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 $T$.

- The goal is to minimize generalization error:
  $$E_{y^{NEW}}E_{(x,y) \in T}\{L(y^{NEW}, \hat{f}(x))\}.$$  

- 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.
Optimization

Elements of training

- High dimensional, nonconvex loss function due to \textit{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)
  - Stochastic gradient descent (SGD); minibatch; Learning rate
- Regularization: Dropout; Some preprocessing.
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: Stochastic gradient descent (SGD)
Optimization: Back propagation

- Training samples, \( \{x_i, y_i\}_{i=1}^n \). 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
  \[ \ell(\{w_l\}, \{b_l\}) = -n^{-1} \sum_{i=1}^n \{y_i \log \frac{\mu_i}{1-\mu_i} + \log(1 - \mu_i)\}. \]

Let \( \theta_i = \log \frac{\mu_i}{1-\mu_i} \) and \( f_L(a) = \frac{\exp(a)}{1+\exp(a)} \). Then

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

and

\[ \frac{\partial \ell}{\partial \theta_i} = \sum_{i=1}^n (y_i - \mu_i). \]
Optimization: Back propagation

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

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

\[
\frac{\partial \ell}{\partial w_{L-1}} = \sum_{i=1}^{n} \frac{\partial \theta_i}{\partial w_{L-1}} (y_i - \mu_i)
\]

\[
= \sum_{i=1}^{n} \frac{\partial \eta_i(L)}{\partial x_i(L)} \frac{\partial x_i(L)}{\partial \eta_i(L-1)} x_i(L-1)(y_i - \mu_i)
\]
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

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

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

- $\epsilon$ is a learning rate.
Difference in optimization between GLM and CNN

- Refreshment of Generalized Linear Models:

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

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

- Score function of GLM:

\[
\frac{\partial}{\partial \beta} \ell(\beta, \phi) = \sum \frac{\partial \eta_i}{\partial \beta} \frac{\partial \theta_i}{\partial \eta_i} \frac{\partial}{\partial \theta_i} \ell(\beta, \phi)
\]

\[
= \sum \frac{\partial \eta_i}{\partial \beta} \frac{\partial \mu_i}{\partial \eta_i} \frac{\partial \theta_i}{\partial \mu_i} \frac{\partial}{\partial \theta_i} \ell(\beta, \phi)
\]

\[
= \sum x_i g'(\mu_i)^{-1} \{\text{var}(y_i|x_i)\}^{-1}(y_i - \mu_i)
\]

- 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:**

\[
\frac{\partial^2}{\partial \beta \partial \beta^T} \ell(\beta, \phi) = \sum [x_i \{g'(\mu_i) \text{var}(y_i|x_i)\}^{-1}] \frac{\partial}{\partial \beta^T} (y_i - \mu_i)
\]

\[
+ \sum \frac{\partial}{\partial \beta^T} [\{x_i g'(\mu_i) \text{var}(y_i|x_i)\}^{-1}] (y_i - \mu_i)
\]

\[
= H_1 + H_2
\]

where \(-H_1 = \sum [x_i \{g'(\mu_i)^2 \text{var}(y_i|x_i)\}^{-1}] x_i^T\) 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:
  \[ \mu_i = f_L(\cdots f_3(f_2(f_1(x_i; \xi_1); \xi_2); \xi_3)\cdots; \xi_L), \quad \xi_l = (w_l, b_l) . \]
- 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.
- For GLM, \(-\ell\) is minimized using the Newton-Raphson method. \(-\ell\) is ‘deterministic’, and the Newton-Raphson method, a second-order optimization method, requires computing the Hessian.
- For CNN, the objective is ‘stochastic’ and a first-order optimization method is used.
Optimization

Gradient descent vs. Stochastic gradient descent

- Let $\sum_{i=1}^{n} f(z_i; w)$ be objective 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 (SGD): use one data point
  \[
  w_{t+1} \leftarrow w_t - \epsilon_t \nabla f(z_i; w_t)
  \]
- Mini-batch SGD: use mini-batch
  \[
  w_{t+1} \leftarrow w_t - \epsilon_t |B_t|^{-1} \sum_{i \in B_t} \nabla f(z_i; w_t)
  \]
Optimization

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 $\theta$

$k \leftarrow 1$

while stopping criterion not met do

Sample a minibatch of $m$ examples from the training set $\{x^{(1)}, \ldots, x^{(m)}\}$ with corresponding targets $y^{(i)}$.

Compute gradient estimate: $\hat{g} \leftarrow \frac{1}{m} \nabla \theta \sum_i L(f(x^{(i)}; \theta), y^{(i)})$

Apply update: $\theta \leftarrow \theta - \epsilon_k \hat{g}$

$k \leftarrow k + 1$

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 $\alpha = k/\tau$. 
Mini-batch application of SGD

- Updating computations, \( w_l \leftarrow w_l - \epsilon_k \frac{\partial L}{\partial w_l}, \quad b_l \leftarrow b_l - \epsilon_k \frac{\partial L}{\partial b_l} \) 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 $\beta$ 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 $\beta$ 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 $\epsilon$, momentum parameter $\alpha$
Require: Initial parameter $\theta$, initial velocity $v$

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

source: http://www.deeplearningbook.org

- The velocity $v$ accumulates the gradient elements. A larger $\alpha$ weighs more on the previous gradients.
SGD with Nesterov momentum

- Momentum:
  \[ v_{t+1} \leftarrow \alpha v_t - \varepsilon g(\theta_t) \]
  \[ \theta_{t+1} \leftarrow \theta_t + v_{t+1}. \]
  Combined: \[ \theta_{t+1} \leftarrow \theta_t + \alpha(\theta_t - \theta_{t-1}) - \varepsilon g(\theta_t). \]

- Nesterov momentum (Nesterov, 1983):
  \[ v_{t+1} \leftarrow \alpha v_t - \varepsilon g(\theta_t + \alpha v_t) \]
  \[ \theta_{t+1} \leftarrow \theta_t + v_{t+1}. \]
  Combined: \[ \theta_{t+1} \leftarrow \theta_t + \alpha(\theta_t - \theta_{t-1}) - \varepsilon g(\theta_t + \alpha(\theta_t - \theta_{t-1})). \]
  In convex batch gradient case, Nesterov momentum accelerate the convergence 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

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

source: Yongchan Kwon
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), ‘Xavier’ 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 may be 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 randomly drops units from the neural network during training.

(\Leftrightarrow \text{Dropout samples from an exponential number of “thinned” networks.})

Dropout approximates the average of predictions by using the unthinned network with smaller weights.
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.
Optimization

Dropout: Model and Learning

- **Standard neural network:**

  \[ \eta_i^{(l+1)} = w_i^{(l+1)} x_i^l + b_i^{(l+1)}, \]
  \[ x_i^{(l+1)} = f(\eta_i^{(l+1)}) \]

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

- **With Dropout:**

  \[ r_j^{(l)} \sim \text{Bernoulli}(p), \]
  \[ \tilde{x}^{(l)} = r^{(l)} \ast x^{(l)}, \]
  \[ \eta_i^{(l+1)} = w_i^{(l+1)} \tilde{x}^l + b_i^{(l+1)}, \]
  \[ x_i^{(l+1)} = f(\eta_i^{(l+1)}) \]
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:

\[ L(w) = ||y - Xw||^2 \]

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

\[
E_R[||y - R \ast Xw||^2|y, X; w] = ||y - pXw||^2 + p(1 - p)||\Gamma w||^2 \\
= ||y - X\tilde{w}||^2 + \frac{1 - p}{p}||\Gamma \tilde{w}||^2
\]

where \(\Gamma = (\text{diag}(X^T X))^{1/2}\), \(\tilde{w} = pw\).

⇒ Ridge regression !!


**Optimization**

**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.** $h_i \ast r_g$ has the same distribution as $h_i + \varepsilon$ where $\varepsilon \sim N(0, h_i^2 \sigma^2)$. Since $E(h_i \ast r_g) = h_i$, 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 $\sigma^2 = (1 - p)/p$,

  
  \[
  E(r_b) = E(r_g) = 1 \\
  \text{Var}(r_b) = \text{Var}(r_g) = (1 - p)/p
  \]
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: $B = \{x_1 ... m\}$; Parameters to be learned: $\gamma$, $\beta$

**Output:** $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

\[
\begin{align*}
\mu_B &\leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i & \text{// mini-batch mean} \\
\sigma_B^2 &\leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2 & \text{// mini-batch variance} \\
\hat{x}_i &\leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} & \text{// normalize} \\
y_i &\leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) & \text{// scale and shift}
\end{align*}
\]
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 → sample a color-offset along PC direction
- Add offset to all pixels of a training image
Comments on 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.
- Mixup has received a lot of attention as a data augmentation method (Zhang, Cisse, Dauphin and Lopez-Paz, 2018). No justification is available yet.
Transfer learning

1. Train on Imagenet
2. Small dataset: feature extractor
3. Medium dataset: finetuning

more data = retrain more of the network (or all of it)

Freeze these

Train this

Slide from A. Karparthy, Bay area deep learning day
Practical issues

- Data preprocessing
- Monitoring optimization: checking training/validation loss, learning rate, updates
- Hyperparameter search using the validation set
Convergence of Gradient descent and stochastic gradient descent methods

- Convergence of GD hinges on convexity and smoothness of a function.
- Assuming Lipchitz continuous gradient, we go over the following convergence analyses:
  - GD with convex objective
  - GD with strongly convex objective
  - SGD with convex objective
  - SGD with strongly convex objective
- Some reference regarding convergence result on DNN’s
Convexity

- Consider unconstrained, smooth convex optimization, $\min_x f(x)$.
- 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): $f : \mathbb{R}^n \rightarrow \mathbb{R}$ such that $\text{dom}(f) \subseteq \mathbb{R}^n$ convex, and $f(tx + (1 - t)y) \leq tf(x) + (1 - t)f(y)$ for $0 \leq t \leq 1$ and all $x, y \in \text{dom}(f)$. 1st-order condition: $f(y) \geq f(x) + \nabla f(x)^T(y - x)$ for all $x, y \in \text{dom}(f)$. 2nd-order condition: $\nabla^2 f(x) \succeq 0$ for all $x \in \text{dom}(f)$.
- Definition (Strong convexity): $f$ is strongly convex if $f(x) - \frac{\mu}{2} \|x\|^2_2$ is convex for some $\mu > 0$. 1st-order condition:
  \[ f(y) \geq f(x) + \nabla f(x)^T(y - x) + \frac{\mu}{2} \|y - x\|^2_2 \]
  for all $x, y \in \text{dom}(f)$. 2nd-order condition: $\nabla^2 f(x) \succeq \mu I$. 

Seoul National University

Deep Learning

September-December, 2019 35 / 54
Smoothness

- The $p^{th}$ derivative is Lipschitz continuous on $Q$ with the constant $L$:
  \[ \| f^{(p)}(x) - f^{(p)}(y) \| \leq L \| x - y \|, \quad \forall x, y \in Q \]

- (Quadratic upper bound) $\nabla f$ Lipschitz with $L$ implies
  \[ f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{L}{2} \| y - x \|^2. \]

- (Consequence of quadratic upper bound) If $f$ has a minimizer $x^*$,
  \[ \frac{1}{2L} \| \nabla f(x) \|^2 \leq f(x) - f(x^*) \leq \frac{L}{2} \| x - x^* \|^2. \]

RH by plugging in $y = x$, $x = x^*$; LH by $y = x - \frac{1}{L} \nabla f(x)$, $x = x$
then use $f(x^*) \leq f(x)$

\[ f(x^*) - f(x) \leq f(x - \frac{1}{L} \nabla f(x)) - f(x) \leq - \frac{1}{2L} \| \nabla f(x) \|^2 \]
Lipschitz gradient and strong convexity: Example

- Consider \( f(\beta) = \frac{1}{2}\|y - x\beta\|^2_2 \).
- L-Lipschitz continuity of \( \nabla f \) implies \( \nabla^2 f(x) = x^T x \preceq LI \), i.e., the largest eigenvalue of \( x^T x \), \( \lambda_{\max}(x^T x) \), is upper-bounded by \( L \).
- Strong convexity with twice differentiability gives \( \nabla^2 f(x) \succeq \mu I \). Since \( \nabla^2 f(x) = x^T x \), the minimum eigenvalue \( \lambda_{\min}(x^T x) \) is lower-bounded by \( \mu \).
  - 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 \( x \).
Gradient descent converges for convex objective

In the case of GD, let \( g(w) = \nabla f(w) \). 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 \( t \leq 1/L \) satisfies

\[
f(w_k) - f(w^*) \leq \frac{L\|w^{(0)} - w^*\|_2^2}{2k} \leq \frac{\|w^{(0)} - w^*\|_2^2}{2tk}.
\]

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 \( \delta \), \( k \) and \( t \) are inversely related.
Gradient descent converges for convex object: Proof

- \( \nabla f(x) \) Lipschitz with \( L \) implies

\[
f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{L}{2} \| y - x \|^2_2
\]

- Plugging in \( x = w_i, \ y = w_{i+1} = w_i - t \nabla f(w_i), \)

\[
f(w_{i+1}) \leq f(w_i) - (1 - \frac{Lt}{2})t \| \nabla f(w_i) \|^2_2
\]

- Taking \( 0 < t \leq \frac{1}{L} \) and 1st-order condition for convexity,

\[
f(w_{i+1}) \leq f(w^*) - \nabla f(w_i)^T (w^* - w_i) - \frac{t}{2} \| \nabla f(w_i) \|^2_2
\]
Gradient descent converges for convex object: Proof

- Replacing $\nabla f(w_i)$ with $-(w_i - w_{i+1})/t$,

$$f(w_{i+1}) \leq f(w^*) + \frac{1}{2t}(\|w_i - w^*\|^2_2 - \|w_{i+1} - w^*\|^2_2).$$

- Summing over iterations,

$$\sum_{i=1}^{k}(f(w_i) - f(w^*)) \leq \frac{1}{2t}\|w^{(0)} - w^*\|^2_2.$$

- Since $f(w_k)$ is nonincreasing

$$f(w_k) - f(w^*) \leq \frac{1}{k} \sum_{i=1}^{k}(f(w_i) - f(w^*)) \leq \frac{1}{2tk}\|w^{(0)} - w^*\|^2_2. \quad \square$$
When \( f \) is strongly convex, convergence rate is faster.

Theorem: Under Lipschitz condition and strong convexity, gradient descent with fixed step size \( t \leq 2/(\mu + L) \) 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 \( L/\mu \). (\( c \downarrow \) and condition number \( \uparrow \) implies slow convergence)
GD under Lipschitz gradient and strong convexity

- Let $g(w) = \nabla f(w)$.

\[
\begin{align*}
\| w_{k+1} - w^* \|^2_2 &= \| w_k - w^* - t(g(w_k) - g(w^*)) \|^2_2 \\
&= \| w_k - w^* \|^2_2 - 2t \langle g(w_k), w_k - w^* \rangle + t^2 \| g(w_k) \|^2_2 \\
&\leq (1 - t\mu) \| w_k - w^* \|^2_2 - 2t(f(w_k) - f(w^*)) + t^2 \| g(w_k) \|^2_2 \\
&\leq (1 - t\mu) \| w_k - w^* \|^2_2 - 2t(f(w_k) - f(w^*)) + 2t^2 L(f(w_k) - f(w^*)) \\
&= (1 - t\mu) \| w_k - w^* \|^2_2 - 2t(1 - tL)(f(w_k) - f(w^*)) \\
&\leq (1 - t\mu) \| w_k - w^* \|^2_2
\end{align*}
\]
GD under Lipschitz gradient and strong convexity

- Applying recursively,

\[ \| w_{k+1} - w^* \|_2^2 \leq (1 - t\mu)^{k+1} \| w_0 - w^* \|_2^2. \]

- By quadratic upper bound,

\[ f(w) - f(w^*) \leq \frac{L}{2} \| w - w^* \|_2^2 \]

Combining, for \( t \leq 2/(\mu + L) \),

\[ f(w_k) - f(w^*) \leq \frac{c^k L}{2} \| w_0 - w^* \|_2^2 \]
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 \( E(\|g_i(w)\|^2) \leq G^2 \).

\[
E(\|w_{t+1} - w^*\|^2) \\
= \|w_t - w^*\|^2 - 2\epsilon_t E(\langle g_i(w_t), w_t - w^* \rangle | w_t) + \epsilon_t^2 E(\|g_i(w_t)\|^2 | w_t) \\
\leq \|w_t - w^*\|^2 - 2\epsilon_t E(f(w_t) - f(w^*) | w_t) + \epsilon_t^2 E(\|g_i(w_t)\|^2 | w_t)
\]

- Taking marginal expectation

\[
E(\|w_{t+1} - w^*\|^2) \leq E(\|w_t - w^*\|^2) - 2\epsilon_t E(f(w_t) - f(w^*)) + \epsilon_t^2 G^2
\]
Convergence of SGD

- Applying the inequality recursively

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

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

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

- Letting \( \bar{w} = \sum_{t=1}^{k} \epsilon_t^* w_t \), where \( \epsilon_t^* = \frac{\epsilon_t}{\sum_{i=1}^{k} \epsilon_i} \), one can show that

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

- Alternatively,

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

- These bounds are not about \(w_k\), but gives some intuition about how \(\sum_{i=1}^{k} \epsilon_i\) and \(\sum_{i=1}^{k} \epsilon_i^2\) should behave.

- With fixed step size \(\epsilon\),

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

As \(k \to \infty\), \(\min_t f(w_t) - f(w^*) \leq \frac{G^2\epsilon}{2}\).
SGD for strongly convex objective with a fixed step size

Bottou et al. (2016) show a convergence analysis of SGD with fixed and diminishing step size for strongly convex objective function. To start, the following assumptions are made:

(A1) the objective function \( f : \mathbb{R}^d \rightarrow \mathbb{R} \) is continuously differentiable and the gradient \( \nabla f : \mathbb{R}^d \rightarrow \mathbb{R}^d \), is Lipschitz continuous with Lipschitz constant \( L > 0 \).

(A2) Strong convexity for \( f \): There exists a constant \( \mu > 0 \) such that

\[
f(\tilde{w}) \geq f(w) + \nabla f(w)^T(\tilde{w} - w) + \frac{\mu}{2}\|\tilde{w} - w\|^2_2.
\]

RHS has minimum at \( \tilde{w} = w - \frac{1}{\mu} \nabla f(w) \). Plugging in, we obtain

\[
\frac{1}{2\mu} \|\nabla f(w)\|^2_2 \geq f(w) - f(w^*). \quad (*)
\]
(A3) For the sequences \( \{ w_k \} \), \( k \in \mathbb{N} \) contained in an open set over which \( f \) is bounded below, there exist scalars \( \eta_G \geq \eta > 0 \), \( M \geq 0 \) and \( M_V \geq 0 \) such that, for all \( k \in \mathbb{N} \),

\[
\nabla f(w_k)^T E[g(w_k) | w_k] \geq \eta \| \nabla f(w_k) \|_2^2
\]

\[
\| E[g(w_k) | w_k] \|_2 \leq \eta_G \| \nabla f(w_k) \|_2,
\]

and

\[
\text{Var}[g(w_k) | w_k] \leq M + M_V \| \nabla f(w_k) \|_2^2.
\]

(A3) bound the first and second moments of the gradient. Define \( M_G := M_V + \eta_G^2 \). We assume \( M_G \geq \eta^2 \).
Theorem

Under the Assumptions (A1)-(A3), the SGD with a fixed stepsize, $\alpha_k = \bar{\alpha}$ for all $k \in \mathbb{N}$, satisfying

$$0 < \bar{\alpha} \leq \frac{\eta}{L M_G}$$

has the expected optimality gap satisfying the following inequality for all $k \in \mathbb{N}$:

$$\mathbb{E}[f(w_k) - f(w^*)] \leq \frac{\bar{\alpha} L M}{2 \mu \eta} + (1 - \bar{\alpha} \mu \eta)^{k-1} \left\{ f(w_1) - f(w^*) - \frac{\bar{\alpha} L M}{2 \mu \eta} \right\}$$

$$\longrightarrow \frac{\bar{\alpha} L M}{2 \mu \eta}, \quad \text{as} \quad k \to \infty.$$
Proof

- By the Lipschitz continuity of $\nabla f$,

$$f(w_{k+1}) - f(w_k) \leq \nabla f(w_k)^T (w_{k+1} - w_k) + \frac{1}{2} L \|w_{k+1} - w_k\|^2$$

$$= -\bar{\alpha} \nabla f(w_k)^T g(w_k) + \frac{1}{2} \bar{\alpha}^2 L \|g(w_k)\|^2.$$ 

- Taking conditional expectation and by the assumption (A3),

$$E[f(w_{k+1}) - f(w_k)|w_k]$$

$$\leq -\bar{\alpha} \nabla f(w_k)^T E[g(w_k)|w_k] + \frac{1}{2} \bar{\alpha}^2 L E \left[\|g(w_k)\|^2 |w_k\right]$$

$$\leq -\bar{\alpha} \eta \|\nabla f(w_k)\|^2 + \frac{1}{2} \bar{\alpha}^2 L E \left[\|g(w_k)\|^2 |w_k\right]$$

$$\leq -(\eta - \bar{\alpha} LM_G/2)\bar{\alpha} \|\nabla f(w_k)\|^2 + \bar{\alpha}^2 LM/2.$$
 optimization

Proof

- Bounding $E[f(w_{k+1}) - f(w_k)|w_k]$:

$$E[f(w_{k+1}) - f(w_k)|w_k] \leq - (\eta - \bar{\alpha}LM_G/2)\bar{\alpha}\|\nabla f(w_k)\|^2 + \bar{\alpha}^2 LM / 2$$

$$\leq - \frac{1}{2} \bar{\alpha}\eta\|\nabla f(w_k)\|^2 + \bar{\alpha}^2 LM / 2$$

$$\leq - \mu \bar{\alpha}\eta\{ f(w_k) - f(w^*) \} + \bar{\alpha}^2 LM / 2.$$ 

- The last inequality is due to strong convexity, (*) in (A2).

- Subtracting $f(w^*)$ from both sides, taking expectations, and rearranging yield

$$E[f(w_{k+1}) - f(w^*)] \leq (1 - \bar{\alpha}\mu\eta)E[f(w_k) - f(w^*)] + \bar{\alpha}^2 LM / 2.$$
Proof (cont’d)

- Subtracting $\bar{\alpha}LM/(2\mu\eta)$ from the both sides,

$$E[f(w_{k+1}) - f(w^*)] - \frac{\bar{\alpha}LM}{2\mu\eta}$$

$$\leq (1 - \bar{\alpha}\mu\eta) E[f(w_k) - f(w^*)] + \frac{\bar{\alpha}^2 LM}{2} - \frac{\bar{\alpha}LM}{2\mu\eta}$$

$$\leq (1 - \bar{\alpha}\mu\eta) \left\{ E[f(w_k) - f(w^*)] - \frac{\bar{\alpha}LM}{2\mu\eta} \right\}.$$  

- Lastly, since $\bar{\alpha} \leq \eta/(LM_G)$,

$$0 < \bar{\alpha}\mu\eta \leq \frac{\mu\eta^2}{LM_G} \leq \frac{\mu\eta^2}{L\eta^2} = \frac{\mu}{L} \leq 1.$$  

- Thus, $E[f(w_k) - f(w^*)] \to \frac{\bar{\alpha}LM}{2\mu\eta}$ as $k \to \infty$. □
Strongly Convex Objective, Diminishing Stepsize

- With equal sized steps, the expected optimality gap does not vanish as \( k \) increases.

- Under assumptions (A.1), (A.2) and (A.3), the SGD method with step size sequence such that, for all \( k \in \mathbb{N} \), \( \alpha_k = \frac{\beta}{\gamma + k} \) for some \( \beta > \frac{1}{\mu \eta} \) and \( \gamma > 0 \) with \( \alpha_1 \leq \frac{\eta}{LM_G} \). Then, for all \( k \in \mathbb{N} \),

\[
E(f(w_k) - f(w^*)) \leq \frac{\nu}{\gamma + k}
\]

where \( \nu = \max\{\frac{\beta^2 LM}{2(\beta \mu \eta - 1)}, (\gamma + 1)(f(w_1) - f(w^*))\} \).
Allen-Zhu, Li and Song (2018) show why SGD can find global minima on the training objective of DNNs in polynomial time using two assumptions: the inputs are non-degenerate and the network is over-parameterized.

A key technique is to derive that, in a sufficiently large neighborhood of the random initialization, the optimization landscape is almost-convex and semi-smooth even with ReLU activations.

The theory applies to the widely-used but non-smooth ReLU activation, and to any smooth and possibly non-convex loss functions and to fully connected NN, CNN and ResNet.