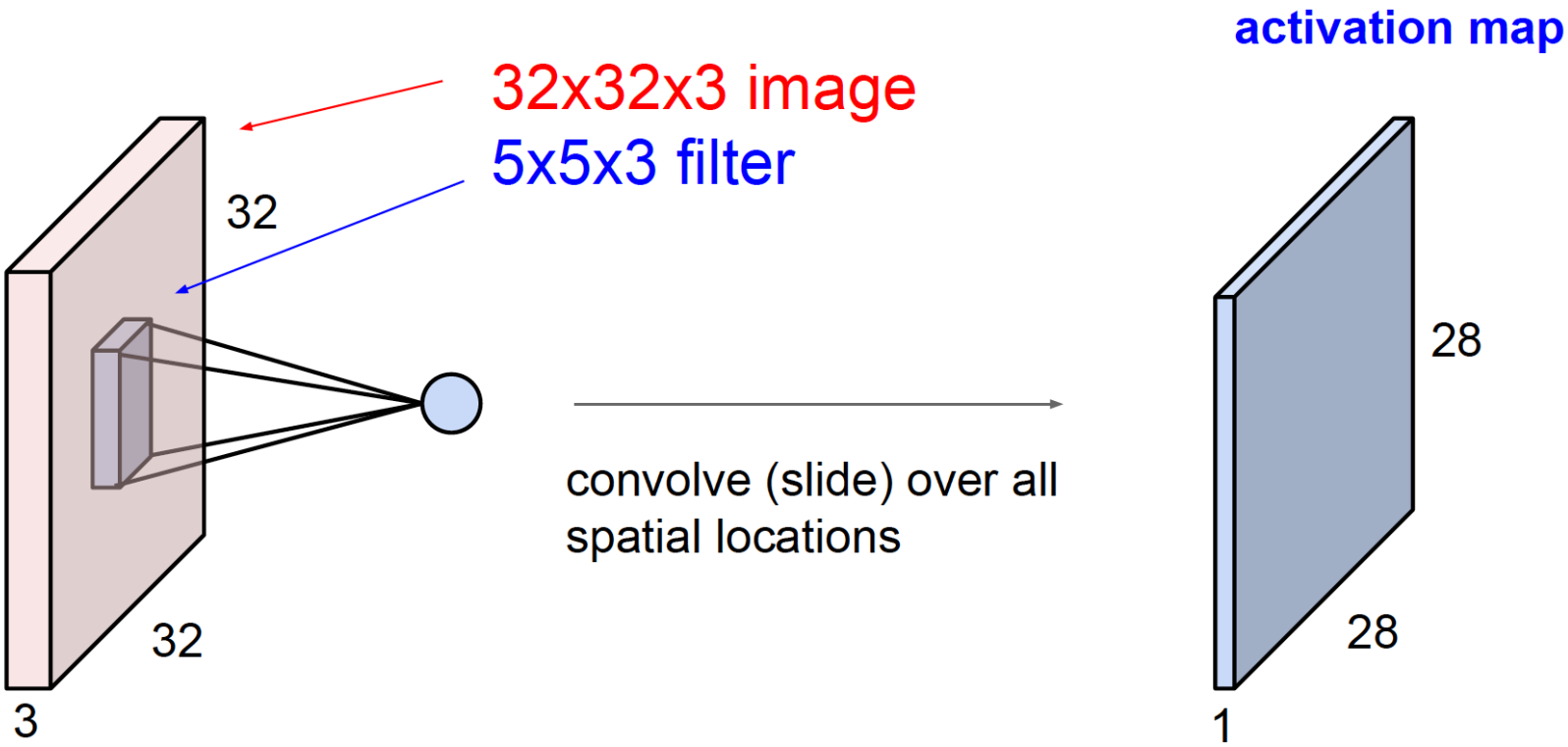


figure from slides of Andrej Karpathy

Resulting output is a 28 by 28 activation map.

Convolutional layer

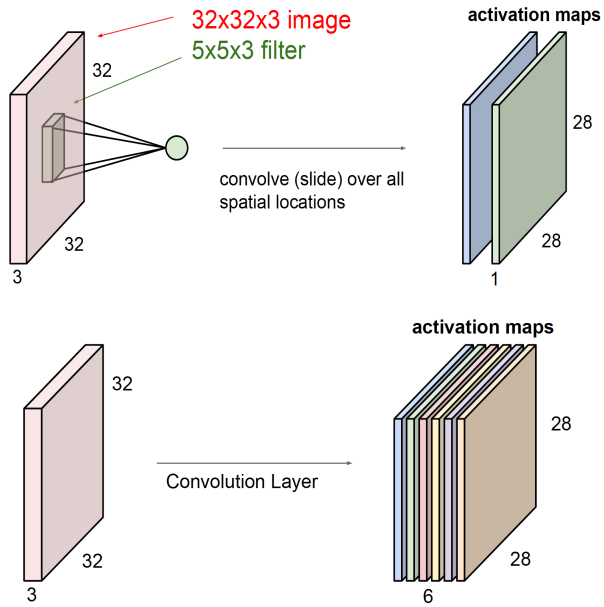


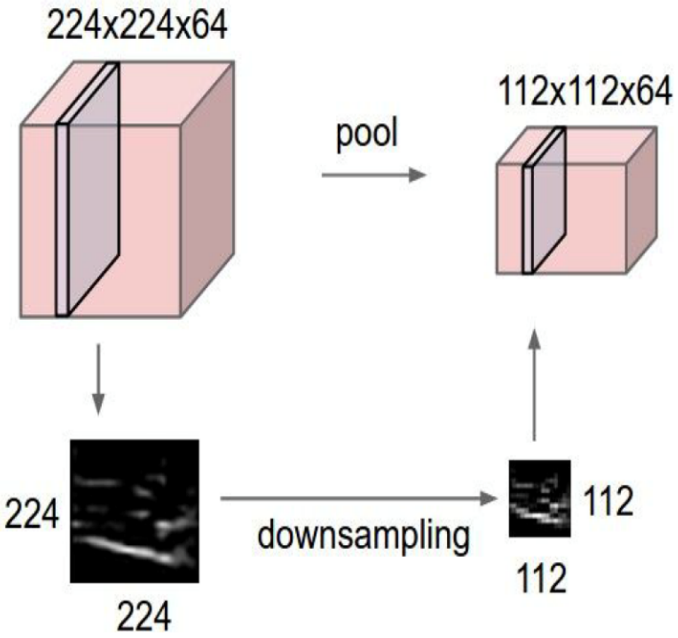
figure from slides of Andrej Karpathy

Apply 6 filters and obtain 6 activation maps.

Role of locality and sharing of convolutional layer

- How locality and sharing reduces the number of parameters?
- If 32x32x3 volume is processed to 28x28x6 volume as in the figure using fully connected layer, the number of parameters=(32*32*3)*(28*28*6)=14.5 Million
- With 6 5x5 filters, we only used (5*5*3)*6=450 parameters.

Pooling layer



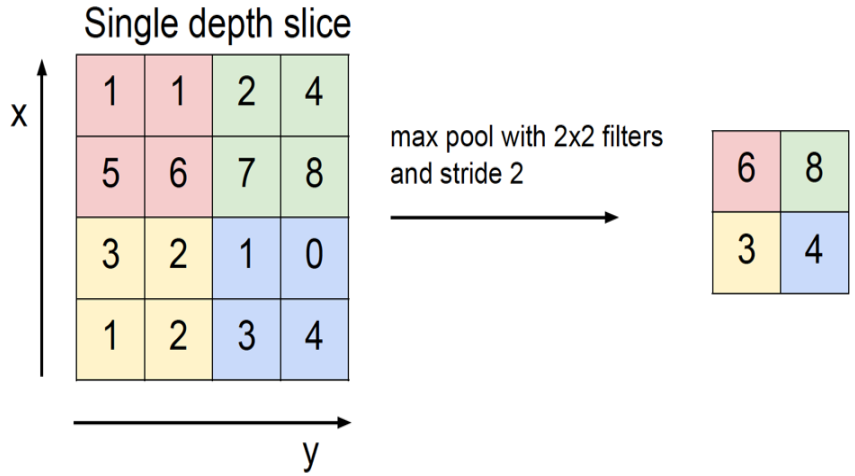
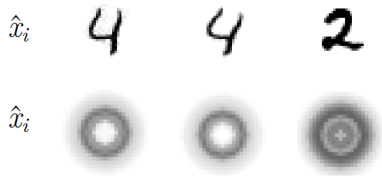


figure from slides of Andrej Karpathy

Average pooling or maxpooling shrinks the representations.

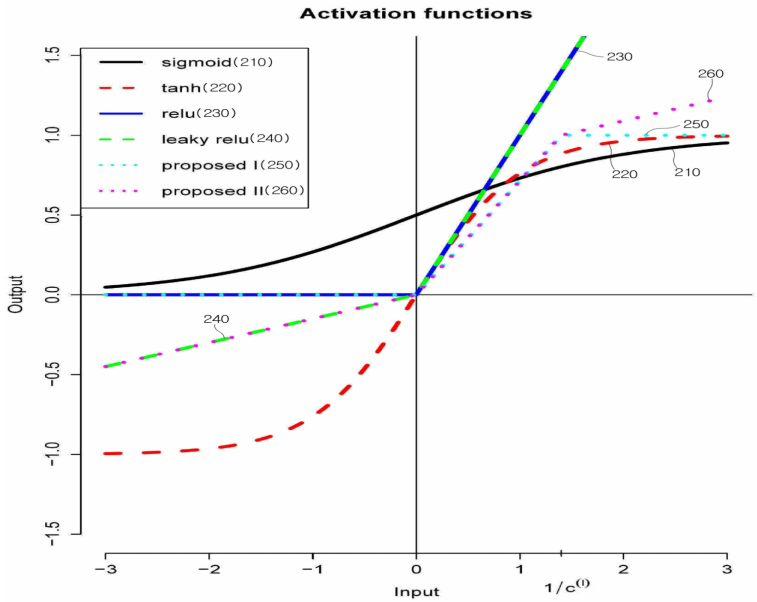
Recall averaging or integration can extract invariant features of the images.



Integration over all rotations

Activation layer

- $sigm(Z) = \frac{1}{1+\exp(-Z)}$
- $tanh(Z)$
- Rectified Linear Unit: ReLU(Z)= $max(Z, 0)$

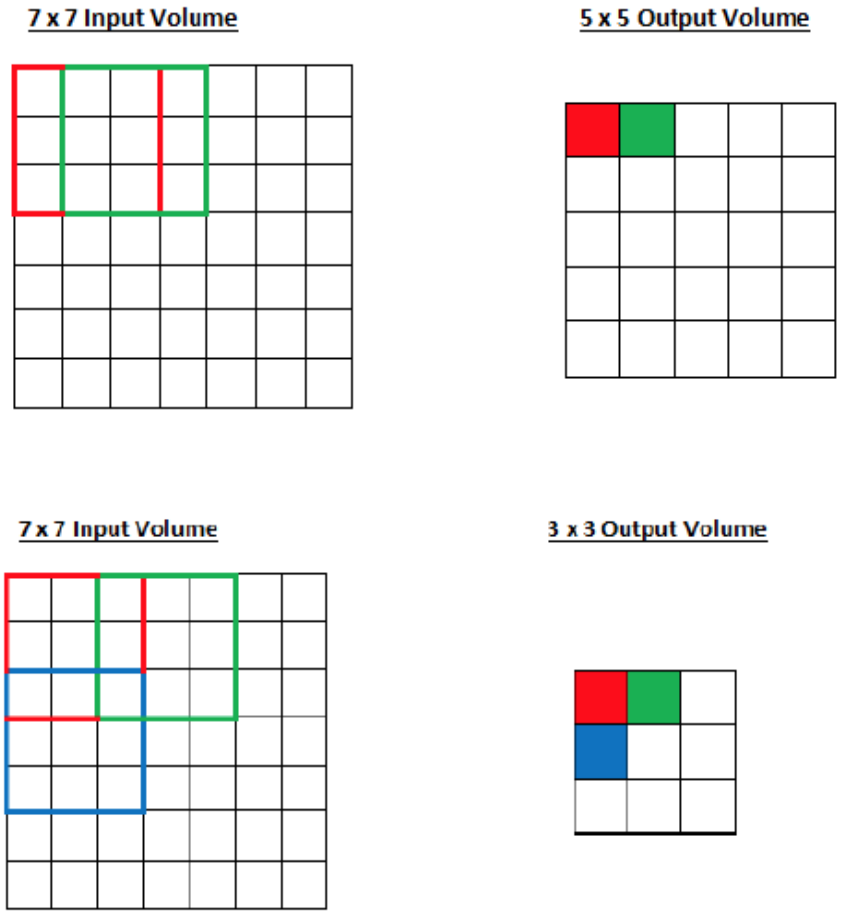


Stacked layers

- The first layer:
  - $Z^1 = W^1 h^0 + b^1$ where $h^0 = x$.
  - $h^1 = g_1(Z^1)$, $g_1(.)$ is activation function
- The l^{th} layer:
  - $Z^l = W^l h^{l-1} + b^l$
  - $h^l = g_l(Z^l)$

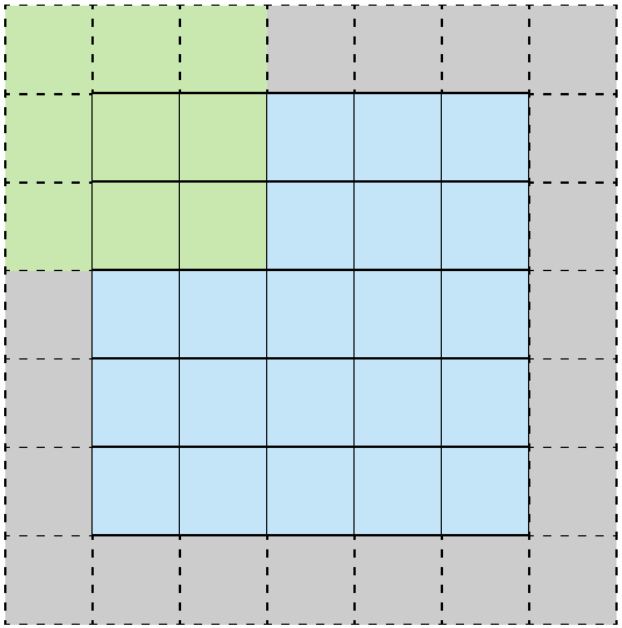
Stride

- Shrink dimensions by subsampling.

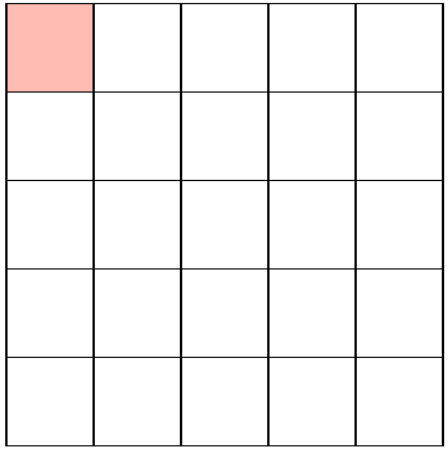


Source:http://adeshpande4.github.io/A-Beginner%

Padding



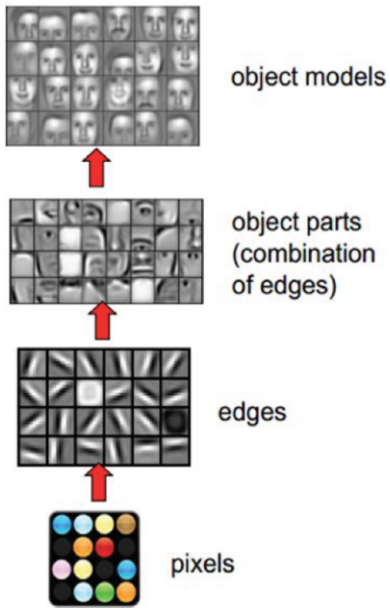
Stride 1 with Padding

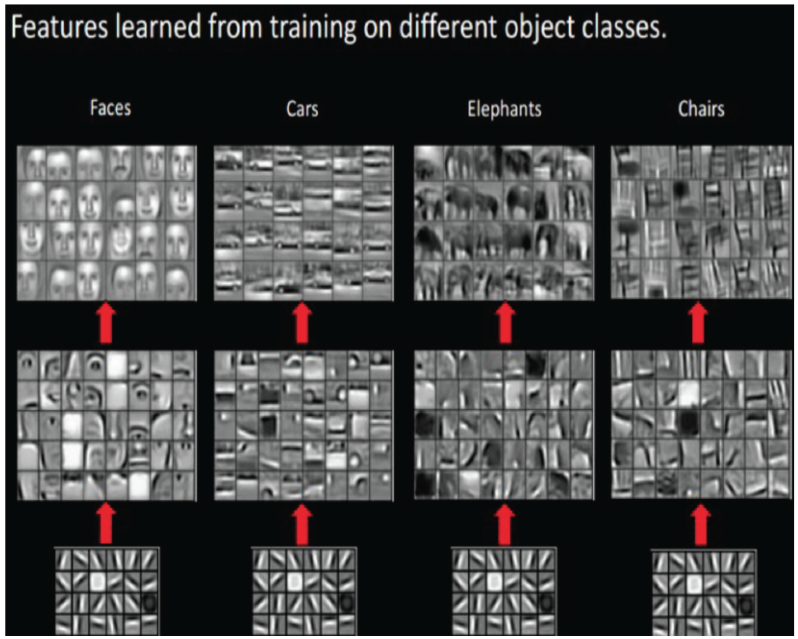


Feature Map

Source:https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2%

Role of multiple layers via visualization

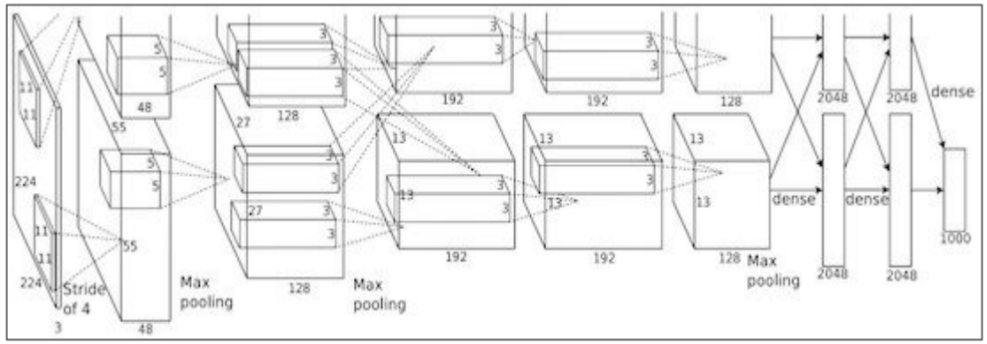




Different architectures

- CNNs popularity is triggered by debut of 'AlexNet' by Krizhevsky et al. (2012) winning ImageNet Large Scale Visual Recognition Challenge (ILSVRC).
- Imagenet competition is an annual computer vision contest running since 2010 after Li launched ImagNet assembling a free database of 14 million+ labeled images.
- Successful training is due to a large dataset, computational power using GPU and some aspects of the algorithm.
- Every year through ImageNet competition new architecture and optimization tips have been proposed and improved the accuracy of classification. We cover AlexNet, VGGNet and ResNet.

AlexNet by Krizhevsky et al. (2012)



Start with 224x224x3 input. End with three fully connected layers.

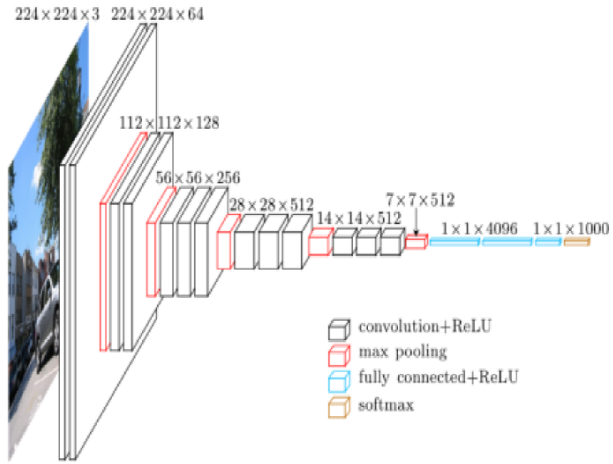
layer	Filter size (stride)	# filters	maxpool (stride)	output
1.1	11x11x3 (4)	48x2		55x55x96
1.2			3x3 (2)	27x27x96
2.1	5x5x96	128x2		27x27x256
2.2			3x3 (2)	13x13x256
3	3x3x256	192x2		13x13x384
4	3x3x384	192x2		13x13x384
5.1	3x3x384	128x2		13x13x256
5.2			3x3 (2)	6x6x256=9216

AlexNet (Krizhevsky et al. 2012)

- Used ReLu
- Heavy data augmentation
- Dropout
- SGD, batch size 128, momentum=0.9, Reducing learning rate manually starting from 0.01.
- Ensemble of 7 CNNs

VGGNet, OxfordNet (Simonyan and Zisserman, 2014)

- Deeper model. More layers (16 layers excluding maxpool and softmax compared to 5 layers for AlexNet).
- Simpler structure.
  - Only 3x3 filters with stride 1, pad 1, and 2x2 maxpool with stride 2, are used.
  - Number of filters multiplied by two (64, 128, 256, 512)



Source: https://blog.heuritech.com/2016/02/29

VGGNet, OxfordNet (Simonyan and Zisserman, 2014)

Table: Structure of VGGNet

block	# cov or fully connected layers	# filter size	
1	2 conv 3x3	64	maxpool
2	2 conv 3x3	128	maxpool
3	3 conv 3x3	256	maxpool
4	3 conv 3x3	512	maxpool
5	3 conv 3x3	512	maxpool
6	3 Fully connected	4096 (2) 1000 (1)	softmax

- maxpool after each block
- 140M parameters (heavy from FC layers)

VGGNet: Number of parameters and memory

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

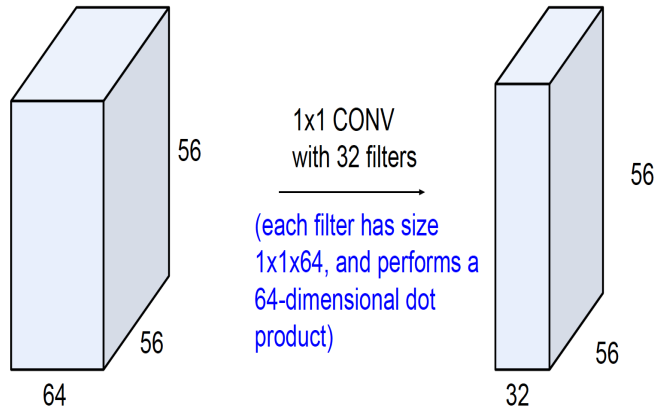
ConvNet Configuration		
B	C	D
13 weight layers	16 weight layers	16 weight layers
put (224 × 224 RGB image)		
conv3-64	conv3-64	conv3-64
conv3-64	conv3-64	conv3-64
maxpool		
conv3-128	conv3-128	conv3-128
conv3-128	conv3-128	conv3-128
maxpool		
conv3-256	conv3-256	conv3-256
conv3-256	conv3-256	conv3-256
	conv1-256	**conv3-256**
maxpool		
conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512
	conv1-512	**conv3-512**
maxpool		
conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512
	conv1-512	**conv3-512**
maxpool		
FC-4096		
FC-4096		
FC-1000		
soft-max		

Role of a small filter

- If we stack two 3x3 convolutional layers, a neuron in the second layer will cover 5x5 input region.
- If we stack three 3x3 convolutional layers, a neuron in the third layer will cover 7x7 input region.
- If the number of filters is C: 7x7 filter needs $Cx(7x7xC)$ parameters; three 3x3 filters need $3xCx(3x3xC)$. Three 3x3 filters need less parameters with more nonlinearity.
- How about even a smaller filter?

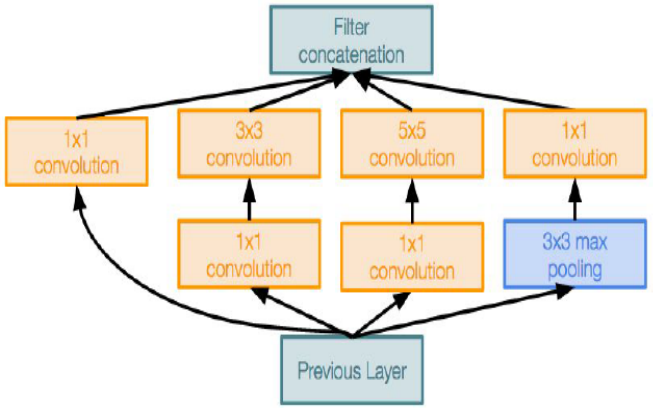
Role of a 1x1 filter

- For a HxWxC input dimension, 1x1x(C/2) filters output HxWx(C/2). (with stride 1 and padding to preserve H, W)
- (1. 1x1x(C/2) 2. 3x3x(C/2) 3. 1x1xC) vs. single 3x3xC? The former needs less number of parameters, less computation, with more nonlinearity.

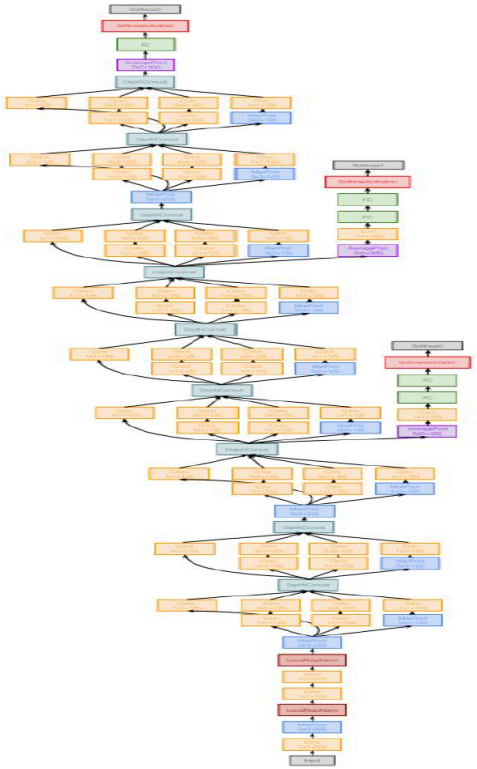


GoogLeNet (Szegedy et al., 2014)

- Design a good local network topology and stack these modules.
- Use of average pooling before the classification
- Computationally expensive
- Auxiliary classifiers connected to intermediate layers



Inception module



ResNet (He, Zhang, Ren and Sun, 2015)

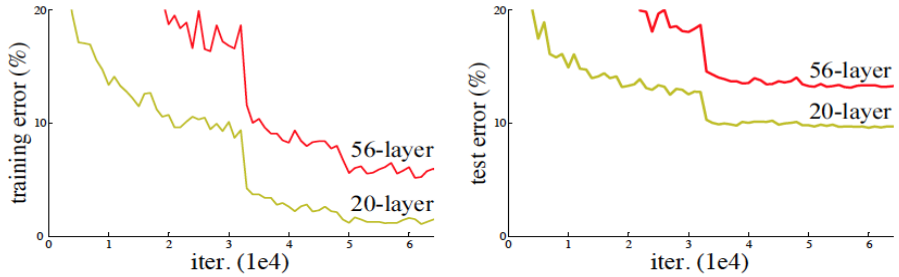


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

- Deeper the better? He et al. (2015) showed that deeper models can have higher training error than shallower models.
- Instead of $f_2(f_1(xw_1)w_2)$ as in Alexnet or VGGNet, ResNet models the residual, i.e., $f_1(xw_1) + f_2(f_1(xw_1)w_2)$ so that $w_2 = 0$ reduces to a shallow model.

ResNet

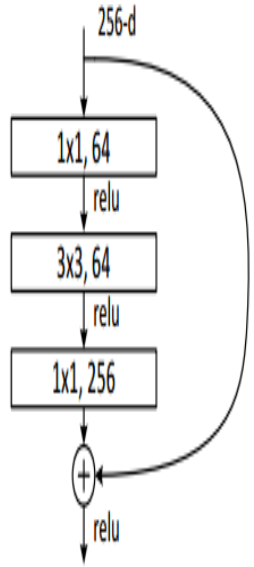
- 152-layer model
- Every residual block has 3x3 conv layers
- Periodilcally, double the

number of filters and downsample spatially using stride 2

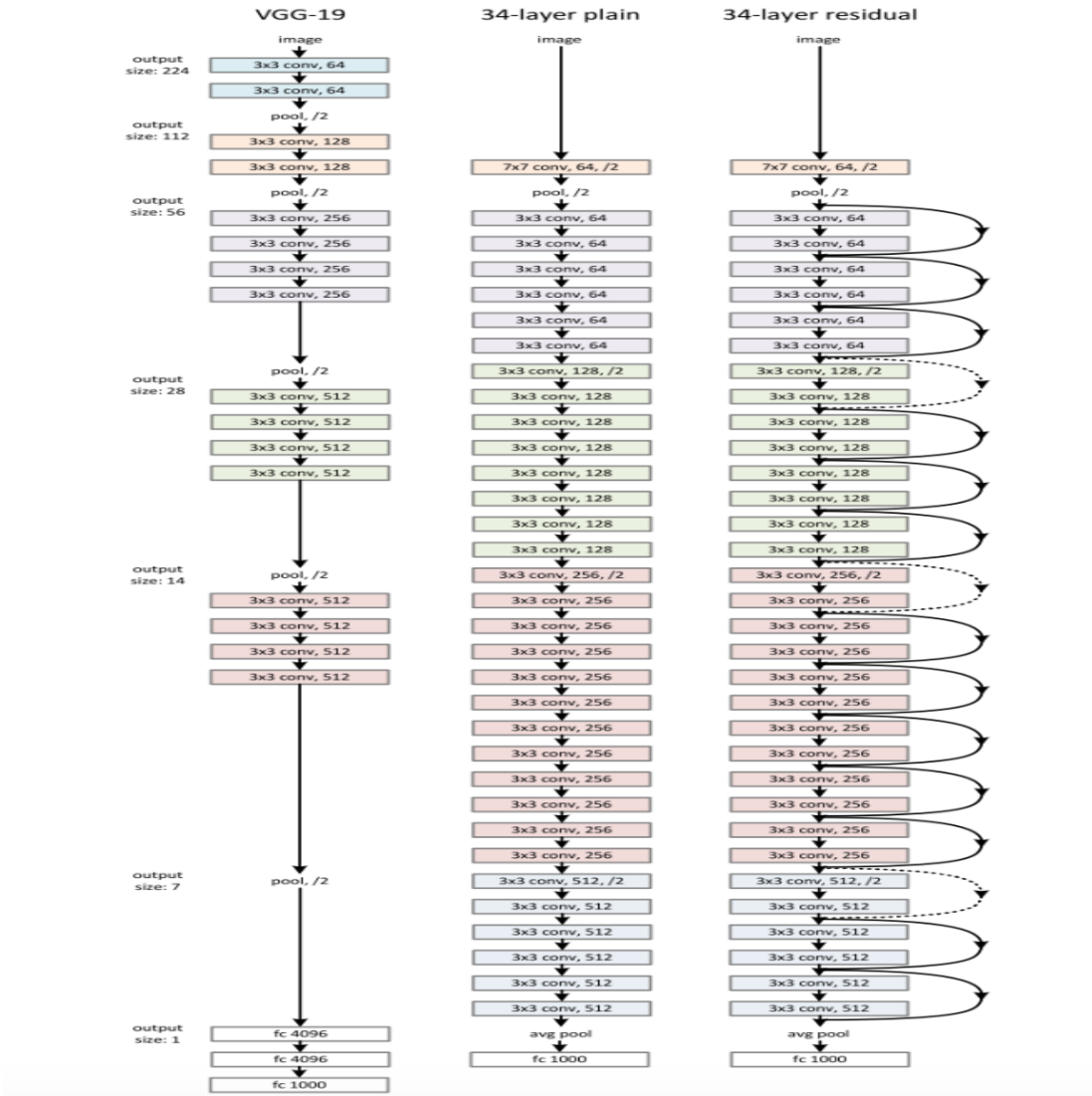
- Additional conv layer at the beginning
- No FC layers at the end
- For deeper networks (50+ layers) use

bottleneck layer to improve efficiency: 1x1→ 3x3 → 1x1

- No dropout
- Batch normalization • No maxpooling



ResNet (He, Zhang, Ren and Sun, 2015)



Performance of various architectures

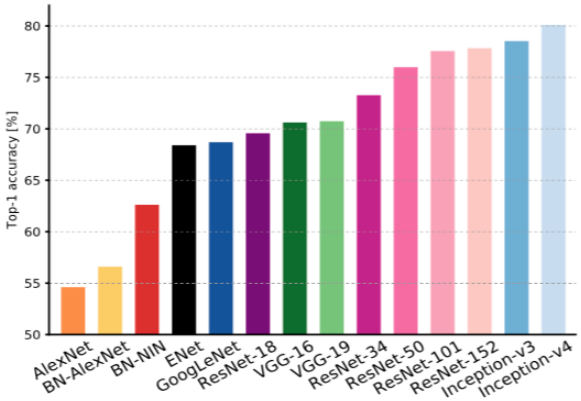


Figure 1: **Top1 *vs.* network.** Single-crop top-1 validation accuracies for top scoring single-model architectures. We introduce with this chart our choice of colour scheme, which will be used throughout this publication to distinguish effectively different architectures and their correspondent authors. Notice that networks of the same group share the same hue, for example ResNet are all variations of pink.

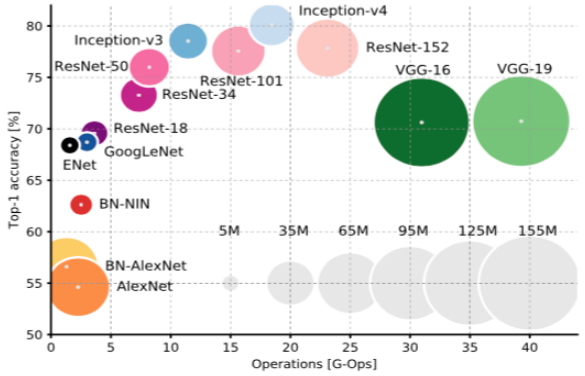


Figure 2: **Top1 *vs.* operations, size $\propto$ parameters.** Top-1 one-crop accuracy versus amount of operations required for a single forward pass. The size of the blobs is proportional to the number of network parameters; a legend is reported in the bottom right corner, spanning from 5×10^6 to 155×10^6 params. Both these figures share the same y-axis, and the grey dots highlight the centre of the blobs.

source: Canziani, Culuciello and Paszke (2017)

Regularizations

- In most cases, the number of parameters exceeds the number of training samples. To avoid overfitting, some regularization is necessary.
- ReLU (non-negative thresholding operator)
- Early stopping
- L_1, L_2 penalty on weights
- Dropout
- Batch normalization
- Data augmentation
- Ensemble

Optimization

Generalization error

- Training error: $n^{-1}\sum_{i=1}^{n} L(y_i, \hat{f}(x_i))$, where $\hat{f}(x_i)$ is fitted value using the training data set, $\{x_i, y_i\}_{i=1}^{n}$, denoted by $\mathcal{T}$.
- The goal is to minimize generalization error: $E_{y^{NEW}} E_{(x,y)\in\mathcal{T}}\{L(y^{NEW}, \hat{f}(x))\}$.
- Data are split to training data, validation data and test data.
- The loss function represents training error. A generalization error estimated through a validation set is used to decide a stopping rule. After training, a test dataset is used to estimate another generalization error.

Elements of training

- High dimensional, nonconvex loss function due to composite link functions or deep structure brings issues that are not present in traditional estimating procedures in statistics.
- Stochastic loss function
- Optimization
  - Solving the first derivative equals zero (Back propagation)
  - Stochasitc gradient descent (SGD); minibatch; Learning rate
  - Generalization error; Validation set
- Regularization: Dropout; Some preprosessing.

Training using minibatch

- The problem of a large number of parameters is more challenging due to nonconvexity induced by composite link functions. Due to this difficulties, the optimization routines are different from usual statistical model fitting.
- Optimization technique is stochastic through using minibatch: Stochasitc gradient descent (SGD)

Optimization: Back propagation

- Training samples, $\{x_i(0), y_i\}_{i=1}^n$. Output of the network: $\mu_i = f_L(\cdots f_3(f_2(f_1(x_i; \xi_1); \xi_2); \xi_3) \cdots ; \xi_L)$, $\xi_l = (w_l, b_l)$.
- In binary classification with softmax, consider minimizing loss function $L(\{w_l\}, \{b_l\}) = -n^{-1} \sum_{i=1}^n \{y_i \log \frac{\mu_i}{1-\mu_i} + \log(1-\mu_i)\}$. Let $\theta_i = \log \frac{\mu_i}{1-\mu_i}$. Then

$$L(\{w_l\}, \{b_l\}) = -n^{-1} \sum_{i=1}^n \{y_i \theta_i - \log(1 + \exp(\theta_i))\},$$

and

$$\frac{\partial L}{\partial \theta_i} = \sum_{i=1}^n (y_i - \mu_i).$$

Optimization: Back propagation

- Denote $x_i(L) = f_{L-1}(x_i(L-1)w_{L-1})$,
  $x_i(l) = f_{l-1}(x_i(l-1)w_{l-1})$, $i = 1, \cdots, n$, $l = 1, \cdots, L$.
  $x_i(0)$ is input data.
  Let $\eta_i(l) = x_i(l)w_l$, $l = 1, \cdots, L$. Note that $\eta_i(L) = x_i(L)w_L = \theta_i$.

$$\frac{\partial L}{\partial w_L} = \sum_{i=1}^{n} \frac{\partial \theta_i}{\partial w_L} \frac{\partial L}{\partial \theta_i} = \sum_{i=1}^{n} \frac{\partial \theta_i}{\partial w_L}(y_i - \mu_i) = \sum_{i=1}^{n} x_i(L)(y_i - \mu_i),$$

$$\begin{aligned} \frac{\partial L}{\partial w_{L-1}} &= \sum_{i=1}^{n} \frac{\partial \theta_i}{\partial w_{L-1}}(y_i - \mu_i) \\ &= \sum_{i=1}^{n} \frac{\partial \eta_i(L)}{\partial x_i(L)^T} \frac{\partial x_i(L)}{\partial \eta_i(L-1)} x_i(L-1)(y_i - \mu_i) \end{aligned}$$

Optimization: Back propagation

- Note that $\frac{\partial x_i(L)}{\partial \eta_i(L-1)}$ depends on the activation function. When $f(.)$ is sigmoid, $\frac{\partial x_i(l)}{\partial \eta_i(l-1)} = f_{l-1}(1 - f_{l-1})$, and when l is small, the gradient may become very small.
- Updating formula for w_l and b_l are

$$w_l \leftarrow w_l - \epsilon \frac{\partial L}{\partial w_l}$$

$$b_l \leftarrow b_l - \epsilon \frac{\partial L}{\partial b_l}$$

- ϵ is a learning rate.

Difference in optimization between GLM and CNN

- Refreshment of Generalized Linear Models:

$$f_Y(y;\theta,\phi) = \exp\{(y\theta - b(\theta))/\phi + c(y,\phi)\}$$

$E(Y) = \mu = b'(\theta)$, $Var(Y) = b''(\theta)\phi$, $\eta = x\beta$, $\eta = g(\mu)$. Canonical link is the link function setting $\theta = \eta$.

- Score function of GLM:

$$\begin{aligned}
\frac{\partial}{\partial\beta} logL(\beta,\phi) &= \sum \frac{\partial\eta_i}{\partial\beta}\frac{\partial\theta_i}{\partial\eta_i}\frac{\partial}{\partial\theta_i} logL(\beta,\phi) \\
&= \sum \frac{\partial\eta_i}{\partial\beta}\frac{\partial\mu_i}{\partial\eta_i}\frac{\partial\theta_i}{\partial\mu_i}\frac{\partial}{\partial\theta_i} logL(\beta,\phi) \\
&= \sum x_i g'(\mu_i)^{-1}\{var(y_i|x_i)\}^{-1}(y_i - \mu_i)
\end{aligned}$$

- Since the last layer is softmax (logit) function, blue part remains the same in CNN and only red part changes.

Difference in optimization between GLM and CNN

- Hessian function of GLM:

$$\begin{aligned}\frac{\partial^2}{\partial\beta\partial\beta^T} logL(\beta,\phi) &= \sum[x_i\{g'(\mu_i)var(y_i|x_i)\}^{-1}]\frac{\partial}{\partial\beta^T}(y_i-\mu_i) \\ &+ \sum\frac{\partial}{\partial\beta^T}[\{x_i g'(\mu_i)var(y_i|x_i)\}^{-1}](y_i-\mu_i) \\ &= H_1 + H_2\end{aligned}$$

  where $H_1 = \sum[x_i\{g'(\mu_i)^2 var(y_i|x_i)\}^{-1}]x_i^T$ is positive semi definite. H_2 is zero for logistic model. For non-canonical model, H_2 is nonzero and the source of negative eigenvalues.

- The loss surface of neural network is studied by examining H_1 and H_2 using random matrix theory (Pennington and Bahri, 2017).

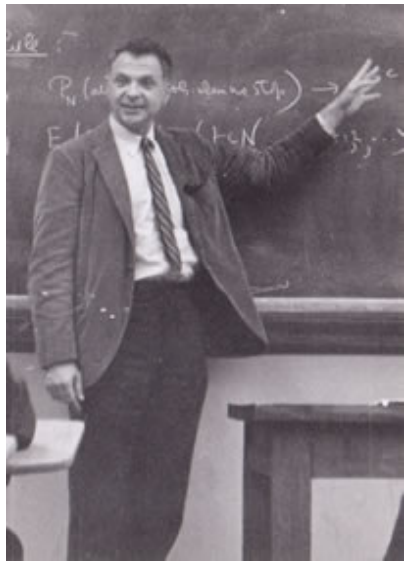
GLM vs. CNN

- Training samples, $\{x_i(0), y_i\}_{i=1}^n$. Output of the network: $\mu_i = f_L(\cdots f_3(f_2(f_1(x_i; \xi_1); \xi_2); \xi_3) \cdots ; \xi_L)$, $\xi_l = (w_l, b_l)$.
- When $(\xi_1, \cdots, \xi_{L-1})$ is known, the model is logit and the loss function is convex.
- The CNN can be viewed as generalized linear models with a compositional link function.
- Highdimensional GLM can be handled via regularization such as ridge or lasso. One of way to cope with highdimensional aspect of the CNN is using L_1 or L_2 penalty for w.

Stochastic gradient descent (Robbins and Monro, 1951)

Algorithm 8.1 Stochastic gradient descent (SGD) update

Require: Learning rate schedule $\epsilon_1, \epsilon_2, \ldots$
Require: Initial parameter $\boldsymbol{\theta}$
  $k \leftarrow 1$
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon_k \hat{\boldsymbol{g}}$
    $k \leftarrow k + 1$
  end while



source: http://www.deeplearningbook.org; photo from wikipedia

- A sufficient condition to guarantee convergence of SGD for monotone g, is $\sum_{k=1}^{\infty} \epsilon_k = \infty$ and $\sum_{k=1}^{\infty} \epsilon_k^2 < \infty$
- A common practice is to schedule $\epsilon_k = (1 - \alpha)\epsilon_0 + \alpha\epsilon_\tau$ with $\alpha = k/\tau$.

Gradient descent vs. Stochastic gradient descent

- Let $\sum_{i=1}^{n} f(z_i; w)$ be object function.
- Gradient descent: use all data

$$w_{t+1} \leftarrow w_t - \epsilon_t n^{-1} \sum_{i=1}^{n} \nabla f(z_i; w_t)$$

- Stochastic gradient descent: use one data point

$$w_{t+1} \leftarrow w_t - \epsilon_t \nabla f(z_i; w_t)$$

- Mini-batch SGD: use mini-batch

$$w_{t+1} \leftarrow w_t - \epsilon_t |B_t|^{-1} \sum_{i \in B_t} \nabla f(z_i; w_t)$$

Gradient descent converges for convex object

- Let $g(w) = n^{-1} \sum_{i=1}^{n} \nabla f(z_i; w)$. Assume that f is convex and differentiable, w^* being global optimum, and $\|g(w) - g(w')\|_2 \leq L\|w - w'\|_2$ for any w, w', i.e., the gradient is Lipschitz continuous with constant $L > 0$.
- Theorem: Gradient descent with fixed step size $\epsilon \leq 1/L$ satisfies

$$f(w^{(k)}) - f(w^*) \leq \frac{L\|w^{(0)} - w^*\|_2^2}{2k} \leq \frac{\|w^{(0)} - w^*\|_2^2}{2\epsilon k}.$$

  We say gradient descent has converge rate $O(1/k)$. To get $f(w^{(k)}) - f(w^*) \leq \delta$, we need $O(1/\delta)$ iterations.
- For given bound δ, k and ϵ are inversely related.

Convergence of GD for strong convexity

- When f is strongly convex, convergence rate is faster.
- Definition (Convex set): $C \subseteq \mathbb{R}^n$ such that $x, y \in C$ implies $tx + (1-t)y \in C$ for all $0 \leq t \leq 1$.
- Definition (Convex function): $f : \mathbb{R}^n \to \mathbb{R}$ such that $\mathrm{dom}(f) \subseteq \mathbb{R}^n$ convex, and $f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$ for $0 \leq t \leq 1$ and all $x, y \in \mathrm{dom}(f)$. If f is twice differentiable, $f(y) \geq f(x) + \nabla f(x)^T (y - x)$ for all $x, y \in \mathrm{dom}(f)$, equivalently, $\nabla^2 f(x) \succeq 0$ for all $x \in \mathrm{dom}(f)$.
- Definition (Strong convexity): f is strongly convex if $f(x) - \frac{\mu}{2}\|x\|_2^2$ is convex for some $\mu > 0$. If f is twice differentiable, it means

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\mu}{2}\|y - x\|_2^2$$

for all x, y, equivalently, $\nabla^2 f(x) \succeq \mu I$.

Convergence of GD for strong convexity

- When f is strongly convex, convergence rate is faster.
- Theorem: Under Lipschitz condition and strong convexity, gradient descent with fixed step size $\epsilon \leq 2/(\mu + L)$ satisfies

$$f(w^{(k)}) - f(w^*) \leq \frac{c^k L}{2} \|w^{(0)} - w^*\|_2^2,$$

  where $0 < c < 1$. Rate of convergence is $O(c^k)$, exponentially fast. We need $O(\log(1/\delta))$ iterations to get $f(w^{(k)}) - f(w^*) \leq \delta$.
- c is inversely related to condition number L/μ. ($c \downarrow$ and condition number $\uparrow$ implies slow convergence)

Conditions in simple case

- Consider $f(\beta) = \frac{1}{2}\|y - x\beta\|_2^2$.
- Lipschitz continuity of $\nabla f = g$ is $\nabla^2 f(x) \preceq LI$. That is $\nabla^2 f(x) = x^T x$ and L is the largest eigenvalue of $x^T x$, $L = \lambda_{\max}(x^T x)$.
- Strong convexity is $\nabla^2 f(x) \succeq \mu I$. That is $\nabla^2 f(x) = x^T x$ and $\mu = \lambda_{\min}(x^T x)$.
  - When $p > n$, $\lambda_{\min}(x^T x) = 0$ and f cannot be strongly convex.
  - In ill-conditioned cases where $\lambda_{\min}(x^T x)$ is small, the condition number is big and convergence is slow.
- If f has Lipschitz gradient and is strongly convex, $\mu I \preceq \nabla^2 f(x) \preceq LI$ for all x.

GD under Lipschitz gradient and strong convexity

- Assume $\mu I \preceq \nabla g(w_t) \preceq LI$, where $g(w) = n^{-1} \sum_{i=1}^{n} \nabla f(z_i; w_t)$.

$$\begin{aligned} w_{t+1} - w^* &= w_t - w^* - \epsilon(g(w_t) - g(w^*)) = w_t - w^* - \epsilon \nabla g(z_t)(w_t - w^*) \\ &= (I - \epsilon \nabla g(z_t))(w_t - w^*). \end{aligned}$$

- Taking the norm

$$\|w_{t+1} - w^*\| \leq \|I - \epsilon \nabla g(z_t)\| \|w_t - w^*\| \leq max(|1 - \epsilon\mu|, |1 - \epsilon L|) \|w_t - w^*\|$$

Setting $\epsilon = 2/(L + \mu)$,

$$||w_{t+1} - w^*|| \leq \frac{L - \mu}{L + \mu} ||w_t - w^*||.$$

Recursively, $||w_\tau - w^*|| \leq \left(\frac{L-\mu}{L+\mu}\right)^\tau ||w_0 - w^*||$

Convergence of SGD

- With SGD, $w_{t+1} = w_t - \epsilon_t g_i(w_t)$. Due to randomness, we find a bound for the expected difference.
- Assume f is convex and $E(\|g_i(w)\|^2) \leq G^2$.

$$
\begin{aligned}
&E(\|w_{t+1} - w^*\|_2^2 | w_t) \\
&= \|w_t - w^*\|_2^2 - 2\epsilon_t E(\langle g_i(w_t), w_t - w^* \rangle | w_t) + \epsilon_t^2 E(\|g_i(w_t)\|^2 | w_t) \\
&\leq \|w_t - w^*\|_2^2 - 2\epsilon_t E(f(w_t) - f(w^*) | w_t) + \epsilon_t^2 E(\|g_i(w_t)\|^2 | w_t)
\end{aligned}
$$

- Taking marginal expectation

$$
E(\|w_{k+1} - w^*\|_2^2) \leq E(\|w_k - w^*\|_2^2) - 2\epsilon_t E(f(w_t) - f(w^*)) + \epsilon_t^2 G^2
$$

Convergence of SGD

- Applying the inequality recursively

$$E(\|w_{k+1}-w^*\|_2^2) \leq \|w_0-w^*\|_2^2 - 2\sum_{t=1}^{k}\epsilon_t E(f(w_t)-f(w^*)) + \sum_{t=1}^{k}\epsilon_t^2 G^2$$

- Using $E\|w_{k+1}-w^*\|_2^2 \geq 0$ and letting $R^2 = \|w_0-w^*\|_2^2$,

$$0 \leq R^2 - 2\sum_{t=1}^{k}\epsilon_t E(f(w_t)-f(w^*)) + G^2\sum_{t=1}^{k}\epsilon_t^2$$

- Since $k^{-1}\sum f(w_t) \geq f(\bar{w})$

$$E(f(\bar{w})-f(w^*)) \leq \frac{R^2 + G^2\sum_{i=1}^{k}\epsilon_i^2}{2\sum_{i=1}^{k}\epsilon_i}$$

Convergence rate of SGD

- Alternatively,

$$min_t E(f(w_t) - f(w^*)) \leq \frac{R^2 + G^2 \sum_{i=1}^{k} \epsilon_i^2}{2 \sum_{i=1}^{k} \epsilon_i}$$

- With fixed step size ϵ,

$$E(f(\bar{w}) - f(w^*)) \leq \frac{R^2}{2k\epsilon} + \frac{G^2 \epsilon}{2}.$$

  If we bound $\frac{R^2}{2k\epsilon} \leq \frac{\delta}{2}$ and $\frac{G^2\epsilon}{2} \leq \frac{\delta}{2}$, then $\epsilon = \delta/G^2$, and $k = R^2/(\epsilon\delta) = (R^2G^2)/\delta^2$.

- To achieve the expected error smaller than δ, one needs $O(1/\delta^2)$ iterations. (cf. $O(1/\delta)$ for gradient descent)

Mini-batch application of SGD

- Updating computations, $w_l \leftarrow w_l - \epsilon_k \frac{\partial L}{\partial w_l}$, $b_l \leftarrow b_l - \epsilon_k \frac{\partial L}{\partial b_l}$ are conducted in minibatch.
- Minibatch should be selected randomly.
- When using GPUs, it is common for power of 2 minibatch sizes to offer better runtime. Typical power of 2 batch sizes range from 32 to 256.

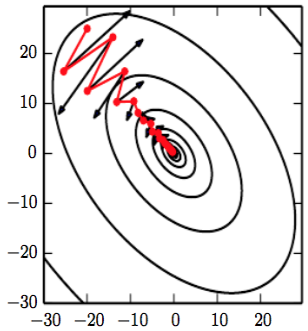
Stochastic gradient descent in simple case

- In high dimensional convex case, consider $y = x\beta$ $y \in \mathbb{R}^n$, $\beta \in \mathbb{R}^p$, $p >> n$. There are many solutions for β that exactly satisfies $y = x\beta$. A particular solution is $\hat{\beta} = x^T(xx^T)^{-1}y$, which minimizes $||\beta||$.
- To see this, for any β that satisfies $y = x\beta$, $x(\beta - \hat{\beta}) = 0$, and $(\beta - \hat{\beta})^T\hat{\beta} = 0$. That is $||\beta||^2 = ||\beta - \hat{\beta} + \hat{\beta}||^2 = ||\beta - \hat{\beta}||^2 + ||\hat{\beta}||^2 \geq ||\hat{\beta}||^2$.
- SGD converges to $\hat{\beta}$ when initial value is chosen as 0.
- A regularized solution $\tilde{\beta} = (x^Tx + \lambda I)^{-1}x^Ty$ converges to $\hat{\beta}$ as $\lambda \to 0$.

SGD with momentum

Algorithm 8.2 Stochastic gradient descent (SGD) with momentum

Require: Learning rate ϵ, momentum parameter α
Require: Initial parameter $\boldsymbol{\theta}$, initial velocity $\boldsymbol{v}$
while stopping criterion not met **do**
  Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
  Compute gradient estimate: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$.
  Compute velocity update: $\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \boldsymbol{g}$.
  Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$.
end while



source: http://www.deeplearningbook.org

- The velocity v accumulates the gradient elements. A larger α weighs more on the previous gradients.

SGD with Nesterov momentum

- Momentum:
  $v_{t+1} \leftarrow \alpha v_t - \epsilon g(\theta_t)$
  $\theta_{t+1} \leftarrow \theta_t + v_{t+1}$.
  Combined: $\theta_{t+1} \leftarrow \theta_t + \alpha(\theta_t - \theta_{t-1}) - \epsilon g(\theta_t)$.
- Nesterov momentum (Nesterov, 1983):
  $v_{t+1} \leftarrow \alpha v_t - \epsilon g(\theta_t + \alpha v_t)$
  $\theta_{t+1} \leftarrow \theta_t + v_{t+1}$.
  Combined: $\theta_{t+1} \leftarrow \theta_t + \alpha(\theta_t - \theta_{t-1}) - \epsilon g(\theta_t - \alpha(\theta_t - \theta_{t-1}))$.
  In convex batch gradient case, Nesterov momentum accelerate the convergence of of the excess error, not in SGD.
- Many adaptive learning rates methods, AdaGrad, RMSProp, Adam etc.

SGD with AdaGrad, RMSProp, Adam

- AdaGrad:

  $r \leftarrow r + g \odot g$

  $\theta \leftarrow \theta - \frac{\epsilon}{\delta + \sqrt{r}} \odot g$

- RMSProp:

  $r \leftarrow \rho r + (1 - \rho) g \odot g$

  $\theta \leftarrow \theta - \frac{\epsilon}{\sqrt{\delta + r}} \odot g$

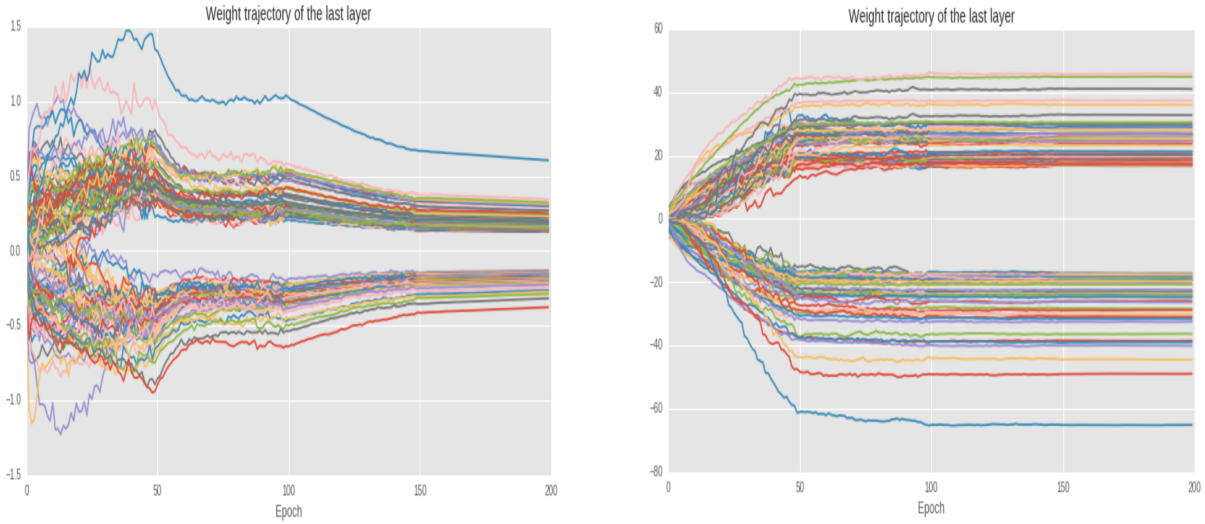
- Adam:

  $s \leftarrow \rho_1 s + (1 - \rho_1) g;\ r \leftarrow \rho_2 r + (1 - \rho_2) g \odot g$

  $\hat{s} \leftarrow \frac{s}{1 - \rho_1^t};\ \hat{r} \leftarrow \frac{r}{1 - \rho_2^t}$

  $\theta \leftarrow \theta - \frac{\epsilon \hat{s}}{\delta + \sqrt{\hat{r}}}$

SGD with Adam



source: Yongchan Kwon

Weights of the last layer using CIFAR10 traced for 200 epochs using SGD and Adam.

SGD for nonconvex functions

- One can apply the same way using minibatch. However, it does not go towards optimum.
- Assume second-differentiable objective, Lipschitz-continuous gradients and bounded variance. The same set of assumptions except convexity.
- Under these assumptions, using similar derivations, convergence to a stationary point can be obtained. (not to a local minimum)
- Under stronger conditions, SGD can be shown to converge to a local minimum (Ge et al. 2015).
- It is unclear whether neural networks satisfy these conditions.

Initial values

- Random initialization may play a role in convergence of gradient descent in the nonconvex case (Lee et al. 2016).
- Mishkin and Matas (2016) 'All you need is a good init'
- Glorot and Bengio (2010), 'Havier' initialization:
  $W_{ij} \sim U(-\sqrt{\frac{6}{m+n}}, \sqrt{\frac{6}{m+n}})$ where m, n are input, output dimensions.
- He, Zhang, Ren and Sun (2015): Design initialization so that output of each layer has unit variance. For layer l, $W_{ij} \sim N(0, 2/n_l)$

Regularizations

- In most cases, the number of parameters exceeds the number of training samples. Regularization is necessary.
- Explicit regularization: L_1, L_2 penalty on weights
- Implicit regularization
  - ReLU
  - Early stopping
  - Dropout
  - Batch normalization
  - Data augmentation
  - Ensemble

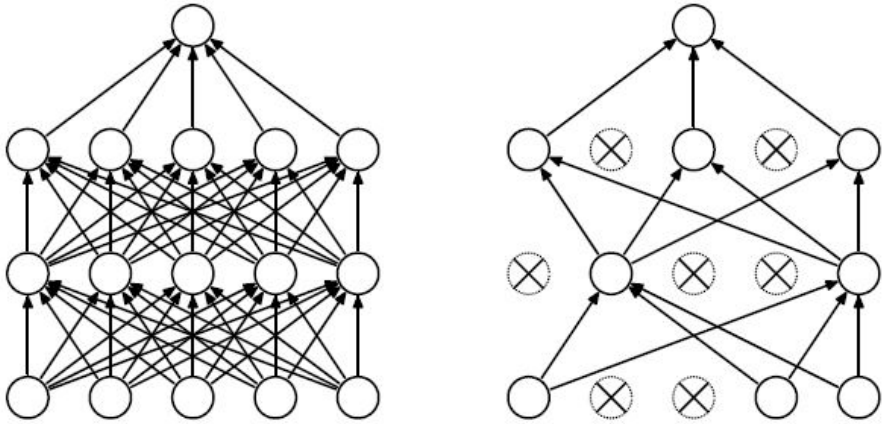
Explicit regularizations: L_1, L_2 penalty on weights

- L_2 regularization: $n^{-1}\sum_{i=1}^{n} L(y_i, f(x_i; w)) + \lambda ||w||_2^2$
- L_1 regularization: $n^{-1}\sum_{i=1}^{n} L(y_i, f(x_i; w)) + \lambda ||w||_1$
- Regularization can in interpreted as Bayesian prior.
- In simple cases, regularization parameter can be chosen by cross-validation. In deep learning, ad hoc method such as checking the test error with validation set is used.

Dropout

- Dropout randomly drops units from the neural network during training.
  ($\Leftrightarrow$ Dropout *samples* from an exponential number of "thinned" networks.)
- Dropout approximates the average of predictions by using the unthinned network with smaller weights.

Dropout



(a) Standard Neural Net (b) After applying dropout.

Each unit is retained with a fixed probability p, independent of other units. (Usually, $p = 0.5$ for hidden unit.)

- Applying dropout amounts to sampling one of 2^n possible "thinned" networks.
- All the 2^n "thinned" networks share weights. Total number of parameters is still $O(n^2)$.
- At test time, the unthinned network with weights multiplied by p is used for prediction $\Rightarrow$ The *expected* output of any hidden unit is the same as the actual output at test time.

Dropout: Model and Learning

- Standard neural network:

$$\eta_i^{(l+1)} = \mathbf{w}_i^{(l+1)} \mathbf{x}^l + b_i^{(l+1)},$$
$$x_i^{(l+1)} = f(\eta_i^{(l+1)})$$

$l \in \{1, 2, \cdots, L\}$: hidden layer index, f : any activation function

- With Dropout:

$$r_j^{(l)} \sim \text{Bernouilli}(p),$$
$$\tilde{\mathbf{x}}^{(l)} = \mathbf{r}^{(l)} * \mathbf{x}^{(l)},$$
$$\eta_i^{(l+1)} = \mathbf{w}_i^{(l+1)} \tilde{\mathbf{x}}^l + b_i^{(l+1)},$$
$$x_i^{(l+1)} = f(\eta_i^{(l+1)})$$

Dropout: Model and Learning

- Dropout neural networks can also be trained using stochastic gradient descent.
- For each training case in a mini-batch, we sample a thinned network. (Any training case which does not use a parameter contributes a gradient of zero for that parameter.)
- The noise provided by dropout may allow optimization process to explore different regions of the weight space that would otherwise been difficult to reach.

Dropout: Marginalizing dropout

Dropout in linear regression:

$$L(\mathbf{w}) = ||\mathbf{y} - X\mathbf{w}||^2$$

With dropout ($R_{ij} \sim \text{Ber}(p)$), the marginalized loss function is

$$\begin{aligned} E_R\left[||\mathbf{y} - R * X\mathbf{w}||^2 \middle| \mathbf{y}, X; \mathbf{w}\right] &= ||\mathbf{y} - pX\mathbf{w}||^2 + p(1-p)||\Gamma\mathbf{w}||^2 \\ &= ||\mathbf{y} - X\tilde{\mathbf{w}}||^2 + \frac{1-p}{p}||\Gamma\tilde{\mathbf{w}}||^2 \end{aligned}$$

where $\Gamma = (diag(X^T X))^{1/2}$, $\tilde{\mathbf{w}} = p\mathbf{w}$.

$\Rightarrow$ Ridge regression !!

Dropout: Extensions

- Dropout multiplies hidden units by Bernouilli random variable. This idea can be generalized to multiplying with random variable from other distributions!
- Multiplying by $r_g \sim N(1, \sigma^2)$ works just as well or even better. $h_i * r_g$ has the same distribution as $h_i + \varepsilon$ where $\varepsilon \sim N(0, h_i^2 \sigma^2)$. Since $E(h_i * r_g) = h_i$, no weight scaling is required at test time.
- Dropout does not require weight scaling as well if we multiply h_i by $r_b \sim \frac{1}{p}\text{Ber}(p)$.
- If we set $\sigma^2 = (1-p)/p$,

$$E(r_b) = E(r_g) = 1$$
$$Var(r_b) = Var(r_g) = (1-p)/p$$

Batch normalization

- Training DNN is difficult since distribution of each layer's input changes from minibatch to minibatch. This slows down the training by requiring lower learning rates.
- Batch normalization layer involves unknown parameters.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

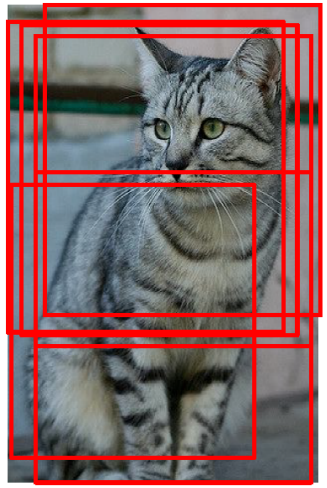
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Data augmentation: Random crops



- In training, sample random 224x224 patch
- In testing, use 10 224x224 crops: 4 corners + center and their flips

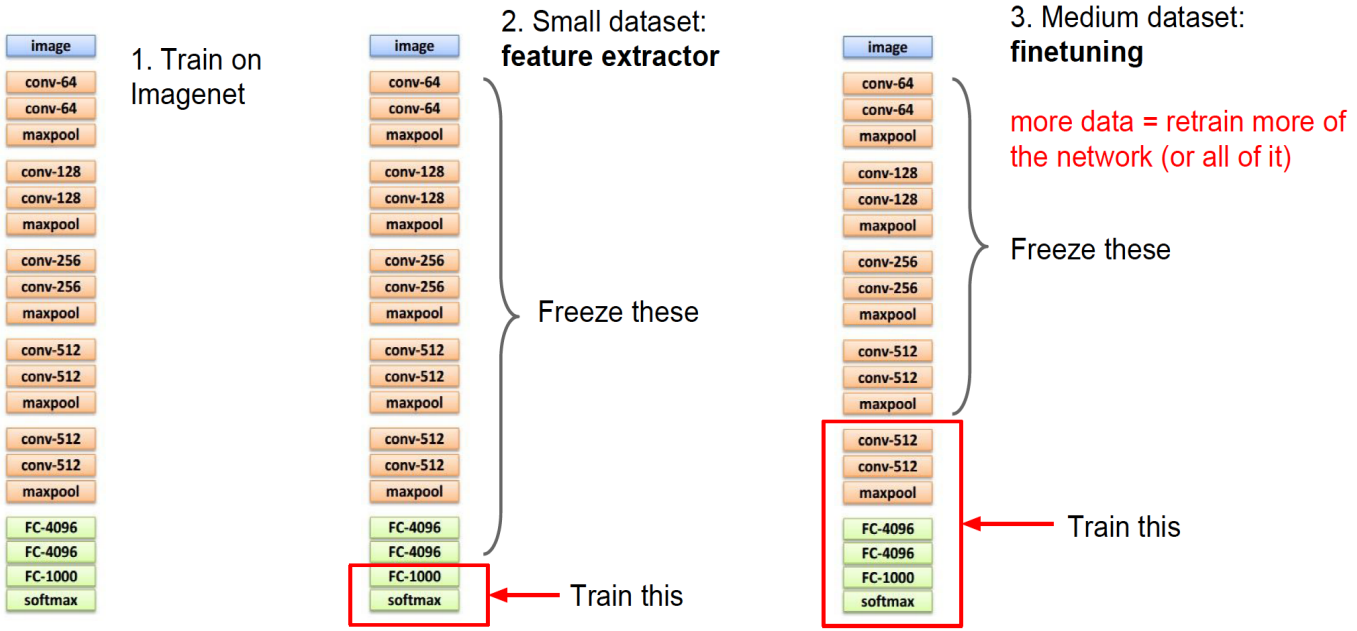
Data augmentation: Color jitter

- Add random error to one of the RGB channels.
- Apply PCA to all RGB pixels in training set $\rightarrow$ sample a color-offset along PC direction
- Add offset to all pixels of a training image

Explanation for data augmentation?

- Bayesian prior can be viewed as data augmentation. e.g. Ridge regression, padding zero cells in categorical data.
- Invariants: $\forall x \in \Omega$, $f(\phi(x)) = f(x)$. e.g. $\phi(x) = -x$, $f(-x) = f(x)$.
- Compositional model seeks invariant or covariant functions. By augmenting data by cropping, jittering, and flipping, some invariants and covariants can be easily identified.

Transfer learning



Slide from A. Karparthy, Bay area deep learning day

Practical issues

- Data preprocessing
- Monitoring optimization: checking training/validation loss, learning rate, updates
- Hyperparameter search